AWS Certified ML

1 Data Engineering

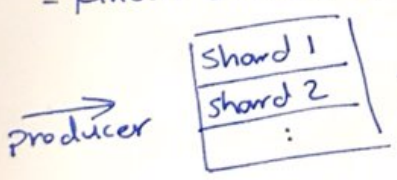
→ S3

- create a "Data Lake"
- you can do data partitioning < By date / By product
- data partioning in S3 can be identified automatically by AWS Glue crawlers
- S3 storage tiers ⇒ General Purpose: Standard (freq access) / intelligent / (infreq access) & One-Zone / Glacier ⇒ Archives
- S3 lifecycle rules ⇒ move data between diff tiers
  General Purpose → infreq Access → Glacier
- S3 Encryption ⇒ SSE-S3
  SSE-KMS
- S3 Security ⇒ user-based ⇒ IAM policies
  ⇒ resource-based ⇒ Bucket policies
- Use S3 bucket for policy to ⇒ grant public access to the bucket
  ⇒ force objects to be encrypted at upload
  ⇒ grant access to another account
- By VPC endpoint gateway → allow traffic to stay within your VPC instead of going through public web

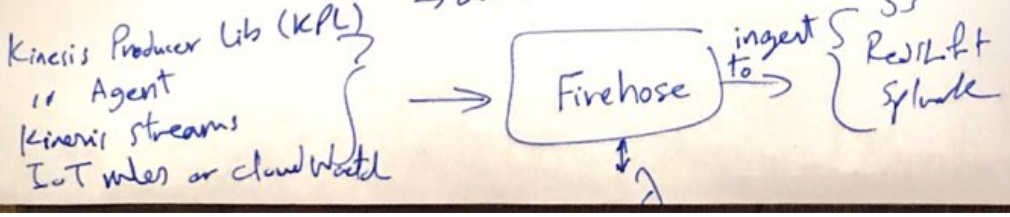
→ Kinesis (managed alternative to Apache Kafka)

Kinesis Streams: low latency ingest at scale
" Analytics: perform real-time analytics on streaming using SQL
" Firehose: load streams into S3 & RedShift & ElasticSearch & Splunk
" Video Streams: meant for streaming video in real-time

- Kinesis Streams are divided in ordered (shards) or partitions
  → have to be provisioned in advance



- Kinesis Firehose → near real-time
  → auto scaling
  → data conversions from CSV/json → Parquet/ORC



- Kinesis Analytics

use case → streaming ETL ⇒ select columns, make simple transformations, on streaming data

- Serverless
- SQL / Flink

→ ML on Kinesis Analytics
* Random cut forest : SQL function used for anomaly detection on numeric columns in a stream
* Hotspot : locate and return information about relatively dense regions in your data

- Kinesis Video Stream
↳ producer
Deeplens → Kinesis Video Stream
RTSP Camera →
→ Rekognition
→ SageMaker
→ EC2 (Tensorflow, MXNet)

→ Glue Data Catalog

S3 → index → Glue Data Catalog
→ Redshift spectrum
→ use schema Athena → QuickSight
→ EMR

- crawlers go through your data to infer schemas & partitions
- Glue crawler will extract partitions based on how your S3 data is organized
- Glue ETL ⇒ fully managed
Transformation ⇒ Find Matches ML : identify duplicate or matching records in your dataset

→ Redshift (OLAP)
↳ use Redshift spectrum to query data directly in S3 (no loading)

→ AWS Data Pipeline ⇒ orchestration service
⇒ destinations include S3 / RDS / EMR

RDS
DynamoDB
→ Data Pipeline
EC2
→ S3 → SageMaker

Exam Review Tips (Data Engineering)

- Content Type : application/x-recordio-protobuf if you want to use data in the protobuf recordIO format for training
- Training Input Mode parameter options while using SageMaker algorithms are File or Pipe
- Kinesis Data Streams PutRecord API uses name of the stream, a partition key and the data blob, whereas Kineris Data firehose PutRecord API uses the name of the delivery stream and the data record
- Redshift spectrum
- SageMaker Factorization Machines algorithm supports for inference x-recordio-protobuf & json
- Kinesis Producer Liberary (KPL) provides built-in performance benefits and ease of use on the client side
- Linear learner & factorization machines process the training data in recordIO-protobuf Float32
- IP insights (entity, IPv4 address) & object2Vec (token, seq) or (token, token) or (seq, seq)
- Using Crawler + Athena is the least effort way to make S3 data available for SQL queries
- AWS Data Pipeline & AWS Glue ETL ⇒ from Redshift to S3
- converting the data to recordIO-protobuf file type can significantly improve the traing time with marginal increase in cost
- XGBoost only supports text/csv & text/libsvm
- Apache MXNet recordIO is the recommended input format for SageMaker image classification algo.
- AWS Glue doesnot support Timestream as the source input type
- Glue ETL Transform job can perform deduplication in a serverless fashion
- Seq2Seq modeling and factorization machines support only the recordIO-protobuf file type for training data

[2] Exploratory Data Analysis (EDA)

- ordinal ⇒ such as rating
  numerical ⇒ such as ages
  categorical ⇒ " " gender

- Data Distributions
  → normal distribution
  → probability mass function
  → poisson distribution
  → Binomial "
  → Bernoulli "

- Time Series Analysis
  → Trends
  → Seasonality
  → time series = seasonality + trends + noise

- Amazon Athena
  → serverless interactive queries of S3 data (SQL)
  → no need to load data
  → Athena not for ETL, use Glue instead

- Amazon QuickSight
  → Biz analytics & visualizations in the cloud
  → Serverless
  → data sources: RedShift / RDS / Athena / S3
  → data sets are imported to SPICE (Super-fast, Parallel, In-memory, Calculation Engine)
  → ML insights: anomaly detection, forecasting, auto-narratives
  → QuickSight not for ETL, use Glue instead

- EMR
  → managed Hadoop framework on EC2 instances
  → EMR cluster

  → spot instances for temp capacity
  reserved " " long-running clusters

Note
  → SVD and PCA ⇒ to reduce dimensions
  → Kinesis → spark

- Feature Engineering
  → Median may be better choice than mean when outliers are present
  → imputing missing data: ML → KNN (numerical data)
    Hamming distance (categorical data) ↓ better
    → Regression → MICE DL

→ Handling unbalanced data
  → oversampling : duplicate samples from the minority classes
  → undersampling : instead of creating more positive samples, remove negative ones

  → SMOTE : Synthetic Minority Over-Sampling TEchnique
    ↳ artificially generate new samples of the minority using kNN

  → adjust thresholds when making predictions about a classification, if you have too many false positive, one way to fix that to simply increase that threshold

→ Handling Outliers
  → Variance measures how "spread-out" the data is
  → AWS's Random Cut Forest ⇒ made for outlier detection

- Bining : bucket observations together based on ranges of values

⇒ Note small batch size tend to not get stuck in local minima

- Regularization in DL : to prevent overfitting
  - Dropout
  - early stopping

- The vanishing Gradient problem
  ↳ How to fix it : ① Multi-level hierarchy
  ② LSTM
  ③ Residual N/w (ResNet
  ④ Better choice for activation → ReLU

- L1 & L2 Regularization
  L1 → sum of weights, used for feature selection, entire features go to zero
  L2 → sum of squared of weights, all features remain considered, just weighted

- Bagging : each resampled model can be trained in parallel
  Boosting : Training is sequential, each classifier takes into account the previous one's success

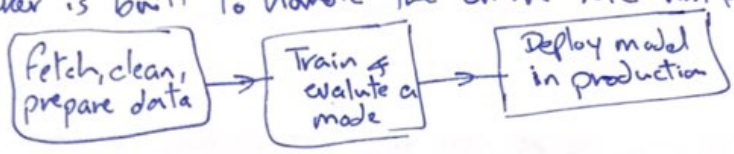
Exam Review Tips (EDA)

- The best way to engineer the cyclic features is to represent them as (x, y) coordinates on a circle using sin & cos functions
- Use exclude pattern `**.metadata` in the crawler definition to ignore the metadata. AWS Glue Crawler supports exclude pattern
- Removing stop words & tokenizing each word & lowering-case each word are the recommended pre-processing for NLP
- Elastic Search, EMR and EC2 are not "Serverless"
- Rule of thumb: drop a feature that will not help a model to learn. Any feature that has low variance or a lot of missing values or has a low correlation to the target label ought to be dropped
- IQR = Q3 - Q1
  minimum outlier cutoff = Q1 - 1.5*IQR
  maximum " " = Q3 + 1.5*IQR
- Histogram is best suited to show distributions of variables while bar charts are used to compare variables.
  Histogram plot quantitive data with ranges of the data grouped into bins or intervals while bar chart plot categorical data
- $tf("fox", d1) = \frac{2}{12}$
  $tf("fox", d2) = \frac{3}{12}$
  corpus $C = 2$
  $idf("fox", D) = \log \frac{2}{2} = 0$
  ↳ "fox" term appears in both documents

  $tf\text{-}idf("fox", d1, D) = tf("fox", d1) * idf("fox", D)$
  $= \frac{2}{12} * 0 = 0$
- Box plot & ~~scatter plot~~ histogram → to spot outliers
- LDA & NTM (Neural Topic Modeling) ⇒ for topic modeling
  PCA → reduce dimensionality
  k-Means → clustering

[3] Modeling

- Sage Maker is built to handle the entire ML workflow



- data must come from S3, ideal format varies with algorithm - often it's RecordIO - Protobuf
- SageMaker deploy in 2 ways.
  1. persistent endpoint for making individual predictions on demand
  2. Batch Transform to get predictions for an entire dataset

** SageMaker's Built-in Algorithms

1. Linear Learner ⇒ Regression / classification
   - training i/p ⇒ RecordIO-wrapped protobuf, float32 only
     ⇒ CSV
   - important hyperparameter:
     - Balance multiclass weights
     - learning rate, mini-batch-size
     - L1 regularization
     - L2 " (Wd ⇒ Weigh decay)
2. XGBoost ⇒ Reg. / class.
   - training i/p ⇒ CSV
     ⇒ libsvm
     ⇒ AWS extended it to accept recordIO-protobuf & Parquet
   - important hyperparam:
     - subsample → prevents overfitting
     - Eta → step size shrinkage, prevents overfitting
     - Gamma → min. loss reduction
     - Alpha → L1
     - Lambda → L2
3. Seq2Seq ⇒ ML Translation / Speech to text
   - training i/p ⇒ RecordIO / Protobuf (tokens must be integer)
     ⇒ must provide training data, validation data, and vocab files
   - important hyper:
     - batch size
     - optimize type (adam, sgd, rmsprop)
     - learning rate
     - # layer encoder
     - # " decoder
     - can optimize on ← accuracy / BLEU score / perplexity (cross-entropy)

④ DeepAR → forecasting 1D time series data

- training i/p ⇒ json
  ⇒ files can be in gzip or Parquet
- important hyper: - context length
  - epoch
  - mini batch size
  - learning rate
  - num cell

⑤ Blazing Text → Text classification (web searches, info retrieval, ranking) — supervised
→ Word2vec (unsupervised learning), word embeddings
→ only works on individual words, not sentences or document,

- training i/p ⇒ for text classification (supervised mode)
  ↳ one sentence / line
  ↳ --label-- (first "word" in sentence)
  ⇒ for word2vec (unsupervised) ⇒ skip-gram, CBoW, Batch_skip-gram
  ↳ text file with one training sentence per line
- important hyper ⇒ word2vec: - mode (batch_skipgram, skipgram, CBoW)
  - learning_rate
  - window rate
  - vector dim
  - negative samples

  ⇒ Text classification: - epoch
  - learning rate
  - word ngrams
  - vector dim

⑥ Object2Vec

- Note → currently support learning embeddings of pairs of tokens, pairs of sequences, and pairs of token and sequence

use cases → collaborative recommendation system
→ multi-label document classification
→ sentence ~~embedding~~ embedding

- training i/p ⇒ JSON lines

⑦ Object detection

- training i/p ⇒ RecordIO or image format (jpg or png)
- uses a CNN with SSD (Single Shot multiple Detector) algorithm
- important hyper: - mini batch size
  - learning rate
  - optimizer (adam, SGD, RMSPROP, ADADELTA)

(8) Image Classification
- input training ⇒ RecordIO or image files (jpg or png)

(9) Semantic Segmentation → Pixel-level
- training i/p ⇒ image files JPG, PNG for annotations
- choice of 3 algorithms → FCN
  → PSP
  → DeeplabV3
- choice of backbones → ResNet50
  → ResNet101
  → both trained on ImageNet

(10) Random Cut Forest → anomaly detection (unsupervised)
- training i/p ⇒ RecordIO - Prot.buf
  ⇒ CSV
- hyper : - num trees
  - num samples per tree

(11) Newral Topic Modeling → organize documents into topics (unsupervised)
- training i/p ⇒ RecordIO - protobuf
  ⇒ CSV
- hyperparameter : mini batch size & learning rate
  - num topic

(12) LDA ⇒ topic modeling algorithm (unsupervised)
→ can be used also in cluster customers based on purchases & harmonic analysis in music
- training i/p ⇒ RecordIO - protobuf
  ⇒ CSV
- hyperparameter : - num topics
  - alpha0

(13) KNN < reg. / class.
input training ⇒ RecordIO - protobuf
⇒ CSV

hyperparameter : - k
- sample size

(14) K-Means → unsupervised clustering
- i/p training ⇒ RecordIO - protobuf
  ⇒ CSV

(15) PCA → dimensionality reduction (unsupervised)
→ the reduced dimensions are called components
- i/p training ⇒ RecordIO - Protobuf
  ⇒ CSV

→ How PCA is used:
- Covariance matrix is created, then Singular Value Decomposition (SVD)
- Two modes → Regular = sparse data
→ Randomized = large "

→ hyperparameters : - Algorithm mode
- Substract mean

⑯ Factorization Machines < Reg. / class.
- good choice for tasks dealing with high dimensional sparse datasets, such as click prediction and item recommendation
⇒ training i/p ⇒ recordIO - protobuf Float32

⑰ IP insights → unsupervised learning for IP address usage patterns
→ identifies suspicious behavior from IP address
i/p training ⇒ (entity, IPv4 address) CSV only
↳ hashed and embedded

- Hyperparameter Tunning
↳ Automatic Model Tunning ⇒ Don't run too many training jobs concurrently, because it learns as it goes

---

[4] ML Ops

→ structure of a training container

```
/opt/ml
└ input
  └ config
  └ data
└ model
└ code
└ output
```

→ Structure of your Docker image

```
. WORKDIR
  nginx.conf
  predictor.py
  serve/
  train/
  wsgi.py
```

→ structure of a Deployment Container

```
/opt/ml
└ model
```

→ Protecting Data in-transit in SageMaker
↳ TLS/SSL
↳ inter-node training communication may be optionally encrypted, may increase time, and cost

---

→ SageMaker + VPC ⇒ Setup S3 VPC endpoint

→ Notebooks are internet-based by default
  ↳ if disabled, your VPC needs an interface endpoint (PrivateLink) or NAT GW to allow outbound connections, for training and hosting to work

→ Spot instances can be interrupted, use checkpoints to S3 so training can resume

→ Elastic Inference accelerators may be added alongside a CPU instance

→ inference pipelines:
- linear seq of 2-5 containers
- any combination of pre-trained built-in algorithms or your own algorithms in docker containers
- Combine (pre-processing → predictions → post-processing)
- Spark ML serialized into MLeap format

01003519014
01050019.14

Exam Reviews (Modeling)

- AUC/ROC metric doesn't require you to set a classification threshold. It's also useful when there's high class imbalance
- The residuals plot would indicate any trend of underestimation or overestimation. Both Mean Absolute Error and RMSE would only give the error magnitude. AUC is a metric used for classification models.
- Object2Vec can be used to find semantically similar objects such as ticket.
  Blazing text Word2Vec can only find semantically similar words
- PCA & Random Cut Forest are parallizable ***
- Transfer learning ⇒ 3rd party pre-trained mode → then remove the original classifier → then add the new classifier
- mode is the mandatory hyperparameter for Word2Vec & Text classification
- Object2Vec, Semantic segmentation and Seq2Seq modeling are NOT parallelizable
- AWS Glue job can't write output in recordIO-protobuf format
- When a model underfits, it exhibits low accuracy on both training & testing data
- Amazon Personalize is ML service to create individualized recommendations for customers using their app
- * predict() method for SageMaker Python SDK
  * create_endpoint_config() & create_endpoint() used in Boto3 Liberary while deploy the model
  * invoke_endpoint() to call inferences
- * XGBoost & Random Cut Forest → only use CPU
- * Seq2Seq → i/p is a seq of tokens (text, audio), and the o/p is another seq of tokens.
  for ex; machine translation, text summerization, speech-to-text
  → Seq2Seq, it uses RNN and CNN models with attention as encoder-decoder architectures
- create_transform() → to create Batch Transform job
  describe_transform() → to get a status about the progress of the job

- Think of L1 as reducing the number of features in the model altogether. L2 "regulates" the feature weights instead of just dropping them

Exam Review (ML Ops)

- How early stopping works in hyperparameter tunning job
  → after each epoch of training, get the value of the object metric
  → compute the running avg of the objective metric for all previous training jobs up to the same epoch, and then compute the median of all of the running averages
  → if the value of the objective metric for the current training job is worse than the median value of running avg of the objective metric for previous training jobs. SageMaker stops the current job
- object Detection, Image classification, Semantic segmentation
- Using the training data in protobuf recordIO along with Pipe mode can significantly improve the training job performance
- Training image & inference Image Registry paths are region based
- VPC has an interface endpoint (PrivateLink) + NAT GW for outband
- num_round & num_class are mandatory hyper for XGBoost
- SageMaker models are store in model.tar.gz format in S3 bucket
- Within an inference pipeline model, Amazon SageMaker handles invocations as a sequence of HTTP Req.
- N/W isolation is not supported by the following managed Amazon SageMaker containers as they require acces to S3 → Pytorch, Reinforcement learning, Sckitlearn, Chainer
- People pathing by AWS Rekognition
- There's no 15 min gap for logs to appear on CloudWatch. The default IAM has the required permissions to writes logs into cloudWatch
- SageMaker supports t, m, c, p and r family instances
- predict() → real-time inference
  deploy(), fit() → for models hosted on SageMaker Hosting Services
- SageMaker inference pipelines can be considered for real-time prediction & process batch Transform
- SageMaker Neo supports image classification
- PCA (randomized mode) for larger datasets