

AWS Certified Machine Learning Specialty Master Cheat Sheet

Distributions

- PDF continuous (normal distribution)
- PMF (mass) discrete
- Poisson – series of events where the average number of successes or failure are known. Poisson = discrete
- Binomial multiple 0/1 trials
- Bernoulli = special case of binomial where we have ONE trial

Time Series

- Know the difference between Seasonality vs. Trends
- Noise is present in time series
- Additive
- Multiplicative (scales with trends)

Machine Learning Concepts

3 Categories

- Supervised – Pre-labeled data
- Unsupervised – Find groupings, clusters by itself
- Reinforcement Learning – e.g. AI for video games. Action / Reward. Learn through trial and error.
- Have set of states, and set of actions for that state and a value Q
- Add or subtract from that value Q to weigh it

Optimization

- AKA Gradient Descent, an optimization algorithm for many ML algorithms
- Find the minima of the “sum of squares” error
- Find slopes where we become less and less steep (the gradient is closer to 0). We take smaller and smaller steps as we get closer to a zero gradient
- We can get trapped in local minima
- Set Learning rate = step size affects how long it takes for us to reach the “minima”

Regularization

- Use when the model overfit. **Prevent overfitting**
- Use fewer neurons, or fewer layers
- **Dropout**, remove neurons at random
- **Early stopping**, stop say at epoch 6 instead of epoch 10
- Uses regression to compute.
- Desensitize the data to a particular dimension.

- L1 regularization vs L2
- L1 is sum of weights, L2 is sum of weights²
- L1 performs feature selection, some features can go to 0, can reduce dimensionality, more computationally intensive
- L2 nothing goes to zero. Computationally efficient. Use L2 when we think all features are important

Hyperparameters (we set before training starts) vs **parameters** (internal to model that get tuned during training)

Hyperparameter

- Learning rate 0-1, step size
- Batch size, number of samples to train at a time
- Epoch, number of times we will process the training data

Cross Validation

- Don't hold out specific records for validation
- Retrain by repartitioning and holding out a % of the data for validation each time
- k-fold cross validation

Feature Selection and Engineering

- Selecting relevant data to be trained on
- Remove unneeded data (low correlation, low variance, missing data) we don't need to train our model. Makes model training faster and hopefully more accurate
- Feature correlation, e.g. Age and Height
- Selection requires trial and error, and domain knowledge
- Engineering new features based on existing features. E.g. Height/age, or pulling the weekday from a date
- PCA and K-means clustering (both unsupervised) can help reducing the feature set

Principal Component Analysis PCA

- **Unsupervised** learning algorithm
- Used for **data preparations pre-processing**, looks for relationships between data using **dimension reduction**
- Find central point of all data on n-dimensional graph
- Turn that point into the origin on the graph
- Draw a minimum bounding box around all of the points
- Longest length of the box is PC1, next longest is PC2, etc.
- Take out dimensions that don't affect the data much
- Project higher dimensional data into lower dimensional (like a 2d plot)

Missing and Unbalanced Data (Imputation)

- Impute a value that is missing, take the mean (Mean replacement of the column). Median might work as well. But this isn't very great TBH
- Remove the sample altogether
- Remove the column or feature altogether
- Unbalanced – outliers
- Outlier detection – random cut forests (AWS developed algorithm)
- 1 – 2 std dev. Std dev = $\sqrt{\text{variance}}$. variance = $(\text{each point} - \text{mean})^2 / \text{number of samples}$
- Unbalanced – not enough examples for all of our classes
- Can create fake data “synthesize data” using expert domain knowledge to create more examples for your class
- Actual good imputation methods
- K nearest neighbours (numerical data)
- Deep learning

- Regression (Mice is an alg to do this) multiple imputation by chained equations

Unbalanced Data

- Discrepancy between the number of positive and negative cases
- Oversample the minority case (works ok)
- Undersample (remove) the majority cases (not that great)
- SMOTE synthetic minority oversampling technique – uses KNN to artificially generate minority cases
- Choose a different threshold for classification, change the mix of FP and FN

Label and One Hot Encoding

- Convert labels to numbers i.e. a lookup table
- One hot encoding, used for categories – e.g. country to a number. E.g. one column with 3 countries values become 3 columns with 0/1 True False values

Binning

- E.g. put age 20-30 into a category, 30-40 into another
- Quantile binning = all bins have same number of records in them

Splitting and Randomization

- Training data
- Validation (tune hyper-parameters)
- Testing
- Always randomize the order of the training data, to get rid of any biases we may have introduced during the collection period
- Testing data should also be picked randomly
- Randomize before doing anything

RecordIO

- AWS format
- Pipe mode, streams data so we don't need to submit records individually
- Faster training throughput
- Best format for SageMaker

Vanishing gradient problem (ways to tackle them)

- Multi level hierarchy – break up your NN into sub networks that are trained individually
- LSTM
- Residual Networks (ResNet)
- ReLu activation function

Gradient Checking

- A debugging technique
- Used to numerically check the derivative values
- Used to validate NN code

ML Algorithms

Logistical Regression

- Supervised
- Binary yes no output
- Fit a sigmoid function (S shaped), less likely to be skewed by outliers

Linear Regression

- Supervised
- Numeric output

SVMs

- Supervised
- Classification output
- Partition into groups with furthest distance
- Where's the best line or hyperplane to separate two classes?

Decision Trees

- Supervised
- Binary, Numeric and Classification output
- Root node is one with most correlation with the label

Random Forests

- Supervised
- Binary, Numeric and Classification output
- A collection of decision trees
- DTs have a drawback, inaccuracies
- RF makes DTs more accurate
- RF picks random features and ignores the other features. Builds a DT. Repeat this so that we get many DTs
- We run a record through all of the trees to get our result. Then we vote based on all of the results

K-Means

- Unsupervised clustering
- Classification
- K is the number of classes we want to find
- Tries to find centre points for each cluster until we reach equilibrium
- What value of K should we use?
- Use variation, least variation wins
- Plot the reduction in variation vs. number of clusters
- This graph looks like an elbow plot. The elbow's number of clusters is what we want

K-Nearest Neighbour

- Supervised
- Classification
- Often used after K-means

- Used to classify new data based on clusters
- Uses K-number of nearest neighbours to classify
- E.g. $k=7$, classify based on the classes of the 7 nearest neighbours

Latent Dirichlet Allocation (LDA)

- Unsupervised
- Classification or Other
- Used for **text analysis**
- Text classification, topic discovery, document tagging, sentiment analysis
- **Documents** are made of **Topics** and made of **Words**
- Collection of Documents is a **Corpus**
- Remove stop words, “and but if”
- “Stemming” Learning, learnability -> learn
- Tokenize (turn words into an array)
- Choose the number of topics (k)

Deep Learning

Activation Functions

- Linear (can't do backprop, no derivative). Does 'nothing' just outputs the input that was given
- Binary step functions – don't work with derivatives_|-
- We want non-linear activation functions:
- Sigmoid (Logistic)
- TanH (more widely used than sigmoid), centred around 0. Good for RNN
- ReLU rectified linear unit (looks like this _/). Very popular, fast computation
- Leaky ReLU, other variants
- Softmax – often the final output layer of classification problem. Converts outputs to probabilities of each classification. Only outputs ONE label
- Sigmoid can output more than one label, e.g image has X and Y

Neural Networks

- Input layer, hidden layers, output layers
- Activation function inside hidden layers introduces nonlinearity,
- sigmoid (0 to 1), ReLU family (most common), Tanh (-1, 1)
- ReLU (0 to 1) piecewise, looks like: _/
- A Bias is introduced in hidden layers to prevent a 0 value, as $0 * \text{anything}$ is 0, and keep this neuron “active” in the network
- Adjusting bias and weights is how we tune a NN
- Forward pass
- Back propagation to optimize a loss function using Gradient descent
- Forward + backwards = 1 epoch

Convolutional NN

- Supervised, classification
- Used when we don't know where to find our feature, e.g. find something in an image, find features in a sentence
- Used in **image classification**
- Hidden layers are convolutional layers
- A convolutional neuron does more than just an activation function
- Combine multiple filters and those calculations form the value that is output from the neuron
- Filters can be pre trained

Recurrent NN

- Supervised, other output (time series data, voice recognition, translation)
- RNN can remember a bit, a small amount of data from past inferences
- Deals with **sequences in time, or sentences**, stock behavior, website logs
- Long short term memory (LSTM) and Gated Recurrent Units (GRU) are RNNs. These solve the issue where the RNN is more biased towards more recent data compared to earlier data
- GRU slightly less performant but trains faster as it is simpler
- LSTM more performant but more computationally expensive
- The previous input into the neuron at time 1 is an input for the next activation at time 2. It is a "memory cell"

Learning Rate

- Is a hyperparameter
- Too high, can **overshoot the minima and miss the optimal solution** (potentially)
- Too low means we take a ton of epochs to reach the minima (**increase training time**)

Batch Size

- # of training samples used in each epoch
- Smaller batch sizes tend to **not get stuck in local minima** when compared to large batch sizes
- Large batch sizes can **converge on the wrong solution** at random

Model Performance and Optimization

Confusion Matrix

- Visualize testing on models
- Which model should we use? We need to test different models

- Good – True positives, correctly predict position from positive
- Good – True negatives
- Bad – FP, FN
- Compare confusion matrix that are generated from different algorithms
- Can be 3x3, 4x4 etc.

Sensitivity (Recall) and Specificity

- Sensitivity = true positive rate = **recall** = $TP/(TP+FN)$, closer to 1 is better, less false negatives
- Specificity = true negative rate = $TN/(TN+FP)$, closer to 1 is better, less false positives
- False positives more acceptable in your business case? E.g. marking non fraud case is OK as long as we can catch all fraud = **High sensitivity**. Or medical, making a false positive is OK, as we can follow up to check.
- **High specificity**, classify video content so that we don't allow child to watch adult content. False positives are unacceptable. We prefer false negatives, i.e. video is suitable but we don't allow it
- **Sensitivity** = cast a net, I want to catch ALL THE FISH in the lake. I don't care if I catch frogs, bugs etc as long as I get all of the fish
- **Specificity** = cast a net, I ONLY want to catch fish, I don't want my net to have other things in it, but I'm ok with only catching some, and not all of the fish in the lake

Accuracy and Precision

- Accuracy = how right am I overall = $(TP + TN) / \text{Total}$, 1 may indicate overfit
- Precision = proportion of actual positives that were identified (subset of accuracy) = $TP/(TP+FP)$ (all positives in the calc)

ROC/AUC

- How to set threshold, or cutoff point for sensitivity vs specificity
- Build lots of confusion matrices and graph Sensitivity vs (1- specificity)
- It's a graph from 0 to 1
- That line is receiver operating characteristics. Look for the knee points
- It allows us to find the best model for max specificity and max sensitivity
- AUC is area under curve is how well this model performs. More area is better, max area is 1. More AUC means this model is better
- ROC – balance between sensitivity and specificity
- AUC – compare different models in terms of their separation power. 0.5 is useless as its the diagonal line. 1 is perfect

Gini Impurity

- Information gain algorithm to see how to create the first node and make the best split in a decision tree
- $1 - (\text{probability of class A})^2 - (\text{probability of other})^2$

- Then calculate a weighted avg using the total numbers and the two gini impurities
- Repeat for all of the features that exist
- Compare the weighted gini impurity. Lowest is better. It best separates the classes.

F1 Score

- Often used as a replacement for accuracy
- F1 combines recall and precision, takes into account more than accuracy
- $2/(1/\text{recall} + 1/\text{precision})$ or $2TP/(2TP+FP+FN)$
- Higher is better
- Use when we care about both precision and recall

TF-IDF

- Term frequency and Inverse Document Frequency
- What terms are most relevant for a document
- Term Frequency – how often a word appears in a doc
- Document Frequency – how often a word appears in ALL DOCS (this can let us get rid of words used everywhere like “and”, “But”)
- TF / DF or $TF * IDF$. $IDF = 1/DF$
- Unigrams, Bigrams, n-grams
- I love certification exams
- unigrams = every word individually
- Bi-grams = every two consecutive words “i love” “love certification” “certification exams”

Ensemble Learning

- Bagging – generate new training sets with random sampling with replacement
- Boosting – training sets will have weights, and as we retrain the weights will change
- Boosting generally has better accuracy, Bagging avoids overfitting

Machine Learning Tools and Frameworks

Jupyter

- Sagemaker is Jupyter as a service
- Runs boxed environments with separated dependencies

ML and DL frameworks

- Keras is an easier way to access Tensorflow (Google)
- AWS is MXNet and Gluon. Gluon is an abstraction of MXNet (like Keras)
- Pytorch and Scikit learn

Tensorflow

- Graph object is like an array that you populate with code lines from top to bottom
- You store constants, variables with no assigned value yet (placeholder), into the default graph of tensorflow
- You can add operations like multiply, add to the graph sequentially
- You can run the graph using Sessions

PyTorch

- Create a tensor which is a multi dimensional array with zeros
- You can just do simple operations like add, multiple on those matrices directly
- You need to add requires_grad=True to add memory to store the order of operations that were done for back propagation purposes
- Graph is created on the fly

MXNet

- Graph is created on the fly
- Similar to PyTorch, turn on the autograd feature to get backprop

Scikit Learn

- Has lots of built in datasets to use
- Support for less popular models. The other focus more on neural nets

Pandas

- Dataframe = table
- Manipulate data

AWS

S3

- Use as a "Data lake"
- Kinesis->S3->Athena->SageMaker
- Athena perform queries against S3 using SQL
- AWS Glue service
- Security and encryption

Glue

- Glue Data Catalog
- Metadata repository for datatables in S3
- Integrates with Athena and Redshift
- Glue Crawler
- Can crawl your data and discover the schema
- Can figure out folder partitions
- First create the crawler
- requires an IAM role to access resources in AWS
- The crawler can run on demand or a repeated basis
- Glue ETL
- Has bundled "transformations" like dropnulls, dropfield, and a ML one called "FindMatches" which can detect duplicate records
- Transform data from 3 different sources: databases ODBC, S3, or dynamodb
- Frequently used as an input to Athena
- We can't access the actual data from Glue, for that we need to query it in Athena
- Glue vs. Data Pipelines: DP is an orchestration service, it doesn't do the actual ETL for you
- Glue is completely serverless

Athena

- SQL interface into S3 data lakes
- Works on many different formats, json, csv, parquet etc
- Save query results back into S3 – preprocess data before ML
- After running a query, a csv file is created in a auto-generated bucket which stores the query result
- Can save queries
- Serverless
- Can create tables or views from queries
- Works well with Glue Catalog

Amazon Quicksight

- Visualize data sources, end user targeted. Should use federated auth
- Not AWS service really
- Create dashboards, reports, graphs
- Quicksight can natively connect to many AWS areas for data use

Kinesis: Data Streams, Data Firehose, Video Streams and Data Analytics

- Ingest lots of data real time, large datasets, iot
- Firehose is a **delivery/ingestion service** to **send into some permanent storage**, one of 4 places: S3, Redshift, Elasticsearch, Splunk (or kinesis data analytics)
- Fully managed NEAR real time
- Can auto convert CSV and JSON to Parquet or ORC when destination is S3 (using Glue)

- Can compress gzip, zip, snappy when destination is S3
- Serverless means you must use Lambda to do data conversions 0 there are a bunch of blueprints we can use
- Data streams is generic endpoints (consuming applications), it can stream into EMR/Spark, Lambda, Kinesis Data Analytics. Requires setting up shards. More shards = more capacity, ~1mb/sec/shard. Real time
- Can write custom code for producer and consumer – create **real time applications**
- Data Analytics – Process Kinesis data Streams or Firehose using **SQL**, Flink, Java
- Select from Stream1...
- Input stream is Kinesis data stream or firehose
- Has output stream and error streams
- For streaming ETL
- SQL can create from template
- Remember: can do SQL function RANDOM CUT FOREST for anomaly detection on columns in a stream
- HOTSPOT, detect dense areas in your data
- Firehose, can take in a never-ending stream of json, dump it into S3

EMR with Spark (Elastic Map Reduce)

- Managed Hadoop service for parallel compute tasks
- e.g. massive data sets that we need to normalize or transform before ML
- Spark is kind of a better “Mapreduce” for EMR
- Petabyte scale
- Integrated with S3 – we can use S3 as a mounted filesystem instead of HDFS (hadoop filesystem) through “EMRFS”
- Has a Master node, core nodes and task nodes inside the cluster
- Master node manages cluster
- Each node plays a different role
- Core nodes store data on hadoop filesystem
- Task nodes can be spot instances (optional node) as they don’t store data on filesystem
- Apache spark is a analytics engine, runs in an EMR cluster
- Spark runs in SageMaker and SparkML runs in EMR too
- Typical workflow, S3 -> EMR/Spark -> SageMaker

EC2 for ML

- Use computer optimized or Accelerated Compute (GPU instances)
- There’s a class of ml.* instances but those are only available from SageMaker
- Lots of AMIs preloaded with machine learning languages and libraries
- Conda libraries
- Bas AMIs with GPU libraries
- You must request limit increases to use any ML suitable compute instances

Batch

- Docker images, serverless

- Cloudwatch and step functions can trigger Batch

Data Engineering Summary

Here's a quick summary of all the services we've mentioned

- Amazon S3: Object Storage for your data
 - VPC Endpoint Gateway: Privately access your S3 bucket without going through the public internet
 - Kinesis Data Streams: real-time data streams, need capacity planning, real-time applications
 - Kinesis Data Firehose: near real-time data ingestion to S3, Redshift, ElasticSearch, Splunk
 - Kinesis Data Analytics: SQL transformations on streaming data
 - Kinesis Video Streams: real-time video feeds
 - Glue Data Catalog & Crawlers: Metadata repositories for schemas and datasets in your account
 - Glue ETL: ETL Jobs as Spark programs, run on a serverless Spark Cluster
 - DynamoDB: NoSQL store
 - Redshift: Data Warehousing for OLAP, SQL language
 - Redshift Spectrum: Redshift on data in S3 (without the need to load it first in Redshift)
 - RDS / Aurora: Relational Data Store for OLTP, SQL language
 - ElasticSearch: index for your data, search capability, clickstream analytics
 - ElastiCache: data cache technology
 - Data Pipelines: Orchestration of ETL jobs between RDS, DynamoDB, S3. Runs on EC2 instances
 - Batch: batch jobs run as Docker containers – not just for data, manages EC2 instances for you
 - DMS: Database Migration Service, 1-to-1 CDC replication, no ETL
 - Step Functions: Orchestration of workflows, audit, retry mechanisms
- Briefly mentioned:
- EMR: Managed Hadoop Clusters
 - Quicksight: Visualization Tool
 - Rekognition: ML Service
 - SageMaker: ML Service
 - DeepLens: camera by Amazon
 - Athena: Serverless Query of your data

Built in AWS ML Services

Rekognition

- Image and video analysis (object and scene detection)
- Pretrained deep learning
- Image moderation

- Facial analysis – age, gender, smiling etc
- Celebrity recognition
- Face comparison – can we find a face from this image in a target image
- Text in image (e.g. signs text)
- Video can be streamed or stored somewhere such as S3
- Stored video – S3 -> Lambda triggered on upload-> recognition
- Streaming video – Kinesis video stream -> recognition -> Kinesis data stream

Polly

- Text to speech
- Many languages, Female or male voices
- Upload a lexicon to customize pronunciation (read full acronyms)
- Can pass text in, in SSML format “speech synthesis markup language”, looks like XML. Like you can put in whisper effects, pauses

Transcribe

- Speech to Text (ASR – Automatic speech recognition)
- Real time or analyse pre-recorded files
- You can create custom vocabularies
- Put words in a text file, specify the language and upload it (or put it into S3)
- You can create transcription jobs
- Speaker identification

Translate

- Batch or real time
- Supports custom terminology you can pass in dictionaries in csv or tmx format

Comprehend

- Text analysis (NLP)
- Can train on our own data
- Features
- Keyphrase extraction
- Sentiment analysis (=ve, -ve, neutral, mixed)
- Syntax analysis (separate into pronouns, verbs, adjectives etc)
- Entity recognition (names, organizations, dates)
- Custom entities
- Language detection
- Custom classification (provide training data)
- Topic modelling
- Multi language support

Lex

- Powers Alexa
- Conversation interface service for **chatbots**
- Tries to understand intent from your speech
- Create a “bot”, can output voice or output text
- Then create Utterances (training data) and “Intent” (labels)

Forecast

- Time series forecasting

Service Chaining with AWS Step Functions

- Combining multiple AWS services to create a full solution out of the ML services
- **S3** for storage, trigger **Lambda**
- Translate => Comprehend
- Aws **step function** orchestrates multiple lambda functions together.
- State machine
- Step functions can pause or wait, e.g poll a service for status (for asynchronous services). Lambda has execution time limits, that we can get around using step functions
- It's logic between lambda functions

Sagemaker

- **Build train** and **deploy** ML models (3 stages of Sagemaker)
- Fully managed service
- End to end lifecycle of ML
- Lots of managed algorithms, we just choose hyperparameters
- Can access/control Sagemaker using the console (web), API (boto3), Python SDK and Jupyter notebooks
- Notebooks
- Have Notebook instance types, with ml. prefix
- You can still spin up other instances, you're not tied to your Notebook Instance Type
- You can give access to S3 buckets
- You don't have access to VPC by default unless you set them
- You access the notebook instance through a presigned url
- Lifecycle configurations are used to run bash commands that run before your notebook instance starts

Sagemaker Build

Data Preprocessing

- Visualize your data (notebooks)
- Explore data
- Feature engineering

- Synthesize (generate more training data for certain labels if we have less cases)
- Convert, e.g. images to recordIO, csv into something else
- Split data (validate, test, train)
- Structure

Amazon Ground Truth

- Build training datasets
- Reduce data labelling costs
- When we have data but it is not labelled
- Workflow includes humans to label (Mechanical Turk)
- Can also be private human team (internal to your company)
- You provide instructions to tell people how to label

Preprocessing image Data

- We don't have enough images, only 60 of a certain class, what can we do?
- Rotate and transform the images of that class to generate more training data, e.g. sharpen, colour contrasts

Algorithms

- 3 sources for Sagemaker
- Built-in to Sagemaker
- AWS Marketplace
- Custom
- Linear Learner
- Can do regression and classification
- Input: RecordIO (preferred), CSV. Inputs can be pipe (faster) or file mode
- Must normalize data first
- BlazingText
- **Text classification**, sentiment analysis, etc
- Used by Amazon Comprehend probably
- 2 modes: Word2vec or text classification
- Object2Vec
- turn objects into features
- Unsupervised, figure out similarity between objects
- Image Classification Alg (object detection is bounding boxes)
- Conv NN
- Image recognition (possibly powers Amazon Rekognition)
- Can use "transfer learning" i.e. build on an existing model
- K-Means
- Web scale k-means clustering algorithm
- Find discrete groupings within data (unsupervised algorithm)
- Latent Dirichlet Allocation (LDA)
- Text analysis, topic discovery
- Unsupervised
- Amazon Comprehend

- Principal Component Analysis (PCA)
- Reduce dimensionality (number of features)
- XGBoost
- Extreme gradient boosting – high performance decision tree algorithms
- Boosted group of decision trees
- Use to make predictions from tabular data
- Not deep learning algorithm
- Lots of hyperparameters
- Seq2Seq
- RNNs, input is a sequence of tokens and output is the same
- Machine translation, text summarization
- DeepAR
- Forecasting 1d time series data
- RNN
- Random Cut Forest
- Unsupervised
- **Anomaly detection**
- There's a lot more algorithms built in to sagemaker

Sagemaker Train

Architecture

- ECS + docker images
- Can create our own images
- docker image structure /opt/ml/code, opt/ml/model
- S3 (Training data) – or elastic file system, or FSx for Lustre
- Has “Channels” that need to be defined e.g. train, validation, model
- Channel tells what kind of data this is?
- EC2 instances (ML class) – we can't get into the OS of these
- P2 family is GPU
- Sometimes can elastically attach GPUs to an instance
- There's “spot instances” for training called “managed spot training”
- Can keep state using “checkpoints” in s3 if your instance is destroyed. It stops gracefully

Training an Image Classifier

- Create “**training job**” from sagemaker
- Requires a role, e.g. to get and write data from s3 buckets where our training data is
- Choose an algorithm (aka which ecr container)
- File input vs pipe input
- Accuracy metric are published into Cloudwatch
- Choose instance size. Some algorithms WILL require you to use GPU instances
- Max execution time
- VPCs
- Hyperparameters. A lot of defaults are set for us, but some require us to fill in e.g.
- Number of classes (neurons in output layer)

- Number of training samples
- Image dimensions, colour channels
- Input data configuration
- Channel name (train, validate, training labels, validation labels)
- Location (e.g. S3 or file system)
- Output data configuration
- S3 location

Hyperparameter Tuning

- Sagemaker auto parameter tuning as a service
- Choose an algorithm -> Set ranges of hyperparameters -> Choose metric to measure (e.g. maximize area under curve)
- Sagemaker will run a whole bunch of training models in parallel. There is a “tuning model” looking at the hyperparams

Sagemaker Deploy

Inference Pipelines

- Chaining models together
- Pass output of one model to be used as input to another model

Real-Time and Batch Inference

- Real-time inference has a SageMaker Endpoint (internal not public)
- We can call a model in real time to get the result (inference) by InvokeEndpoint from EC2, Lambda
- Batch inference – Create a “batch transform job” likely with data from S3. Push that S3 into the batch job, and then the output goes back into S3

Deploy

- Create a model definition
- Choose a IAM role, pass in the “training image” (ECR docker container) from the “Training Job”, the model S3 location
- Create an Endpoint configuration
- Point it to the model definition
- Create the Endpoint
- Name, choose the endpoint configuration
- We use the Endpoint to make inferences. Endpoints can’t be accessed publicly. You can access it from Lambda, or CLI, with the “aws sagemaker-runtime invoke-endpoint....” Command
- After invoking, the output is probably just an array of numbers, labels, whatever
- Accessing SageMaker Endpoints from an App
- AWS api/sdk à SageMaker Endpoint is one way
- API Gateway à Lambda à SageMaker Endpoint is another

Security

SageMaker Notebooks

- IAM policy – CreatePresignedNotebookInstanceUrl –
- Give notebook root access (server access)? Set this during creation default is true . Lifecycle scripts run as root.
- SageMaker instance profiles, e.g. to grant permissions to S3
- SageMaker doesn't support resource-based policies e.g. an S3 bucket policy
- From Notebooks, we can see S3 buckets, and files, but we can't copy them by default

SageMaker VPCs

- The default is a public VPC i.e. access to internet
- If we are in a private VPC, we need a S3 VPC endpoint to access S3

Other

- Horovod or parameter servers – how to do distributed training in tensorflow
- Production variants – how to do a/b (% traffic to A, % to B) testing of models using production data
- Amazon NEO is a cross compiler that lets you use models in different architectures

AWS Machine Learning-Specialty Summary:

-
- AWS course page: <https://aws.amazon.com/certification/certified-machine-learning-specialty/>
 - AWS exam guide: [https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty%20Exam%20Guide%20\(1\).pdf](https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty%20Exam%20Guide%20(1).pdf)
 - AWS sample questions: <https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty%20Sample%20Questions.pdf>
-

Domain 1: Data Engineering (20%)

Create data repositories for machine learning. Identify and implement a data-ingestion solution. Identify and implement a data-transformation solution. Opinion: IMHO this domain should be reduced to 15% or even 10%. I found the questions pretty repetitive, and they were about Big Data, not about Machine Learning. If you've already passed the Big Data Specialty certification, you'll be fine. If not, make sure you're very familiar with Kinesis and its different flavours, or you'll have a miserable time.

Domain 2: Exploratory Data Analysis (24%)

Sanitize and prepare data for modeling. Perform feature engineering. Analyze and visualize data for machine learning. Opinion: typical Data Science stuff, not really tied to any particular AWS service. Cleaning data, handling missing values, performing basic feature engineering. If you have hands-on ML experience, this won't be a problem at all. Questions don't go very deep. I was surprised to get a few questions on data viz, most of them pretty vague and awkward to answer without looking at any actual data. IMHO they should be dropped and replaced with more questions on feature engineering.

Domain 3: Modeling (36%)

Frame business problems as machine learning problems. Select the appropriate model(s) for a given machine learning problem. Train machine learning models. Perform hyperparameter optimization. Evaluate machine learning models. Opinion: a reasonable mix of high-level questions on framing business problems (algo selection, etc.), SageMaker-related questions (built-in algos, HPO, etc.) and Deep Learning questions (CNN, LSTM, regularization, etc.). Again, if you do this for a living and if you've spent some time with SageMaker, you should be fine. I didn't get any complex algorithm question, and none on specific Deep Learning frameworks (TensorFlow, etc.). IMHO, this could be a little more challenging than it is :)

Domain 4: Machine Learning Implementation and Operations (20%)

Build machine learning solutions for performance, availability, scalability, resiliency, and fault tolerance. Recommend and implement the appropriate machine learning services and features for a given problem. Apply basic AWS security practices to machine learning solutions. Deploy and operationalize machine learning solutions.

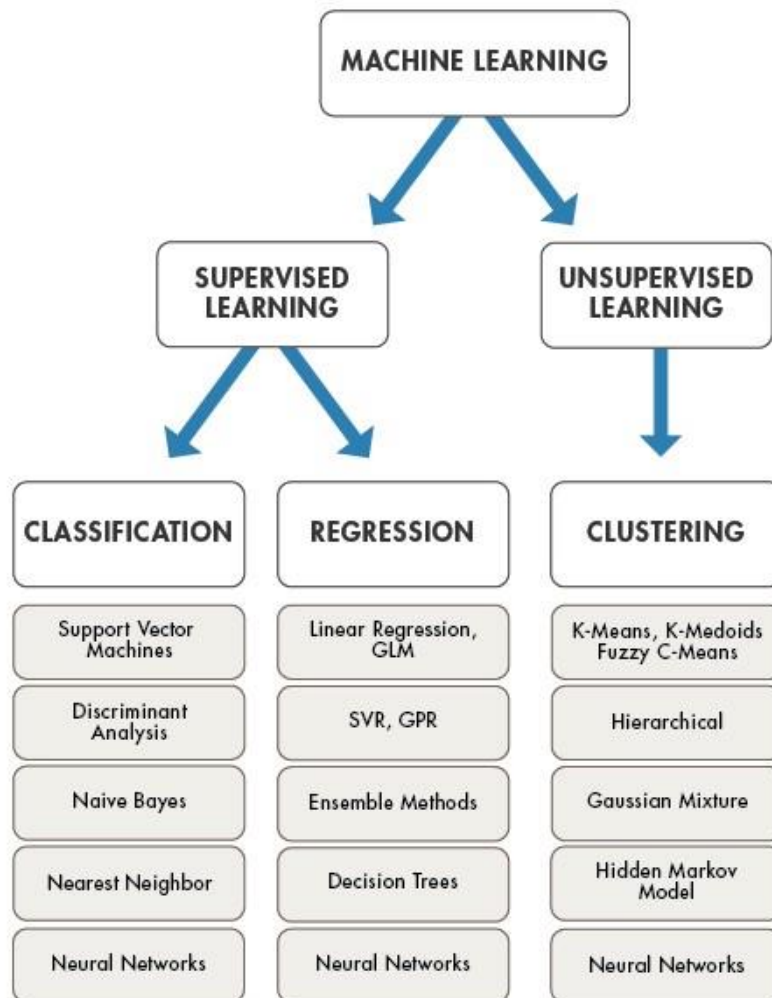
Query logging using Athena Cloudtrail integration with Athena Amazon Macie
Glacier Vault lock Quicksight Different e-mail protocols in secure
port <https://www.siteground.com/tutorials/email/protocols-pop3-smtp-imap/>

White paper


- https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf
 - <https://d1.awsstatic.com/whitepapers/aws-kms-best-practices.pdf>
 - https://d0.awsstatic.com/whitepapers/compliance/AWSSecurityatScaleLogginginAWS_Whitepaper.pdf
 - <https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf>
 - https://d1.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf
 - https://d1.awsstatic.com/whitepapers/Security/Secure_content_delivery_with_CloudFront_whitepaper.pdf
 - https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf
-


- AWS Sagemaker Developer Guide
 - <https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html> Wong
 - Tai Sin's recommended Sagemaker build-in9 big algorithm video
 - Learning Path and AWS university from AWS's official website:
 - <https://aws.amazon.com/training/learning-paths/machine-learning/exam-preparation/>
-

Machine learning






MACHINE LEARNING IN EMOJI



 SUPERVISED

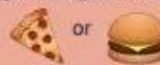
 UNSUPERVISED

 REINFORCEMENT

	SUPERVISED	human builds model based on input / output
	UNSUPERVISED	human input, machine output human utilizes if satisfactory
	REINFORCEMENT	human input, machine output human reward/punish, cycle continues

BASIC REGRESSION

	LINEAR	<code>linear_model.LinearRegression()</code> Lots of numerical data
	LOGISTIC	<code>linear_model.LogisticRegression()</code> Target variable is categorical



CLUSTER ANALYSIS

	K-MEANS	<code>cluster.KMeans()</code> Similar datum into groups based on centroids
	ANOMALY DETECTION	<code>covariance.EllipticalEnvelope()</code> Finding outliers through grouping



FEATURE REDUCTION

T-DISTRIBUTION STOCHASTIC NEIB EMBEDDING	<code>manifold.TSNE()</code> Visualize high dimensional data. Convert similarity to joint probabilities
PRINCIPLE COMPONENT ANALYSIS	<code>decomposition.PCA()</code> Distill feature space into components that describe greatest variance
CANONICAL CORRELATION ANALYSIS	<code>decomposition.CCA()</code> Making sense of cross-correlation matrices
LINEAR DISCRIMINANT ANALYSIS	<code>lda.LDA()</code> Linear combination of features that separates classes



CLASSIFICATION

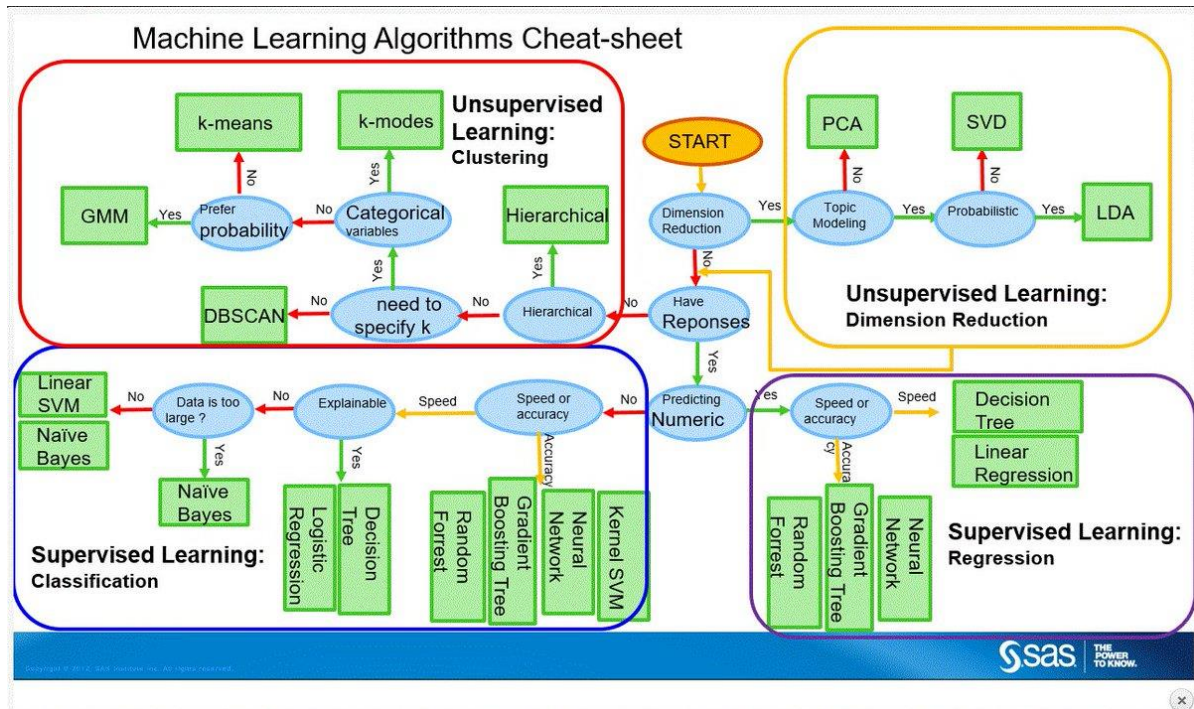
			NEURAL NET	<code>neural_network.MLPClassifier()</code> Complex relationships. Prone to overfitting Basically magic.
			K-NN	<code>neighbors.KNeighborsClassifier()</code> Group membership based on proximity
			DECISION TREE	<code>tree.DecisionTreeClassifier()</code> If/then/else. Non-contiguous data Can also be regression
			RANDOM FOREST	<code>ensemble.RandomForestClassifier()</code> Find best split randomly Can also be regression
			SVM	<code>svm.SVC()</code> <code>svm.LinearSVC()</code> Maximum margin classifier. Fundamental Data Science algorithm
			NAIVE BAYES	<code>GaussianNB()</code> <code>MultinomialNB()</code> <code>BernoulliNB()</code> Updating knowledge step by step with new info



OTHER IMPORTANT CONCEPTS

BIAS VARIANCE TRADEOFF	
UNDERFITTING / OVERFITTING	
INERTIA	
ACCURACY FUNCTION	$(TP + TN) / (P + N)$
PRECISION FUNCTION	$TP / (TP + FP)$
SPECIFICITY FUNCTION	$TN / (FP + TN)$
SENSITIVITY FUNCTION	$TP / (TP + FN)$

@emilynamillion made this



Model parameter :

Parameters are those which would be learned by the machine like Weights and Biases.

Hyperparameter :

Hyper-parameters are those which we supply to the model, for example: number of hidden Nodes and Layers, input features, Learning Rate, Activation Function etc in Neural Network,

Difference between model param vs HyperParam

- <https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/>

Learning Rate :

Determines the size of the step taken during gradient descent optimization ,Between 0 and 1

Batch Size :

- The number of sample used to train at any one time.
- Could be all one or some of your data (batch ,stochastic ,or mini-batch)
- Often 32 ,64 and 128

- Calculable from infrastructure

Epoches :

- The number of times that the algorithm will process the entire training data.
- Each epoch contains one or more batches
- Each epoch should see the model get closer to the desired state
- Usually a high number :
- 10,100,1000 and up

Disclaimer: All data and information provided on this site is for informational purposes only. This site makes no representations as to accuracy, completeness, currentness, suitability, or validity of any information on this site & will not be liable for any errors, omissions, or delays in this information or any losses, injuries, or damages arising from its display or use. All information is provided on an as-is basis.