AWS Machine Learning Specialty

- Sample questions from AWS
- Practice exam from AWS
- Udemy course with practice exam
- Udemy course with practice exam 2
- Udemy practice exams (difficult)
- Udemy practice exam 2
- Examtopics question dump

Table of contents

1. Data engineering

1.1 S3

- S3 Analytics: gives recommendations to when to transition objects to the right storage class. Only works for Standard and Standard IA. Creates a daily report
- Amazon FSx for Lustre: used to serve S3 training data to SageMaker, speeds up training and setup
- SageMaker can only read from S3. If data is in DynamoDB, use data pipeline to export to S3 as JSON, and then converted to CSV

- Data warehouse: only store structured data
- Data lake: structured, semi-structured and unstructured data

1.2 Glue

- **FindMatches** ML: de-duplicate data during ETL
- Can detect sensitive data and redact PIIs and sensitive information, at cell and column level
- Glue cannot write the output in RecordIO-Protobuf. For ETL for this type of output, use EMR
- Glue source input types: DocumentDB, Oracle, PostgreSQL -> Timestamp not accepted
- Python Shell supports Glue jobs relying on libraries such as numpy, pandas and sklearn

1.3 Kinesis data streams

- One shard can handle up to 1000 transactions/s or 1MB/s. If the file is 8kb, that means 8MB/s. So we need 8 shards
- number_of_shards = max(incoming_write_bandwidth_in_KB/1000, outgoing_read_bandwidth_in_KB/2000)
  - Outgoing bandwidth also plays a role in determining number_of_shards
- Stream can deliver to Glue, which can transfom files into Parquet, and deliver to S3
- Stream cannot convert files to Parquet on the fly
- it can retain data, but minimum only for 24h

1.4 Kinesis data firehose

- Buffer interval: wait the buffer time until data comes, and then send all of it downstream. max interval: 900s
- Can convert csv -> json, but not csv -> parquet
- Can ingest data and transform to parquet on the fly

- Can send data directly to Redshift with Redshift streaming ingestion

1.5 Kinesis Video Stream

- Provides video playback
- Data retention 1h - 10 years

Video stream allows one producer, all cameras compose one producer. Consumers can be:

- Rekognition
- SageMaker
- EC2 consumer tensorflow MXNet
- Fargate: can decode frames

Consumers store checkpoints and processing status in DynamoDB. Inference results can be sent to KDS.

1.6 Data pipelines

- ETL service (runs in EC2) to manage task dependencies. For example, it can move data from RDS to S3 weekly
- Data sources might be on-premises
- Highly available, retries and notifications on-failure

1.7 AWS Batch

- Serverless service to run batch jobs as Docker images
- *Dynamic provisioning of instances (EC2 and spot)*
- Automatically determines optimal quantity and type based on volume of jobs and requirements
- Can schedule Batch jobs using CloudWatch events, or using Step functions

1.8 Step functions

- Service to orchestrate workflows
- Supports advanced error handling and retry mechanism outside the code
- Supports audit of the workflow history
- Max execution time of 1 year

1.9 Redshift

- Redshift ML: to access the model with SQL commands. No need to move it to S3 and then SageMaker
- Redshift streaming ingestion: send data directly to Redshift from Kinesis Firehose

1.10 EBS

Can be used for storage for SageMaker instance, but only up to 16TB and it has not so high I/O. To optimize that, store data in S3 and use Pipe mode.

2. Feature engineering

- tf-idf: term frequency, inverse document frequency

  - Table dimensions: first axis is the amount of sentences, the other axis is the amount of *unique* unigrams + amount of unique bigrams, if that is what the analysis is based on
  - If one word appears in all sentences, idf = log(1) = 0 -> tf-idf = 0
- Quantile binning transformation: process to discover non-linearity in the variable's distribution by grouping observed values together
- t-SNE (t-distributed stochastic neighbor embedding): non-linear dimensionality reduction algorithm, similar to PCA (which is linear)
- MICE (multiple imputations by chained equations): algorithm to deal with missing data in dataset, works with categorical values
- SMOTE: uses a k-nearest neighbours approach to exclude members of the majority class while in a similar way creating synthetic examples of a minority class
- SG ground truth: manages human labeling, dynamically creates a model so fewer samples have to be labeled in the future. Labels are still labeled by your team, not 3rd party people

Batch size:

- If bigger -> fewer batches -> faster training -> might get stuck in local minima ---> you should increase learning rate
- If smaller -> helps optimization technique to jump local minima and explore other areas for global minima -> usually small learning rate, so slower training

Machine learning:

- Sigmoid is used for **binary** classification
- Softmax applies to multiclass problems, returns the probability for each label, and the sum of all probabilities adds up to a 1
- If after more layers the model is not converging, there's a vanishing gradient because of multiplying small derivatives from all the layers. Instead of using sigmoid, use relu
- L1 Regularization works by eliminating features that are not important
- L2 Regularization keeps all the features but simply assigns a very small weight to features that are not important
  - If model is underfitting (high error in training), reduce regularization
- If model is underfitting
  - Add more features
  - Reduce regularization
- If model is overfitting
  - Reduce features
  - Add regularization
  - Add dropout
  - Add early stopping
  - Add more training data
- How to handle outliers in data
  - Logarithmic transformation (normalizes in log)
  - Robust standardization (scales values taking the standard deviation into account)

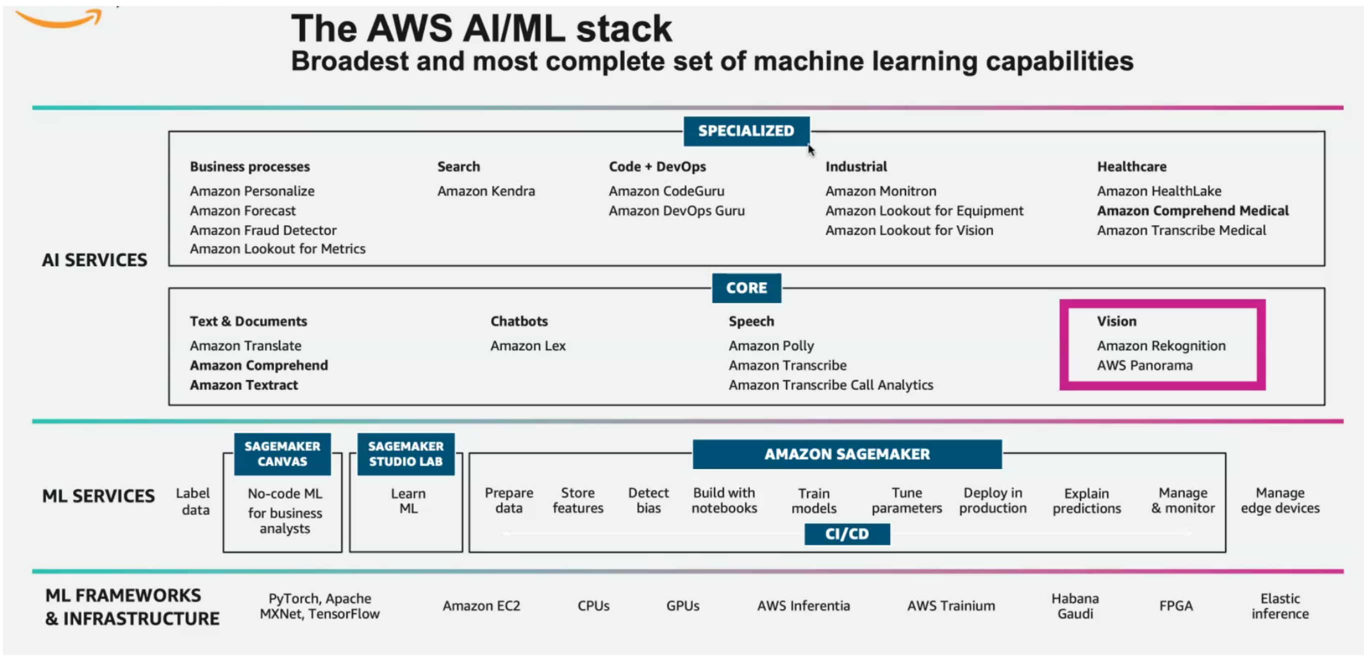
- In an imbalanced dataset where there are more negative samples than positive, change the cost function so that false positives have a higher impact on the cost value than false negatives

- K-fold cross validation: for imbalanced datasets, better to use stratified

Graphs

- To see geographical data: heatmap
- To spot outliers: box plot, histogram, scatter plot

3. Sagemaker



Supported file formats:

- For text: csv, libsvm
- For images: jpg, png

Instance types:

- M4, M5: for CPU
- P3: for GPU

- With lifecycle configuration, you can automate initial installation of libraries when creating a notebook
- Spot instances can be used for training: it saves a lot of money but increases training time. Uses EC2, they checkpoint to S3 so training can resume in case the instance gets interrupted
- Spot instances can be used to process anticipated traffic surges in the cheapest way when using EMR cluster. Use it for Spark task nodes only. Not for master and core nodes
- For training instances: accelerated computing instances come with GPUs
- Horovod: **distributed** DL framework for Tensorflow, keras, when we need several GPUs. Spark doesn't work with distributed training

3.1 Data input

Data input to SageMaker:

- File mode

- Uses disk space to store both final model artifacts and full training dataset
- Pipe mode
    - streams S3 data to the training container, improving the performance of training jobs
    - Accepts protobuf.recordIO format, not Parquet
    - Higher I/O
    - you reduce the size of the EBS volumes of the training instances. to optimize, convert data into protobuf.recordIO
    - It only needs to store the final model artifacts

3.2 SageMaker built-in algorithms

Built-in SG algorithms can't be edited. To use a custom architecture:

- Customize the SG container with your own code in TF framework, import it to ECR, use it for training
- Bundle your own Docker container with TF Estimator loaded with the network you want, import it to ECR, use it for training

Required parameters before starting a SG built-in job:

- Provide the ARN of an IAM role that SageMaker can assume to perform tasks on your behalf
- Specify the resources, ML compute instances, and ML storage volumes to deploy for model training
- Specify the output path on an Amazon S3 bucket where the trained model will persist

---

- Support vector machine
    - can be used for classification or regression
    - works with linear or non-linear problems
    - For radial non-linear clustering problems, kNN can handle this too
- Logistic regression
    - classification algorithm for categorical variables
- **Linear learner**
    - For classification and regression. Trains multiple models in parallel and selects the most optimal one
    - Preprocessing: automatically normalizes and **shuffles** data. Both these steps should be done before training
    - Training: stochastic gradient descent, can select optimization algorithm, tunes L1 and L2 regularization
    - ContinuousParameterRanges of logarithmic helps with learning rates with a range that spans several orders of magnitude. For example, if you specify a range of values between .0001 and 1.0 for the learning_rate, searching uniformly on a logarithmic scale gives you a better sample of the entire range than searching on a linear scale, because this would, on average, devote 90 percent of your training budget to only the values between .1 and 1.0, leaving only 10 percent of your training budget for the values between .0001 and .1
        - Logarithmic ranges tend to find optimal values more quickly than linear ranges
- **XGBoost**
    - Input can be recordIO-protobuf, parquet, csv
    - Models are de/serialized with pickle
    - CSVs must not have a column header record, and target variable must be the first column

  - Can be used as framework within notebooks, or as a built-in sagemaker algorithm
  - Difficult to tune all of its hyperparameters
    - To prevent overfitting: use **subsample** and Eta, max_depth not too big
    - To help with imbalanced classes, use scale_pos_weight
  - Training can be parallelized across machines
  - Automatically handles correlated features, numeric features on a different scale, and numeric-categorical variables
  - Has a lower and upper bound it can predict for regression, which is determined based on the range of values seen in the training. Tree-based algorithms do the same. Linear regression and neural networks don't do this
- **K-nearest-neighbors**
  - Used for classification and regression
  - Input: recordIO-protobuf or csv
  - Train: CPU or GPU. Inference: CPU (lower latency) or GPU (higher throughput on large batches)
- **Factorization machines**
  - Supervised method, classification or regression to deal with sparse data
  - Used for building recommender systems and for collaborative filtering
    - leverages other user's experiences. users with similar tastes (based on observed user-item interactions) are likely to have similar interactions with items they haven't seen before. works better when there is a lot of data
  - Input: recordIO-protobuf with *float32*. csv isn't practical for sparse data
- **DeepAR**
  - Forecasting 1D time series data with RNNs. The same model is trained over several related time series. Finds frequencies and seasonality
  - Input is JSON format in Gzip or Parquet
  - Better results than autoregressive integrated moving average (ARIMA) and error, trend and seasonality (ETS)
  - Time series filling methods:
    - Middle filling: fills missing values between the item start and item end data of a data set
    - Back filling: fills missing values between the last recorded data point and the global end date of a dataset
    - Future filling: fills missing values between the global end date and the end of the forecast horizon
- **PCA**
  - Dimensionality reduction, unsupervised
  - regular: For datasets with sparse data and a moderate number of observations and features
  - randomized: For datasets with both a large number of observations and features
  - Input: recordIO-protobuf or csv
- **IP insights**
  - Unsupervised learning of IP address usage patterns, identify suspicious behavior
  - Can be used for fraud detection
  - Input: csv only
- **Reinforcement learning**
  - Uses deep learning framework with Tensorflow and MXNet
  - Can distribute training and/or environment rollout
- **Automatic model tuning**

  - Define hyperparams and ranges, and metrics you're optimizing for
  - Sagemaker creates a HyperParameter tuning job that trains as many combinations as you decide. The set of hyperparams producing the best results can be deployed as a model
  - Don't optimize too many hyperparams at once. Limit your ranges to as small a range as possible
  - Don't run too many training jobs concurrently. Ideally one
- **AutoML**
  - Load csv input data from S3 and select target column
  - Automatic preprocessing and model creation, model notebook is available
    - Problem types: binary/multiclass classificaiton, regression
    - Algorithm types: linear learner, XGBoost, deep learning
  - Model leaderboard with ranked list of recommended models, you can pick one
  - Deploy and monitor the model, refine via notebook if needed
  - Integrates with SG Clarify for explainability, uses Shap values, assigns each feature an importance value for a given prediction
  - Can have human guidance

3.3 Text algorithms

- **Seq2seq**
  - Used to generate text as output
  - Input: recordIO-protobuf with integer tokens. Text files must be tokenized
  - Input data must contain training and val data and vocabulary file
  - Pre-trained models and public datasets are available
  - Training can only be done in one machine
- **BlazingText**
  - Supervised method for text classification. Uses word2vec
  - Each line should start with `__label__mouse` (label immediately appended to prefix), and tokens within the sentence should be space separated
  - The order of the words doesn't matter because it uses skip-gram and continuous bag of words
- **Object2vec**
  - Similar to word2vec but for arbitrary objects, like sentences with labels
  - Creates low-dimensional embedding layer
  - Can be used to find similarity between different texts
  - Input: data must be tokenized into integers. Training data consists of pairs of tokens and/or sequences of tokens
  - Training: two input channels, two encoders (which network to choose is a hyperparam), and a comparator that decides the output label
  - Can only be trained on a single machine (CPU or GPU)
  - Use INFERENCE_PREFERRED_MODE envvar to optimize for encoder embeddings
- **Neural topic modeling**
  - Unsupervised method to organize documents into topics
  - Input: recordIO-protobuf or csv. words must be tokenized into integers
- **Latent Dirichet Allocation (LDA)**
  - Unsupervised
  - Input: recordIO-protobuf or csv. Pipe mode only supported with recordIO
  - Training: *single-instance CPU*
  - Observations: documents. Feature set: vocabulary. Feature: word. Output categories: topics

3.4 Images algorithms

- **Object detection**
  - Input: recordIO or image format (jpg/png). With image format, also a JSON file for annotation data for each image
  - Training optimized for GPU, can be trained on multi-machine
- **Image classification**
  - Input: Apache MXNet RecordIO (not protobuf!) or jpg/png images. Image format requires .lst files to associate image index, class label and path to image
  - GPU for training, multi-GPU and multi-machine supported
- **Semantic segmentation**
  - Input: jpg images and png annotations, also label maps to describe annotations. Augmented manifest image format is supported for Pipe mode
  - jpg images accepted for inference
  - Only single-machine GPU supported for training. Inference on CPU/GPU

3.5 Clustering algorithms

- **Random cut forest**
  - Algorithm to find anomalies. Used in Kinesis data analytics
  - Input: recordIO-protobuf or csv. Optional test channel for computing accuracy, precision, recall and F1
  - No GPU
- **K-means clustering**
  - Input: recordIO-protobuf or csv
  - CPU and GPU, but CPU recommended
  - To find the optimal value of k: use the "elbow method" on a plot of the total within-cluster sum of squares (WSS) as a function of k
  - Can be used as dimensionality reduction

3.6 Apache Spark & Sagemaker

- Spark is used to parallelize pre-processing of big data on SG: sagemaker-spark package to integrate both, or do it separately: first pre-process in EMR, store in s3, take it from there for SG
- Use sagemaker_pyspark and the model's SageMakerEstimator if you want to use Spark for pre-processing, but you still want it to run in SG
- Allows combining pre-processing big data in Spark with training and inference in SageMaker
- You can connect notebook to a remote EMR cluster running Spark, or use Zeppelin
- Training df should have: feature column that's a vector of doubles, and optional labels column of Doubles. Then call fit() on SGEstimator to get a SGModel. Call transform() on SGModel to make inferences. This works with Spark Pipelines as well
  - To run the .fit() locally, provide instance_type="local" when initiating the Estimator

3.7 Sagemaker tools

- SG Data Wrangler: import, transform, analyze, export data within SG Studio
  - Provides a Data Quality and Insights report that automatically verifies data quality (such as missing values, duplicate rows, and data types) and helps detect anomalies (such as outliers, class

imbalance, and data leakage) in your data

- SG Feature store: find, discover and share features in Studio
- SG JumpStart: one-click models and algorithms from model zoos
- SG Debugger
    - Stores gradients and tensors over time as model is trained
    - Defines rules for detecting unwanted conditions while training
        - Monitor system bottlenecks
        - Profile model framework operations
        - Debug model parameters
    - Collects logs and fires Cloudwatch event when rule is met
    - Automatically generates training report
    - Built-in actions to receive notifications (email, sms) or stop training if rule is met
    - SG Debugger Insights Dashboard: see all about the Debugger in visual form
- Model monitor
    - No-code system get alerts on quality deviations on deployed models
    - Monitoring types:
        - Model is overfitting
        - Data quality drift
        - Bias drift
        - Feature attribution drift (compares feature ranking of training vs. live data)
    - Data stored in S3 and metrics in CloudWatch
    - Integration with Tensorboard, QuickSight, Tableau, or just visualize with SGStudio
- SG Canvas
    - No-code tool to build ML models
    - Upload csv data from S3, select column to predict, build and make predictions. Can do classification and regression, it does automatic data cleaning (missing values, outliers, duplicates) and shares models and datasets with SG Studio

4. High-level AI services

4.1 Comprehend

- Extract key phrases, entities, sentiment, language, syntax detection, topics and document classificaction
- Can train on own data
- Supports custom entity recognition model to identify new entity types not supported as one of the preset generic entity types. Better solution than using regex or string matching

4.2 Translate

- Deep learning for translation. Automatic language detection
- Supports custom dictionary, in CSV or TMX formatting, with their translations. Appropriate for proper names, brand names, etc.

4.3 Transcribe

- Speech to text
    - Input in FLAC, MP3, MP4 or WAV in specified language
    - Streaming audio supported (HTTP/2 or WebSocket), but only for english, french and spanish

- Speaker identification, how many speakers
- Channel identification: two callers can be transcribed separatedly, merging based on timing of utterances
- Automatic language identification
- Support custom vocabularies: vocab lists (just list of special words)
- Supports vocabulary filtering, if a list of offensive words is provided
- Can do content redaction, but only for PII info, not for censoring offensive words

4.4 Polly

- Text to speech
- Supports lexicons: customize pronunciation of specific words and phrases
- SSML (speech synthesis markup language)
    - Alternative to plain text, gives control over emphasis, pronunciation, breathing, whispering, speech rate, pitch, pauses
    - "Emphasis" tag exists, "pronunciation" tag doesn't
    - If something is pronounced wrong, use pronunciaton lexicons
- Speech marks: can encode when sentence/word starts and ends in the audio stream. Useful for lip-synching animation

4.5 Rekognition

- Object and scene detection, image moderation, facial analysis, face comparison, celebrity recognition
- Text in image
- Video analysis
    - Objects/people/celebrities marked on timeline, people pathing
- Images come from S3, or provide image bytes as part of request
- Video must come from Kinesis Video Streams: H.264 encoded, 5-30FPS
- Supports custom labels, you can provide a small set of labeled images. You can start with only 10 images
- Typical workflow
    1. Video uploaded and stored in data lake
    2. Lambda triggered, create metadata for celebrities, emotions, using rekognition video and transcribe
    3. Output is sent to Opensearch

4.6 Forecast

- Time series analysis
- AutoML chooses best model for time series
- Works with any time series. It can combine with associated data to find relationships

4.7 Lex

- Chatbot engine
    - Handles speech-to-text and chatbot logic. No comprehend needed
- Utterances invoke intents ("I want to oder a pizza")
    - For more complex text, if you need text processing, Lex integrates with Comprehend

- Lambda functions are invoked to fulfill the intent
- *Slots* specify extra information needed by the intent (pizza size, toppings etc.). Has to be coded
- Can be deployed to AWS Mobile SDK, Facebook Messenger, Slack, Twilio
- Alexa uses Transcribe and Polly

Amazon Lex Automated Chatbot Designer

- You provide existing conversation transcripts
- Lex applies NLP and deep learning to remove overlaps and ambiguity
- Intents, user requests, phrases, values for slots are extracted
- Ensures intents are well defined and separated

4.8 Personalize

- Recommendation engine
- For big batch operation: add data to S3, give schema to Personalize and then it will monitor S3 for that data
- For real-time, send data through API
- Explicit schema in Avro format must be provided
- **GetRecommendations**
  - Returns recommended products, content, etc.
  - Returns similar items
- **GetPersonalizedRanking**
  - Rank a list of items provided
  - Allows editorial control/curation
- It can create recommendations for new users and new items that it hasn't seen before (the cold start problem). Just recommends popular items to new users. And for new items, they don't stay new for long, as soon as someone buys it, it starts building relationships to other items and Personalize recomputes this every two hours
- Intelligent user segmentation, automatically classify users into groups for marketing campaigns
- To maintain relevance, keep the dataset current: incremental data import. To do that real-time, use PutEvents API call to feed in real-time user behavior. Retrain the model, by default it updates every 2h. It should also do a full retrain (trainingMode=FULL) weekly

4.9 Other AI services

- Textract: OCR with forms, fields, tables support

  - Use Augmented AI (A2I) to get low-confidence results from Textract's AnalyzeDocument API operation reviewed by humans

- DeepRacer: research tool, RL-powered 1/18-scale race car

- Fraud detector: upload historical fraud data. It builds custom model from a template you choose. It accesses the risk based on if the account is new, guest checkout, "try before you buy" abuse, or online payments

- Contact Lens for Connect: contact lenses made for customer support call centers. Ingests audio from recorded phone calls. Allows search on calls and chats

- Kendra: Enterprise search for intranet with NLP. Combines data from filesystems, SharePoint, intranet, into one searchable repository
- Augmented AI (A2I): human review of ML *predictions*. Builds workflows for reviewing low-confidence predictions by leveraging the Mechanical Turk workforce. MTurk uses humans so it is slow
- DeepLens: research tool, DL-enabled video camera. Not suitable for commercial uses or surveillance camera. not designed to be mounted on walls, ceilings, posts
- Panorama: CV at the edge, at IP cameras. low latency than sending predictions to the cloud
- TorchServe: Model serving framework for PyTorch
- Neuron: SDK for ML inference specifically on AWS Inferentia chips (Inf1 instance type). Integrated with SageMaker or other DL AMIs
- CodeGuru: automated code reviews. Finds lines of code that hurt performance, resource leaks, race conditions. Offers recommendations. Supports Java and Python
- Lookout: anomaly detection from sensor data to detect equipment/metrics/vision issues
- Monitron: industrial equiment monitoring and predictive maintenance. Provides sensors, gateways, service and app

4.10 SageMaker on the Edge

- **Neo**
  - Machine Learning for edge devices: ARM, Intel, Nvidia, can be embedded in anything
  - Optimizes code for specific devices: tensorflow, MXNet, PyTorch, ONNX, XGBoost
  - Consists of a compiler and a runtime
- **Greengrass**
  - Neo-compiled models can be deployed to an HTTPS endpoint
    - Hosted on C5, M5, M4, P3 or P2 instances
    - Must be the same instance type used for compilation
  - Or you can deploy model to IoT Greengrass, without Neo
    - Inference at the edge with local data, using model trained in the cloud
    - Uses Lambda inference applications

5. Evaluation

- Confusion matrix
- rMSE
- Residual plot: used for regression, shows how far away (positive or negative) the predicted value is compared to the expected value. Whether the model is underestimating or overestimating on the target
- Formulas
  - f1-score: 2 * precision * recall / (precision + recall). gives same importance to both classes
  - precision: tp / (tp + fp)
  - recall: tp / (tp + fn)
    - Important when the cost of a fn is higher than that of a fp

        - For multiclass, total recall = sum(recall for each class)/#classes
    - false negative rate: fn / (fn + tp)
    - true negative rate / specificity: tn / (tn + fp). If high, means that there are few fp
- Number of positive samples: tp + fn
- Number of negative samples: tn + fp
- ROC / AUC: AUC is area under the ROC curve. Good when we want to check the positive class performance (good for imbalanced datasets)
    - if ROC is a straight line, then AUC=0.5, model is just random. a good result should have a ROC curve like a logarithmic curve towards 1
    - ROC compares the true positive rate and the false positive rate at different thresholds
- Precision - AUC: for imbalanced datasets, where we care more about the positive class
- Correlation: negative correlation coefficient means the bigger the x, the lower the y

6. Machine Learning operations

Steps to deploy:

- Create model. Include:
    - S3 path where the model artifacts are stored
    - Docker registry path for the image that contains the inference code
- Create endpoint configuration
- Create endpoint

Deployment types:

- Canary: releases the new version to a small set of users, then gradually increases traffic to more and more users. So, traffic at the beginning is 0:1, then begins to update periodically. This mitigates the risk of changes to the production. you can choose which users to direct to the new version
- Rolling: similar to canary, but starts deploying to servers and not to users.
- A/B: to test something new, it might start with 80-20, 60-40. that means automatically 20 or 40% of users get new version. in canary it goes gradually
- Blue/Green: create two separate, but identical environments -> two endpoints. One environment (blue) is running the current application version and one environment (green) is running the new application version. Increases application availability and reduces deployment risk by simplifying the rollback process if a deployment fails. Once testing has been completed on the green environment, live application traffic is directed to the green environment and the blue environment is deprecated

6.1 Docker

Structure of a training container

```
/opt/ml
---input
------config
---------hyperparameters.json
---------resourceConfig.json
------data
---------<channel_name>
------------<input data>
```

```
---model
------<model files>
---code
------<script files>
---output
------failure
```

Structure of training container:

- nginx.conf
- predictor.py (Flask web server to make predictions at runtime)
- serve/ (launches gunicorn server, multiple flask web servers defined in predictor)
- train/ (invoked when running the training)
- wsgi.py (wrapper to invoke flask app for serving results)

```
FROM tensorflow/tensorflow:2.0.0a0
RUN pip install sagemaker-containers
# copies training code inside container
COPY train.py /opt/ml/code/train.py
# defines train.py as script entrypoint
ENV SAGEMAKER_PROGRAM train.py
```

All files in `/opt/ml/` and `/tmp/` can be encrypted with a KMS key

Custom inference containers have following restrictions:

- Accept all socket connections within 250ms
- Respond to both /invocations and /ping on port 8080
- Respond to ping within 2s
- Model artifacts should be in **tar** format

If you plan to use GPU, make sure container is nvidia-docker compatible. Only CUDA toolkit should be included in the container, not the NVIDIA drivers

6.2 Inference

- Sagemaker endpoints are by default not open to the public and needs AWS credentials to access it. For open-to-the-public-endpoints, use API Gateway
- Multi-model endpoint works for several models. They use a shared serving container which hosts several models. Improves endpoint utilization compared with using single-model endpoints, and reduces the deployment overhead. To increase availability, add SG instances of the same size and use the existing endpoint to host them
- You can test out many models on live traffic using **production variants**, to distribute traffic among different models
- Automatic scaling: set a scaling policy to define target metrics, min/max capacity, cooldown periods
- To have high availability with SG endpoints, deploy multiple instances for each production endpoint and configure VPCs with at least 2 subnets, each in a different AZ

6.2.1 Elastic inference

- To reduce inference latency, cheaper than using a GPU instance
- Only works for Tensorflow, PyTorch and MXNet pre-built containers. ONNX can be used to export models to MXNet
- Only works with custom containers built with EI-enabled Tensorflow, PyTorch or MXNet
- Only works with image classification and object detection built-in algorithms

6.2.2 Serverless inference

- Serverless endpoints. Good option for infrequent or unpredictable traffic: it will scale down to zero when there are no requests
- Specify container, memory requirement, concurrency requirement

6.2.3 Inference recommender

- Recommends best instance type and config for your model and deploys to optimal inference endpoint
- Automates load testing, model tuning
- How it works:
  - Register model in model registry
  - Benchmark different endpoint configs
  - Collect and visualize metrics to decide on instance types
  - Existing models from zoos may have benchmarks already
- Instance recommendation: runs load test on recommended instance types, takes 45mins
- Endpoint recommendation: cuustom load test, you specify instances, traffic patterns, latency requirements, throughput requirements. Takes 2h

6.2.4 Inference pipelines

- Linear sequence of 2-15 containers
- Any combination of pre-trained built-in algorithms or your own algorithms in containers
- Combines pre-processing, predictions, post-processing
- SparkML and scikit-learn containers work well
  - SparkML can be run with Glue or EMR, it will be serialized into MLeap format
- Can handle both real-time inference and batch transforms
  - Batch transform uesful for inference on the whole dataset. It can exclude attributes before running predictions. You can also join the prediction results with partial or entire input data attributes when using data that is in CSV, text, or JSON format

7. Security

- SG supports authorization based on resource tags and identity-based policies, but not resource-based policies
- In transit, *inter-node training communication* can be encrypted, via console or API when setting up a training/tuning job, which requires the creation of a VPC. Increases training time eand cost. Complies with regulatory requirements
- Notebooks, training and inference containers are Internet-enabled by default, which can be a security issue. If you disable this, the VPC needs an interface endpoint (PrivateLink) or NAT Gateway and allow

outbound connections. This prevents S3 access
- Notebooks with default IAM role have automatic access to S3 buckets whose name contains "sagemaker"
- To connect to AWS endpoint from a VPC, use VPC **interface endpoint** using PrivateLink for private connectivity

---

Cloudtrail

- *does not monitor* calls to SG InvokeEndpoint
- Provides a record of actions taken by an user/role/service in SG
- Records are kept for 90 days
- CloudWatch keeps the SG monitoring statistics for 15 months. But console limits the search to metrics that were updated in the last 2 weeks

---

To provide access to SG resources:

- Provide access to externally authenticated users through identity federation
- Create a role to delegate access to your resources with the 3rd party AWS account