|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Engineering | | | | | |
| S3 | Object have keys = full path. Max obj size = 5TB  Bucket names must be globally unique (convention to use lowercase & numbers) | | | | |
| S3 Standard (ms access) -> S3 Intelligent Tiering (unpredictable access pattern, monitoring fee) -> S3 Standard-IA (retrieval fee, min storage duration, min obj size) -> S3 One Zone-IA -> S3 Glacier (min to hour access) -> S3 Glacier Deep Archive (hours access, min 180 days) | | | | |
| Data in Transit: Secure Sockets Layer (SSL) or Transport Layer Security (TLS)  Server Side Encryption: - SSE-S3 uses AES-256. - SSE-KMS. - SSE-C (customer managed)  Client Side Encryption (data encrypted before upload)  Access Controls and Audit Trails: IAM, bucket policies, CloudTrail | | | | |
| EFS | | | Fully Managed, Stored across multiple AZ, Integrates w IAM  S3 better for: large amount of data, cost efficiency for ML  EFS better for: Hierarchical storage, simultaneous read/write (IOPS) for ML | | |
| EBS | | | For a single EC2 instance. EBS resides within a single AZ. High IOPS  EBS better for: pre-processing on a local disk, low latency requirements, high IOPS,  EFS better for: shared file system | | |
| Batch processing | | | | Data not required immediately, large transformations, lower cost  Challenges: data format, encoding, windows & missed runs, | |
| Stream processing | | | | Data needed immediately, less complex transformation, higher cost  Challenges: time windows, missed data, transformation challenges | |
| Amazon Kinesis | | | Kinesis Data Shard: sequence of data records in a stream  - 5 transactions per sec for reads and 1000 records per sec for writes  - Fixed unit of capacity that aids in organization. – Consist of seq num, partition key, data blob  - Hash functions defines in which partition | | |
| KDS: mostly managed service  Data storage 1 to 7 days  Real time processing  Multiple consumers/destinations  Support for Spark | | KDF: full managed service  No data storage  Near real-time processing  Management by Firehose  No support for Spark |
| 1. Collection and Ingestion: KDS, Temporary Data Store. 2. Preprocessing: KDF, SQL or Apache Flink  3. ML Model Training: S3, Batch Process, SageMaker. 4. Real-time Inference: Kinesis Data Analytics, Flink  5. Monitoring: Data Drift, Kinesis Data Analytics | | |
| AWS Glue Crawler | | | Crawler scans data stores and populate a data catalog (info about data) through Jobs  Data source: S3, JDBC DBs, DynamoDB, Redshift, Kafka, Mongo DB | | |
| ETL | | | Data extraction: Full, Incremental  Transform: Cleaning, validation, transformation, enrichment, anonymization, structuring, reformatting  Load: Full, Incremental, Upsert  Use for: privacy or security concerns (less movement and risk), compliance, small datasets  E (Glue, Kinesis, Glue crawler). T (EMR, Redshift). L (S3, DynamoDB, RDS) | | |
| ELT | | | Load data into data lake (also called as staging). For data lake can use Lake Formation  Use for: data lakes, large data, multiple uses cases, speed, cost | | |
| EMR | | | Simplifies big data frameworks: spark, Hadoop. Use for large scale data processing  Is a Managed Hadoop Framework. Use EC2/EKS for processing. Used for transformation, analysis, viz  Can leverage streaming data. | | |
| Spark | | | Core components: Spark Core (job execution engine),  Driver (read code and assign tasks to worker nodes) -> Cluster Manager (to acquire resources on Spark cluster and allocate them to diff Spark apps) -> Worker nodes (consist of Executors running tasks)  RDD (resilient distributed datasets): immutable collections of data items  - Fault tolerant. – Lazy transformations, Actions  EMR (main way to work w Spark). AWS Glue (hidden but is what powers Glue) | | |
| Flink | | | Real time analytics. Architecture: Client -> Job Manager (similar to driver) -> Task Managers  Event Time vs Processing Time. Watermarks (determine what to do w late data)  More customizability compared to Flink but harder to set up | | |
| Hadoop and Hive | | Hadoop: distributed processing of large data. HDFS split & distribute data. MapReduce (process, aggregate)  Hive: data warehousing solution built on Hadoop. Hive Client -> Hive server -> MapReduce -> HDFS  - HiveQL (HQL; SQL like language). – Tables and databases. – Partitioning and bucketing  EMR uses Hadoop/Hive. Glue also can | | | |

Use Athena partition pruning to reduce amount of data scanned by only querying relevant partitions

- If many small files in 1 directory, to read all as 1 object in Glue: use recursive partitioning (recursively traverse directory)

RecordIO protobuf format: binary data format optimized for training on SageMaker (faster data loading & lower memory)

Firehose can only convert data from JSON to parquet or ORC

Implement ETL pipeline using Glue to process streaming data records in real-time:

- Create Kinesis Data Stream and add Glue Data Catalog table w data stream as source. Write Glue PySpark job w new table as source and configure to perform transformations and write to tables in S3

To use EMR w Spark to transform data in S3 w Glue Data Catalog:

- Configure new S3 crawler and ETL job within Glue. Within the Scala MLlib script, access the data using spark.setCurrentDatabase and spark.sql to run SQL query

To use Glue to transform data in S3 to ingest into Redshift:

- Glue Relationalize transformation to transform DynamicFrame into a relational data format -> Ingest to Redshift

Kinesis Video Streams: collect, process, store, analyze time-encoded data like video and audio streams

Services that can feed data to EMR MapReduce Jobs: Kinesis, AWS Data Pipeline

EMR: only task nodes can be on Spot Instances. Master and core nodes should not be

Kinesis data shard limit: 1000 records per sec for writes up to max 1 MB per sec

- 5 reads per sec up to max 2 MB per sec

AWS Step Functions integrates w EMR for cluster creation, modification and termination

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Exploratory Data Analysis | | | | | | | |
| EDA Framework | | Univariate Analysis: frequency, mode, measures of central tendency (mean, median), measures of variability (range, standard deviation, variance, IQR)  Multivariate Analysis: scatter plot, correlation matrix  Data prep: - preparing instances/rows (duplicates, outliers, missing data, imbalanced data)  - preparing features (feature selection, encoding techniques, normalize/scaling, binning & transforming) | | | | | |
| Missing Data | | Missing Completely at Random (MCAR): missing data unrelated to both observed & unobserved data  - Mean, Median or KNN imputation  Missing at Random (MAR): Missing data dependent on observed data  - MICE (Multivariate Imputation by Chained Equations): runs multiple imputation threads parallelly, then combines result to product final dataset  -- Maintains the r/s btw the variables in the original data & reduces the bias  -- Effective for datasets w large amounts of missing data  Missing Not at Random (MNAR): Missing data related to both observed & unobserved data  - Selection models and shared parameter models | | | | | |
| Preparing text data | | | | | Remove stop words, tokenize. Libraries: NLTK, SpaCy | | |
| Resampling Imbalanced dataset | | | | | | Undersampling, oversampling, SMOTE (synthetic minority oversampling technique) | |
| Outliers | | Detect: 1) Z-score = z = x – m/sd. Outliers if lie outside 3 s.d. (generally)  2) Boxplot: min = quarter 1 – 1.5 \* IQR, max = Q1 + 1.5 \* IQR. Outlier if outside min and max  Fix: 1) Delete outliers. 2) Log transformation. 3) Replace outlier w median value | | | | | |
| Amazon SageMaker Ground Truth | | S3 -> Labeling Job in SageMaker Ground Truth (Workforce + Labeling UI) -> S3  Workforce: 1) Amazon Mechanical Turk = vendor provided workforce. 2) Private workforce  3) AI apps: Rekognition, Comprehend, Textract, Transcribe  Amazon SageMaker Ground Truth Plus: Managed service, Amazon dedicated workforce or 3rd party vendor w expertly trained personnel. – Customer can monitor and review data as it is labelled  Mechanical Turk also can used for image/video processing, info gathering, data verification | | | | | |
| Feature types | | Feature selection: choose subset of most relevant features  Feature extraction: create new features from existing ones, to reduce dimensionality of data  Feature transformation: transform features into a more suitable representation for ML model  Features: 1) Qualitative: - Nominal (no order). – Ordinal (ordered). – Boolean (binary values)  2) Quantitative: - Discrete (countable). – Continuous (infinite) | | | | | |
| Normalize, Standardize, Transform | | Feature transformation:  mitigate effect of skewness in data distribution  modify features to create new features  polynomial transformation, log, exponential, box-cox transformation  Help identify non-linear r/s | | | | | Feature scaling:  change feature’s scale in dataset  does not create new features  MinMaxScaler, StandardScaler, RobustScaler, MaxAbsScaler |
| StandardScaler = z-score normalization. Assumes data is normally distributed  MinMaxScaler = rescales data btw 0 and 1 using max and min  MaxAbsScaler = take absolute value of feature & divide by max value; [-1, 1]  RobustScaler = (X – Q1)/(Q3 – Q1). Uses IQR, and is robust to outliers  Normalization: if data does not have a Gaussian distribution | | | | | |
| Data Binning | | Aka data discretization or data bucketing  - Removes noise & address skewness. – Possible info loss, need determine bin size  1) Equal-width strategy: each bin have same range. 2) Equal-frequency strategy/quantile strategy  3) K-means binning: use K-means clustering algo to partition continuous data  - used when data dist not uniform  4) Decision tree binning: builds decision tree on continuous variable, and algo will determine split points  - effective when data has non-linear r/s | | | | | |
| Encoding categorical data | | Ordinal Encoding: convert category to unique number according to order.  Nominal Encoding: create new column for each unique category. One-hot encoding, Label encoding  Binary Encoding: nominal feature passed through label encoder & converted to numerical data  - Numerical data then converted to binary digits. – Digits split into individual columns  - Usually result in less columns than if to one hot encode directly  Helmert Encoding: encode dependent variables | | | | | |
| Extract Features from Text | | Bag of Words (BOW): get frequency of each word, then basically 1 hot encode  N-gram: produce groups of words of n size. n-gram w n = 1 is just BOW  TF-IDF: importance of word in document = TF \* IDF = count/total in doc \* log(total num of docs in corpus / no of docs containing the term + 1). - Higher score = more important  Word embedding: captures semantic and syntactic and represented as a vector  - CBOW (continuous BOW): takes a fixed sized windows of words around the target word. Predicts target word based on context words. - Skip-gram: predicts context word based on target word  Stemming: convert word to base form | | | | | |
| Extract Features from Images & Speech | | Image: 1) Traditional Computer Vision: Pixel Intensity values, Edge detection techniques  2) DL techniques: CNN, Transfer learning, Auto encoders | | | | | |
| Grayscale: Each pixel values = brightness. 0 to 255  Color: mean pixel value of RGB channels. Average across the 3 channels for each pixel  Edge features: subtract values on either side of the pixel. Higher num = pixel more likely to be edge  - Use Prewitt Kernel | | | | | |
| Speech: 1) Traditional Speech processing: Mel Frequency Cepstral Coefficient (MFCC), Linear Predictive Coding (LPC). 2) DL techniques: LSTM, Gated Recurrent Unit (GRU) | | | | | |
| Feature selection & Dimensionality Reduction | | | | 1) Feature selection: a) Filter: Variance thresholding, Chi-square Test  b) Wrapper (use performance of ML model to determine relevance of feature subsets): Forward selection, backward elimination, recursive feature elimination  c) Embedded (select features as part of model training process): Lasso Regression, Ridge Regression, Gradient Boosting Machines, Elastic Net  2) Feature extraction (to lower dimension): a) Linear dimensionality reduction (uses linear transformation): PCA, Linear Discriminant Analysis  b) Nonlinear dimensionality reduction: t-distributed Stochastic Neighbour Embedding (t-SNE), Isometric Mapping | | | |
| SageMaker Processing | | Pay-as-you-go pricing, Tight integration w other SageMaker components, supports multiple programming languages, libraries, frameworks, Version control for scripts  Processing Options: 1) SKLearnProcessor. 2) PySparkProcessor  - Define ProcessingInput, ProcessingOutput | | | | | |
| Probability Distributn | Find patterns in dataset, Express uncertainty, Determine goodness-of-fit, Perform Monte Carlo simulation  1) Discrete dist: Bernoulli, Binomial, Poisson. 2) Continuous dist: Normal, Log-Normal, Exponential | | | | | | |
| Visualizing r/s & Dist | | 1) Scatter plot: show r/s btw 2 numerical features. - Can detect non-linear patterns & outliers.  - Showing more than 2 features makes it less readable  2) Bubble chart: scatter plot + size of bubble represent 3rd dimension.  – Diff to distinguish overlapping data points. – Lacks interactivity like zooming, filtering, drill-down  3) Histogram: group data into bins/buckets & calculate freq. - See freq dist quickly.  – Can use for categorical data. – Num of bins determine granularity. – Not suited for detecting outliers  4) Boxplot: shows min, first quartile, median, 3rd quartile, max. Box = IQR  - Present summary of large amt of data. – Can detect outliers. – Exact data point value not known. – Not good for representing skewness present in data  5) Heatmap: represent numerical data distribution w colour intensity. - Good to identify patterns quickly. – Easy to spot high & low values. – Effective in communicating large amt of data in a compact fashion  - Challenging to interpret data, if unfamiliar w colour legends. – Not good for sparse data | | | | | |
| Visualizing comparison & composition | | | 1) Bar chart: bars to represent freq / value of diff categories. For categorical data unlike histogram  - Easy to compare values of diff categories visually. – Cannot represent continuous data  - As num of categories incr, chart can become cluttered  2) Line chart: how data changes over time. – Simple & easy to interpret.  – Can compare trends in multiple categories at the same time. – Cannot visualize categorical data.  – Not effective for sparse datasets w missing values. – Outlier can distort chart  3) Pie chart: express a part-to-whole r/s in data. – Easy to understand.  - As num of categories incr, chart can become cluttered. – Not suited for time series data  4) Stacked bar chart: shows magnitude of value + categories that make up the total value  - Shows contribution of each subcategory that make up the category  - Height of each bar shows importance of a category. – More categories = more cluttered  - Not effective to represent data w negative values | | | | |
| Descriptive Statistics | | 1) Measure of frequency: frequency, mode. 2) Measures of central tendency: mean, median  3) Measure of variability: range, standard deviation, variance | | | | | |
| Leptokurtic Distributions: Definition, Example, Vs. Platykurtic1) Skewness = measure of asymmetrical nature of data distribution  Negative skewness = left skewed = mean < mode  Positive skewness = right skewed = mean > mode  Pearson first coefficient of skewness = (mean - mode) / standard deviation  Pearson second coefficient of skewness = 3(mean - mode) / standard deviation  2) Kurtosis = measure of outliers existing in the data  - Large/huge/heavy/fat tail = more outliers. – Small/thin tail = less outliers  - Mesokurtic distribution = normal distribution  - Leptokurtic = heavy tail distribution = higher kurtosis than mesokurtic  - Platykurtic = short tail distribution = smaller kurtosis than mesokurtic  3) Correlation = define r/s btw 2 features. [-1, 1] | | | | | |
| Cluster analysis using Elbow method | | 1) Partition clustering: user specify num of clusters. – Each cluster contains ≥ 1 observation and are unique. – K-means algo  - Use Elbow method to determine optimal number of clusters  - Start w k = 1. Compute Within-cluster sum of squares (WCSS). Repeat by increasing k. Find elbow point  2) Hierarchical clustering: user don’t specify num of clusters. – Cluster assignments determined by creating a hierarchy and cutting tree at a specified depth. - Clusters are merged to form a dendrogram.  – Computationally expensive  – Agglomerative clustering algo: bottom up approach. Each data point considered a cluster. Clusters are merged recursively based on dist btw the 2 points  - To calculate dist btw points: use linkage methods  a) Single linkage: shortest dist btw any 2 points in the 2 clusters  b) Complete linkage: max dist btw any 2 points in the 2 clusters  - Opposite of agglomerative hierarchical clustering: Top down approach. Consider all data points as a single cluster. Computes measures of dissimilarity btw data points. Highest dissimilarity is split  3) Density-based clustering: user don’t specify num of clusters. – Cluster assignments based on density of data points. – DBScan algo | | | | | |
| Amazon QuickSight | | BI self-service solution to visualize and analyse data quickly, irrespective of their technical expertise  - Integration w multiple data source. – No infra management. – Auto data refresh. – Support diff data formats. – Collaboration & sharing. – Security & compliance  - Fast performance w SPICE (Super-fast, Parallel, In-memory Calculation Engine). – Mobile app support.  - Data insights powered by ML  Data source -> Dataset (subset of data) -> Analysis (basic workspace for creating viz) -> Sheet (pages that display viz) -> Visuals (single chart/graph). Dashboard = read-only snapshots of Analysis | | | | | |
| QuickSight ML Insights | | 1) Forecasting: predict business metrics, perform interactive analysis, discover hidden insights  - Uses Random Cut Forest algo to detect trends & fill missing values  2) Anomaly detection. – Use RCF algo as well  3) Autonarratives: share story behind the data | | | | | |
| Data Requirements: - Must have 1 metric dimension. – Must have 1 categorical dimension  - Must have 1 date dimension for anomaly detection & forecasting. - At least 15 data points | | | | | |

To identify columns that can be grouped tgt and visualize results quickly: embed numerical features using t-SNE algo & create scatter plot to show cluster of points

For missing data imputation, DL for categorical data, KNN for numerical data?

Poisson distribution: num of events occurring within a fixed time or space. Discrete distribution

Exponential dist: time btw events in a Poisson process. Continuous dist

Univariate selection: assess ea feature individually w target variable.

- E.g. chi square/ANOVA for categorical, F-test/correlation coefficient for numerical

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modeling | | | | | | | | | | | | | | | | |
| ML | | Supervised: 1) Regression: Linear/Multiple/Polynomial regression (single/multiple/polynomial features)  2) Classification: Binary/Multiclass classification . 3) Time series | | | | | | | | | | | | | | |
| Unsupervised learning: 1) Clustering. 2) Dimensionality reduction. 3) Anomaly detection | | | | | | | | | | | | | | |
| Reinforcement learning | | | | | | | | | | | | | | |
| Supervised Classification | | | | | | | | Binary: Logistic regression (sigmoid curve, computationally efficient but sensitive to outliers), SVM (draws a hyperplane to separate classes, nearest points = support vectors, generalizes well to unseen data but computationally expensive & memory intensive).  Multiclass: KNN (no training but memory intensive for prediction), Naive Bayes (Bayes theorem, assume each feature is independent, can handle missing data w/o imputation)  Multilabel (can be multiple at once): Ensemble methods, Deep learning  Imbalanced data: use SMOTE | | | | | | | | |
| Eager learners: spend more time in training, less time in predictions. (Logistic Regression, SVM)  Lazy learners: don’t create model in training, more time in predictions (KNN) | | | | | | | | |
| Accuracy, precision, recall | | | | | | | | |
| Supervised Regression | | | | | | | | | | | | | | MSE, RMSE, R-squared | | |
| Supervised Time Series | | 1) Trend: directionality of data over time  2) Seasonality: periodic fluctuations that occur at regular intervals within the data  3) Cyclical variations: non-repeating fluctuations that may occur at irregular intervals  4) Irregularity: randomness/noise in data | | | | | | | | | | | | | | |
| 1) Stationary Data: data changes predictably. More reliable forecasts  - Mean, variance constant over time. – Covariance btw observations at diff time lags also constant  2) Non-stationary data: - mean exhibits upward/downward trend  - variance exhibits periodic fluctuations at fixed intervals  - Use detrending, differencing, transformation techniques to convert to stationary data | | | | | | | | | | | | | | |
| Cons: cannot handle missing values, assumes linear r/s btw features, dependency on historical data | | | | | | | | | | | | | | |
| Clustering | | Hard clustering: each data point belongs to 1 cluster only (K-means)  Soft clustering: each data point assigned a prob of belonging to a cluster (Fuzzy C-means) | | | | | | | | | | | | | | |
| 1) Centroid-based clustering: num of clusters predetermined, calculate centroid (K-means)  2) Density-based clustering: num of clusters not predetermined, randomly select data point and expand its neighbourhood within a specified radius, cluster formed if num of data points > threshold  - Good at handling outliers (DBScan algo)  3) Hierarchical clustering: group data based on similarities & build hierarchy  - Smaller clusters merged into larger clusters, forming dendrograms. – Bottom-up (ea data points is a cluster initially; agglomerative hierarchical clustering algo) & Top-down  4) Distribution-based clustering: assumes data points are generated from a mixture of probability distributions & seeks to model these distributions  - Estimates the mean, variance, and mixture weights using the expectation-maximization algo (EM algo)  - Once estimated, assign ea data point to cluster w highest prob of generating that data point  - Allow for soft clustering. – Gaussian Mixture Model | | | | | | | | | | | | | | |
| Unsupervised: Association | | | | | | | | | | | Data mining technique to detect implicit r/s and hidden patterns btw features  Terminology: If {condition} Then {action}. Condition = antecedent, Action = consequent  - If association rules involve a single item only, then r/s btw antecedent & consequent = single cardinal relation | | | | | |
| Metrics: 1) Support = freq of X and Y tgt in a transaction / total num of transactions  2) Confidence = freq of X and Y / Freq of X. 3) Lift = confidence of X and Y / Support of Y  Lift > / < 1 = antecedent & consequent more/less frequent tgt. Life = 1 = independent | | | | | |
| 1) Apriori algo: discovers frequently occurring items & create association rules (uses BFS algo)  2) FP-growth algo: constructs a compact data structure (FP tree) to represent the entire dataset & mines the frequently occurring items. – More efficient for large datasets  3) Eclat algo (equivalent class transformation): discovers frequently occurring items (uses DFS algo) | | | | | |
| Reinforcement Learning | | | | | | | | | | | Self-learning ML technique that enables an agent to learn in an interactive env by trial & error to achieve optimal results | | | | | |
| - Agent: entity that is learning to make decisions. - Environment: domain where agent operates  - State : specific condition of the env at a given time. - Action: decision made by agent on current state of env. – Reward: feedback provided by env to agent. – Policy: agent’s decision-making process | | | | | |
| 1) Model-based RL: - agent builds an internal representation of env. – acts on env and notes new state. – associate reward value w state transition. – repeat until model is built  2) Model-free RL: - agent doesn’t build an internal model of env. – use trial and error to learn env dynamics. – records action taken and state achieved. – sequences them to develop a policy | | | | | |
| Deep Learning | | Perceptron: Multiple weights to diff inputs + bias -> 1 node -> activation function -> output  Error is backpropagated and recompute the training process  Activation function: Step function, Sigmoid (prob), ReLU (eliminate negatives), Softmax (prob sum to 1, for multiclass classification output layer), Tanh ([-1, 1], since 0-centered, faster convergence for shallow NN)  ANN: Input + Hidden + Output layer. Error = cost function | | | | | | | | | | | | | | |
| CNN | | Input -> Convolution layer -> Pooling layer -> Flattening layer -> Fully connected layer (FC)  Convolution = multiplication operation btw 2 functions resulting in an output value  - Use filter/kernel to slide over image & perform convolution operation. - Output matrix = feature map  - Padding, Stride/step size.  Use activation layer to introduce non-linearity to. CNN  Pooling layer to down sample: 1) Max pooling. 2) Average pooling  Flattening layer: convert 2/3 dimensional array into a linear vector. Acts as bridge btw convolution & FC | | | | | | | | | | | | | | |
| RNN | | Process sequential data. Time-series, Text, Audio-visual data | | | | | | | | | | | | | | |
| FNN: assumes input & outputs independent  Info flows in 1 direction from input to output layer  Cannot maintain historical data | | | | | | | | | | | | | RNN: output depends on all sequence elements  Info flows in both directions, allowing it to maintain internal memory  Considers curr input & learnings from past input | |
| RNN creates 1 hidden layer and loop over it many times | | | | | | | | | | | | | | |
| 1) LSTM (long short-term memory). - As gap btw context gets wider, RNNs become less effective, LSTM can capture long-range dependencies over extended seq. – Contain memory cells  - Uses input, output, forget gate to control info flow  2) GRU (gradient recurrent unit): capture long-range dependencies. – Uses reset, update gates to direct info flow. – Maintains simpler architecture w fewer gating units that selectively update hidden state | | | | | | | | | | | | | | |
| RNN configs: 1) One-to-one: single input and single output. (Word prediction)  2) One-to-many: single input generate multiple output. (Image captioning)  3) Many-to-one: Multiple input generates single output. (Sentiment analysis)  4) Many-to-many: multiple input generates multiple output. (Language translation) | | | | | | | | | | | | | | |
| Transfer learning | | Use knowledge of a pre-trained model to improve performance on a new related task  - Freeze common starting layers, unfreeze weights of later, task-specific layers or introduce new layers | | | | | | | | | | | | | | |
| SageMaker | | Data collection: SageMaker Ground Truth -> Data analysis: SageMaker Data Wrangler  -> Data processing: SageMaker Feature Store -> Build model: SageMaker Notebook  -> Train/Test model: Built-in Algos -> Deploy model: SageMaker Hosting Services or SageMaker Batch Transform -> Monitor model: SageMaker Model Monitor | | | | | | | | | | | | | | |
| Training data: S3. Training code run in SageMaker compute instances  Training job/image stored in ECR. Training data & job must run in the same region | | | | | | | | | | | | | | |
| Implementation Options:  1) Built-in algos: no coding. Requires training data, hyperparams, compute resources  2) Script mode: Develop custom Python script but use pre-installed Python libraries  3) Custom docker image: requires docker expertise + image must be uploaded to ECR | | | | | | | | | | | | | | |
| SageMaker XGBoost | | Gradient boosted trees algo  Ensemble learning: multiple ML models combined to improve prediction accuracy  Boosting: ensemble learning technique that sequentially combines predictions to improve performance  - Initialise same weights to all models. – Ea weak learner trained against a subset of training data  - Error of weak learner computed. – Models w larger error rates assigned higher weights & retrained  - Repeat & final prediction based on weighted total of all weak learners  Boosting algo types: 1) Adaptive boosting (classification). 2) Gradient boosting (classification/regression) | | | | | | | | | | | | | | |
| Decision Tree (DT) algo: Can solve classification/regression  - Divide dataset into smaller subsets based on their features. – Predicts output by evaluating seq of true/false feature questions | | | | | | | | | | | | | | |
| SageMaker XGBoost: classification or regression. - File type: libsvm, CSV, parquet, protobuf.  – Instance type: CPU, GPU. – Hyperparams: num\_round, num\_class (both required, have other optional)  - Metrics: MAE, MSE, RMSE, MAP, Accuracy, AUC, F1 score | | | | | | | | | | | | | | |
| SageMaker Linear Learner algo | | For both regression & classification. Uses SGD to best fit line to the data points.  Iteratively adjusts model params to minimize loss function. - File: CSV, protobuf, JSON (inference only).  – Instance: CPU, GPU. – Required hyperparams: num\_class, predictor\_type.  – Metrics: Cross entropy loss, absolute error, MSE, Precision, recall, accuracy | | | | | | | | | | | | | | |
| SageMaker KNN | | Non parametric: no assumption about underlying data. For regression/classification  Training process: 1) Sampling: specify sample dataset size.  2) Dimensionality reduction: type of dimension reduction is specified as hyperparam.  3) Indexing: for efficient lookups btw data points, indexing type specified as hyperparam  - File: CSV, protobuf, JSON (inference only). – Instance: CPU, GPU.  – Required hyperparams: feature\_dim (num of features in input dataset), K, predictor\_type (inference type), sample\_size, dimension\_reduction\_target (target dimension to reduce to)  – Metrics: MSE, accuracy | | | | | | | | | | | | | | |
| SageMaker Factorization Machines algo | | | | | | | | Supervised learning algo & extension of a linear model designed to capture higher-order r/s in dataset  Higher order: interactions btw ≥ 3 features vs Non-linear: r/s btw features cannot be expressed by a linear fn (since involve ≥ 2 features, r/s is not necessarily higher-order)  , where , are latent vector representation of a feature  Cons: - only consider pairwise features, - don’t support CSV, - don’t support multi-class classification  - requires lots of data, - recommended CPU only, - don’t perform well on dense data | | | | | | | | |
| - Binary classification/regression. - File: protobuf, JSON (inference only). – Instance: CPU.  – Required hyperparams: feature\_dim (num of features in input dataset), num\_factors (dimensionality of factorization), predictor\_type (inference type). - Metrics: RMSE, accuracy & cross-entropy | | | | | | | | |
| For recommendation system or Ad-click prediction | | | | | | | | |
| SageMaker DeepAR Forecasting algo | | | | | Supervised learning algo for forecasting 1-dimensional time-series data using RNN  1-dimensional data: sequential data where each observation corresponds to a single variable measured at regular intervals  Cold start problem: not enough history/data to train model | | | | | | | | | | | |
| Types of Forecast: 1) Point-in-time forecast: single predicted value for ea time step in forecast period  2) Probabilistic forecast: provide range of possible values for ea time step + associated probabilities | | | | | | | | | | | |
| Input file format required features: 1) start = string representation of a timestamp w/o time zone  2) target: array of floating-point values/int representing the time-series data  3) dynamic\_feat (optional): array of arrays of floats/ints representing vector of custom feature time series data. 4) cat (optional): array of categorical features that can be used to encode the groups that the record belongs to | | | | | | | | | | | |
| - Forecasting (time-series). – File: gzip, parquet. – Instance: CPU, GPU  - Required hyperparams: context length (num of time points model can see before making prediction), epochs, prediction\_length (num of time steps model is trained to predict), time\_freq (granularity of time series in dataset). – Metrics: RMSE, mean\_wQuantileloss, final\_loss | | | | | | | | | | | |
| SageMaker PCA | | Unsupervised learning algo to reduce num of features in dataset w/o losing meaningful info  - Combine uncorrelated original features into components. – 1st component captures max variance in data. – 2nd component captures remaining max variance orthogonal to 1st component  - Eigenvalues & eigenvectors used to determine magnitude and direction of these components | | | | | | | | | | | | | | |
| Modes of operation: 1) Regular: for datasets w sparse data & moderate num of features  2) Randomized: for datasets w large num of observations & features | | | | | | | | | | | | | | |
| - Dimensionality reduction (unsupervised). – File: CSV, protobuf, JSON (inference). – Instance: CPU, GPU  - Hyperparams: feature\_dim, mini\_batch\_size, num\_component | | | | | | | | | | | | | | |
| SageMaker Random Cut Forest (RCF) | | Unsupervised learning algo to detect anomalies  - Computes dist from mean for every data point and assigns score to each of them (low score = normal, high score = anomaly). – Reservoir sampling is a common algo to draw samples from dataset | | | | | | | | | | | | | | |
| - Anomaly detection. – File: CSV, protobuf. – Instance: CPU. – Hyperparams: feature\_dim. – F1 score | | | | | | | | | | | | | | |
| SageMaker IP Insights algo | | Unsupervised learning algo that learns usage patterns for IPv4 by capturing associations btw IP addr & entities. – Stores the IP addr and entity details as key-value pairs. – Queries the data for any login attempts & return a score. – High score = anomalous behaviour. – Uses a Neural Network | | | | | | | | | | | | | | |
| - Anomaly detection. – File: CSV, JSON (inference), JSONL (inference). – Instance: CPU, GPU  - Hyperparams: num\_entity\_vectors, vector\_dim (size of key-value pairs). – Metrics: AUC | | | | | | | | | | | | | | |
| SageMaker K Means | | Unsupervised learning algo that groups similar data. Similarity determined based on specified attributes  - SageMaker version slightly modified. – Euclidean distance used. | | | | | | | | | | | | | | |
| - Unsupervised. – File: CSV, protobuf, JSON (inference). – Instance: CPU, GPU (single instance only)  - Hyperparams: feature\_dim, K. – Metrics: msd (mean square dist), ssd (sum of square dist) | | | | | | | | | | | | | | |
| SageMaker Object2Vec | | | | | Highly customizable neural embedding algo to create vector representations of objects  - Accepts a pair of objects & their r/s labels as inputs. – Each obj initially represented as a random vector  - Adjust these vectors s.t. similar objs are tgt. – Use NN to understand and learn embeddings  - Error in training process back-propagated, and embeddings updated to minimize the loss | | | | | | | | | | | |
| - Neural embedding. – File: Sentence-sentence, label-sentence pair, JSON/JSONL (inference)  - Instance: CPU, GPU. – Hyperparam: enc0\_max\_seq\_len (num of features), enc0\_vocab\_size  - Metrics: mse, accuracy, cross\_entropy | | | | | | | | | | | |
| SageMaker Latent Dirichlet Allocation (LDA) | | | | | | | | | | Unsupervised learning algo to describe a set of observations as a mixture of diff categories  - Generative probabilistic model for discovering the underlying topics in a collection of docs | | | | | | |
| - Unsupervised learning. – File: CSV, protobuf, JSON (inference). – Instance: CPU (single instance)  - Hyperparams: num\_topics, feature\_dim, mini\_batch\_size. – Metrics: per-word-log-likelihood (pwll) | | | | | | |
| SageMaker Neural Topic Model (NTM) | | Unsupervised learning algo to organize a corpus of docs into topics that contain word groupings based on their statistical distribution. - For topic modeling like LDA. – Uses NN | | | | | | | | | | | | | | |
| LDA: Probabilistic graphical model using Dirichlet distribution  Only single instance CPU  Highly interpretable | | | | | | | | | | | | | | NTM: NN model  Can use CPU & GPU instances  Less interpretable |
| - Unsupervised learning. – File: CSV, protobuf, JSON/JSONL (inference). – Instance: CPU, GPU  - Hyperparams: num\_topics, feature\_dim. – Metrics: total\_loss | | | | | | | | | | | | | | |
| SageMaker Blazing Text | | Highly optimized implementation of word2vec and text classification algos  - Based on Facebook’s FastText algo but 20 times faster. – Inference can be done in real-time.  – Expects a single pre-process text file, where ea line contains a single sentence  - Focuses on word-embeddings. For more complex sentences, use Object2Vec | | | | | | | | | | | | | | |
| |  |  |  | | --- | --- | --- | | Modes | Word2Vec (unsupervised) | Text classification (Supervised) | | Single CPU Instance | cbow, skip-gram, Batch skip-gram | Supervised | | ≥ 1 GPU | cbow, skip-gram | Supervised w 1 GPU | | Multiple CPU instances | Batch skip-gram | None | | | | | | | | | | | | | | | |
| - Unsupervised/supervised. – File: space separated tokens, JSON/JSONL (inference). – Instance:CPU,GPU  - Hyperparams: mode. – Metrics: mean\_rho (word2vec, Spearman’s rank correlation coefficient), accuracy | | | | | | | | | | | | | | |
| SageMaker Seq-to-Seq | | Supervised learning algo that uses a NN where a seq of input tokens is transformed into another seq of tokens as output. – Contains 3 layers:  1) Embedding layer: - encoded input tokens mapped to dense feature layer. - initialized w pre-trained word vector like FastText  2) Encoder layer: - compresses input into fixed-length feature vector, usually LSTM or GRU.  3) Decoder layer: - converts encoded feature to an output seq of tokens | | | | | | | | | | | | | | |
| - Supervised. – File: protobuf, JSON (inference only). – Instance: GPU. – No required hyperparams  - Metrics: accuracy, bleu, perplexity | | | | | | | | | | | | | | |
| SageMaker Image Classification algo | | | | | | | | Uses CNN  Modes of operation: 1) Full training mode: Training performed from scratch. Requires large dataset  2) Transfer learning mode: Leverage previously trained images. Requires smaller dataset | | | | | | | | |
| - Supervised. – File: recordIO, image (jgp, png), Requires .lst file. – Instance: GPU, CPU (inference)  - Hyperparams: num\_classes, num\_training\_samples. – Metrics: accuracy | | | | | | | | |
| SageMaker Object Detection algo | | Identifies all instances of objects within image. Uses Single Shot multibox Detector (SSD)  Supports VGG and ResNet networks. Supports full training mode or transfer learning mode | | | | | | | | | | | | | | |
| - Supervised. – File: recordIO, image (jgp, png), Requires .json file for annotation.  – Instance: GPU, CPU (inference). - Hyperparams: num\_classes, num\_training\_samples. – Metrics: mean average precision (MAP) | | | | | | | | | | | | | | |
| SageMaker Semantic Segmentation algo | | | | | | | | Tags every pixel in an image w a class label from a predefined set of classes  Provides info about shapes of objects present in image. Output represented as a grayscale image, aka segmentation mask | | | | | | | | |
| - Supervised. – File: Data required in train (jpg), validation (jgp), train\_annotation (png), validation\_annotation (png) channels. – Instance: GPU, CPU (inference). - Hyperparams: num\_classes, num\_training\_samples. – Metrics: mIOU (Jaccard index), pixel accuracy | | | | | | | | |
| SageMaker Infra | | CPU: For simpler/smaller models  - Low latency. – Less expensive | | | | | | | | | | | | | GPU: For complex/larger models  - High throughput. – More expensive | |
| SageMaker EC2 Instances: 1) General purpose: ml.m5.xlarge, ml.m5.4xlarge, ml.m5.12xlarge  2) Compute-optimized: ml.c5.xlarge, ml.c5.2xlarge, ml.c5.8xlarge  3) Accelerated computing: ml.p3.xlarge, ml.p3.8xlarge, ml.p3.16xlarge | | | | | | | | | | | | | | |
| SageMaker Inference recommender: recommend appropriate EC2 instance to use, instance count, … | | | | | | | | | | | | | | |
| Spot Training: cost effective solution by using unused EC2 capacity at a lower cost  - Instance might be reclaimed by AWS w short notice. – Use checkpoints to save data from local to S3  - Training script must be robust to handle interruptions and restarts  - Recommend to setup CloudWatch events to be notified once job is terminated or interrupted  1) Billable time: absolute clock time (if num of instances > 1, billable time multiplied by instance count)  2) Training time: duration which model is being trained actively  Spot Savings Formula = (1 – Billable time in seconds/training time in seconds) \* 100%  In Estimator obj, args: ‘use\_spot\_instances’ = True, ‘max\_run’ (max duration in sec training job can run)  ‘max\_wait’ (max duration in sec training job can run including time waiting for spot instances)  ‘checkpoint\_s3\_uri’ (S3 location to store checkpoint) | | | | | | | | | | | | | | |
| Distributed Training: trained across multiple machines to reduce training time for DL tasks  1) Data parallelism: training dataset split across multiple GPU instances  - Each GPU has the same model replica. – Use weighted average of gradients to back propagate  - SageMaker Distributed Data Parallelism Library (SMDDP)  - Supported by PyTorch, PyTorch Lighting, Hugging Face Transformers  - Supported by ml.3dn.24xl, ml.p4d.24xl, ml.p4de.24xl  2) Model parallelism: use when model has too many layers that cannot be fit in a single GPU instance  - SageMaker Model Parallelism Library (SMP V2)  - Supported by PyTorch, PyTorch Lighting, Hugging Face Transformers, Hugging Face Accelerate  - Supported by ml.p4d.24xl, ml.p4de.24xl, ml.p5.48xl | | | | | | | | | | | | | | |
| Splitting, Shuffling, Bootstrap Data for Training | | Split data into train, test, validation set | | | | | | | | | | | | | | |
| K-Fold Cross-validation: prevent overfitting but computationally expensive  Stratified K-Fold CV: effective for imbalanced dataset (dist of class same) but computationally expensive  Leave-one Out CV: most computationally expensive | | | | | | | | | | | | | | |
| Data shuffling: Ensures data dist is randomized. – Prevents bias. – Enhance model generalization  - Built-in algo perform internal data shuffling that can be configured.  – Custom python script need explicit data shuffling | | | | | | | | | | | | | | |
| Bootstrapping: create multiple samples of data w replacement  - Improves model stability and variability. – Allows estimation of confidence intervals for model performance metrics. – Helps understand bias-variance trade-off in model predictions  - Requires custom scripts as part of model training process | | | | | | | | | | | | | | |
| Optimization Techniques | | | | | | | Ensure model converge quickly. Prevents overfitting and underfitting  Gradient Descent: learning rate/step size hyperparam  SGD: use a single data point at each iteration  Batch GD: randomly samples a subset of data points at each iteration | | | | | | | | | |
| SageMaker Built-in Algo | | | | | | | Use image\_uris to choose image -> Create an Estimator object + configure training job -> Set hyperparameters -> fit | | | | | | | | | |
| SageMaker custom Training Script | | | | | | | | | | | | | Create an Estimator object + configure training job + specify custom training script (set parameter ‘script\_mode’) -> Set hyperparameters -> fit | | | |
| SageMaker Debugger | | Debug model parameters like weights, gradients, biases of training job  Monitor and profile training jobs by saving the internal model state at regular intervals in real-time  Can detect non-convergence conditions quickly (save cost)  Handles anomalies using integrated tools: - built-in rules to monitor common conditions (computation issues, system bottlenecks, over utilization of resources)  Support multiple ML frameworks and algos: TensorFlow, PyTorch, MXNet, XGBoost | | | | | | | | | | | | | | |
| Modify training script w ‘sagemaker-debugger’ option -> Configure training job w Debugger -> Monitor training job -> Based on rules set, send alerts and take actions | | | | | | | | | | | | | | |
| Best practices: - save training data to S3. – Use Debugger built-in rules to analyse matrix, tensors during training. – Automate actions based on built-in rule status. – Monitor SageMaker infra resource utilization | | | | | | | | | | | | | | |
| SageMaker Canvas | | Drag-and-drop UI for non-technical users to quickly create ML predictions  - Import data from CSV, DBs, data lakes, export back to same source  - Visually explore and prep data. – Auto selects best ML algo. – Auto tune hyperparams  - Share model w other data scientists for further refinement.  – Continuously train & retrain model when newer versions of dataset becomes available  - Ready-to-use models: Sentiment analysis, Entity Recognition, Language Detection, Personal Info detection, Object detection, Text detection, OCR, Document analysis, Document queries | | | | | | | | | | | | | | |
| Over & Under-fitting | | Complex model usually will overfit and have poor generalization. MAE = L1 Loss =  R-Squared (statistical measure of how well a model approximates data) = , TSS = , RSS =  Bias = error due to an over-simplistic model not capturing the data complexity  Variance = how much model results changes. High variance = model interprets noise as signal | | | | | | | | | | | | | | |
| Hyperparam tuning | | | | Model trained and evaluated for ea combi using CV  1) Grid search: exhaustive search mtd that evaluates every possible combination of specified hyperparam values.  2) Random search: selects hyperparams values from specified dist and evaluates random combi  - More efficient than Grid Search.  3) Bayesian Optimization: uses a probabilistic model to determine next best hyperparam combi to use.  – More efficient than Grid or Random search  For AWS SageMaker, hyperparam tuning = Automatic Model Tuning (AMT) | | | | | | | | | | | | |
| Regularization Technique | | | | | | | | | L1 = Lasso Regression. Penalty term = . Tend to shrink weights to 0, hence feature selection  L2 = Ridge Regression. Penalty term = . Tend to shrink weights  Early stopping: stop training when model performance is not improving much  Dropout: randomly drop neurons, generalizing model  - Dropout rate: prob of neuron being dropped during training  - Binary mask: binary value indicating if element will participate in training or not  - Forward pass: data flows from input to output layer.  – Backward pass: model learns from mistakes and calculates error | | | | | | | |
| Cross validation | | | | | | | | | | | | Time Series CV: - Train = t1, test = t2. – Train = t1 + t2, test = t3… - Train = t1 + tn-1, test = tn | | | | |
| Optimize Hyperparam | | | | | | To specify hyperparams using a parameter range in SageMaker:  1) Categorical Params: “CategoricalParameterRanges”: [{“Name”: “…”, “Values”: [“auto”, “exact”…]}]  2) Cts params: ContinuousParameter(0.5, 10, scaling\_type=”Auto”)  3) Integer: IntegerParameter(100, 2000, scaling\_type=”Auto”) | | | | | | | | | | |
| Hyperparam Scaling Types: 1) Auto: auto choose best scale. 2) Linear: select in linear increment  3) Logarithmic: only for positive params for range.  4) Reverse logarithmic: only from ranges btw 0 and 1. Use if range sensitive to small changes  OR do param\_grid = {param1: [], param2: []} # For sklearn | | | | | | | | | | |
| 1) Learning rate: control step size to iterate towards min value of loss function. Low LR = higher cost  2) Batch size: num of training sample used in 1 iteration to train a NN.  - Small batch size = slower convergence = more computing resources required = higher cost  3) Epoch: Num of times whole training dataset is passed forward & backward during training  - More epochs = better training but might overfit. – Minimal impact on resources required but higher cost if more epochs  4) Num of layers: more layers = more memory usage = higher cost | | | | | | | | | | |
| Resource Limits: 1) Max parallel hyperparam tuning jobs = 100  2) Max num of training jobs per tuning job = 750. 3) Concurrent training jobs per tuning job = 10  4) Hyperparams to search = 20. 5) Metrics per tuning job = 20. 6) Run time per tuning job = 30 days | | | | | | | | | | |
| 1) Early stopping: get value of objective metric, compute median, compare w previous jobs  - Supported algo: Linear Learner, XGBoost, Image Classification, Obj Detection, Seq-to-Seq, IP Insights  2) Warm Start: leverage previously concluded training job, conserve resources, longer start time as need load resources from previous job | | | | | | | | | | |
| To create and run Hyperparameter Tuning Job  hyperparam\_ranges = {“hyperparam1”, ContinuousParameter(), “hyperparam2”, IntegerParameter()}  hp\_tuner = HyperparameterTuner(model, objective function, hyperparam\_ranges, max\_jobs = 6,  max\_parallel\_jobs = 2, objective\_type = “Maximize”)  hp\_tuner.fit(inputs={“train”: s3\_train, “validation”: s3\_validation}, job\_name = “”) | | | | | | | | | | |
| Confusion Matrix | Can be used to evaluate binary & multiclass classification  FP = Type I error. FN = Type II error | | | | | | | | | | | | | | | |
| Classification metrics | | | | | | Accuracy = (simple to calculate & easy to understand, not suitable for imbalance data)  Precision = = % of +ve predictions correct (to minimize FP, but don’t consider FN)  Recall = = Sensitivity/TPR = % of actual +ve, how many predicted (min FN, but don’t consider FP)  Specificity/TNR = = % of actual -ve, how many predicted (don’t consider FN)  F1 score = = harmonic mean btw precision and recall (f1 score is lower when either P or R is lower compared to using arithmetic mean, hence balancing both metric; less interpretable)  AUC-ROC curve (receiver operating curve) = plots r/s btw TPR and FPR (1 – specificity) for diff threshold - higher AUC = better | | | | | | | | | | |
| Regression metrics | | MAE = L1 Loss = (insensitive to outliers, treats all errors equally)  MSE = (less interpretable compared to MAE, useful for optimization in ML algo)  RMSE = (easily interpretable, sensitive to outliers)  MAPE (percentage) = (similar to MAE, useful for comparing model performance, sensitive to 0 values)  , TSS = , RSS = | | | | | | | | | | | | | | |
| Online Model Evaluation | | Online evaluation = use live data & input data is not split  - Provides real-time feedback and immediate insights. Performance degradation can be quickly detected  - But resource intensive. Can suffer from data drift, leading to degraded performance | | | | | | | | | | | | | | |
| Additional metrics: 1) Latency = time taken for model to generate predictions since it received data  2) Throughput = num of predictions model can maker per unit of time  3) Data drift = similarity of data on which model was initially trained  4) Click-through rate = ratio of users who click on a specific link over total num of users to whom the recommendations are made  5) Conversion rate = % of successful outcomes from total num of recommendations made | | | | | | | | | | | | | | |
| A/B Testing on SageMaker. – Can deploy multiple version of models behind the same endpoint  - Specify % of traffic for each model version by specifying the weight for ea production variant  - OR can set target variant header in request to specify model variant to use | | | | | | | | | | | | | | |
| Comparing ML Models using Production Parameters | | | 1) Time complexity = time taken by algo for a given input data size.  Let n = num of training samples, f = num of features  For Linear regression, training complexity = O(n \* f2 + f3). Testing complexity = O(f)  For Logistic regression, training complexity = O(n \* f). Testing complexity = O(f)  2) Space complexity = additional memory algo needs for a given input data size  Linear regression = O(f). Logistic regression = O(f)  3) Sample complexity = num of training samples required for training to gain desired level of performance  Higher dimensionality or dataset w higher noise will require more data samples  4) Parametricity: whether model is parametric or non-parametric  - Parametric model: fixed num of params, faster to train & easier to interpret (Linear/Logistic regression)  - Non-parametric model: num of params can grow w training data. More flexible as no assumption on data (KNN, decision tree) | | | | | | | | | | | | | |
| SageMaker Debugger | | train\_data = sagemaker.inputs.TrainingInput(s3\_path, distribution=”FullyReplicated”)  In model Estimator(role, instance\_count, instance\_type, …,  debugger\_hook\_config = DebuggerHookConfig(s3\_output\_path = …,  collection\_configs = [CollectionConfig(name=”metrics”, parameters= …  CollectionConfig(name=”feature\_importance”, parameters={“save\_interval”: str(save\_interval)})]),  rules = [Rule.sagemaker(rule\_configs.loss\_not\_decreasing(), rule\_parameters={}),  Rule.sagemaker(rule\_configs.create\_xgboost\_report())]  )  Estimator.fit() | | | | | | | | | | | | | | |

AWS Panaroma: add CV to an on-premises camera network

AWS IoT Greengrass: For IoT devices to run w/o internet access. Can use for CV if no existing camera available

AWS Polly: for custom sounds (acronyms) -> Create an appropriate pronunciation lexicon = file that defines how words/phrases should be pronounced by Polly. Must follow Pronunciation Lexicon Specification (PLS) standard

SageMaker Training job: need to specify 1) location of training data on S3. 2) IAM role for SageMaker to assume. 3) Output path specifying which S3 to persist model in

To use custom algo in SageMaker, Docker container image must have an executable file named “train”, that acts as ENTRYPOINT for the container.

Feature store w data in S3. To share features across accounts: 1) create IAM role to allow access to feature store

2) Share feature repo by using AWS Resource Access Manager (AWS RAM)

- Resource share = entity defining resources to share and principals to share with

To configure custom Docker container to leverage NVIDIA GPUs: build docker container to be NIVIDIA-Docker compatible

TensorFlow ML model on single GPU taking too long. Need update model on hourly basis:

- Change TensorFlow code to implement a Horovod distributed framework supported by SageMaker.

Amazon Lex: Custom Slot Types

- Categories spoken by users outside defined types but related: Add unrecognized words as synonyms in custom slot type

Glue to create datasets in S3. To access ETL scripts directly from SageMaker notebooks within a VPC. Also run Glue job and invoke SageMaker training job:

- Create AWS Glue development endpoint in VPC. – Create SageMaker notebooks by using Glue development endpoint. – Create IAM policy and IAM role for SageMaker notebooks

SageMaker DeepAR model evaluation metrics: Coverage score should be approx equal to the quantile itself

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ML Implementation and Operations | | | | | | |
| Deploy model using SageMaker | | | Create a SageMaker endpoint (HTTPS engpoint). Client app can call SageMaker API InvokeEndpoint()  SageMaker hosted endpoint using EC2 instance. For HA, can create auto-scaling group of EC2  EC2 run containers using ECS | | | |
| from sagemaker.serializers import CSVSerializer # to serialize input data to CSV formatted string  xgb\_predictor = xgb\_model.deploy(initial\_instance\_count=2, instance\_type = “ml.t2.medium”, serializer = CSVSerializer())  xgb\_predictor.endpoint\_name | | | |
| 1) Serverless endpoint use Lambda under the hood. No infra to manage  Suitable for workloads that have idle periods btw requests which can tolerate cold starts  2) EC2: select deep learning AMI and an appropriate instance type (e.g. P5)  3) ECS: create docker container w own model and deploy using ECS | | | |
| Security Best practices | | | | | S3 bucket should enable encryption (S3 or KMS) and have appropriate bucket policies  SageMaker Execution roles: IAM role-based access | |
| Multi Region and AZ | High availability (HA): system continues to function if any single component fails  Fault tolerance: system continues to function w/o degradation in performance if any single component fails  AWS Well-architected framework: Reliability pillar  To avoid single point of failure, in VPC, can configure 2 subnets: each in a diff AZ. Use ELB to route traffic | | | | | |
| VPC Endpoints | | | Ensures secure communication btw VPC and AWS services  - Uses AWS PrivateLink instead of the public internet | | | |
| 1) Interface endpoint: VPC endpoint is associated w ≥ 1 private IP addresses in VPC subnets  - SageMaker runtime, endpoint sits in a SageMaker VPC  - For your app in your custom VPC to connect to SageMaker, put VPC endpoint at your custom VPC  - Add VPC endpoint policy to allow access to IAM users  - Modify security group on endpoint network interface to restrict access to specific instances | | | |
| Multi-model endpoints | | | Can deploy multiple models in 1 container behind a single endpoint  - Scalable, cost effective, Common inference frameworks (PyTorch, MXNet, XGBoost, scikit-learn, …) are supported | | | |
| SageMaker Batch Transform | | | Use for processing of large datasets and don’t need a persistent endpoint | | | |
| transformer = estimator.transformer(initial\_instance\_count=1, instance\_type=”ml.m5.large”)  transformer.transform(s3\_path, content\_type=”text/csv”, split\_type=”Line”)  transformer.wait() | | | |
| Creating AMI and Golden Images | | | AMI = preconfigured Amazon Machine Image = Golden Image  - Used as reference for creating EC2 instances that need the same config  - Includes OS, software packages, configs. – AMI can be used w auto scaling to auto provision infra  From an existing running EC2 instance -> Create image | | | |
| Monitoring and Logging w CloudWatch | | | CloudWatch collects utilization metrics for training job instances (CPU, memory, GPU utilization)  CloudWatch also integrates w SNS and can send email/notification if metric breaches threshold  CloudWatch logs: monitor and centrally store log files generated by apps on EC2  CloudWatch events: near real-time events or changes  CloudWatch alarms: notifications based on thresholds configured | | | |
| Logging SageMaker API calls w CloudTrail | | | | | | CloudTrail can log API calls. For SageMaker: CreateTrainingJob, CreateModel, CreateEndpoint  Most recent events from last 90 days available in CloudTrail console Event history  To store CloudTrail data, create a trail to deliver log files to S3 bucket |
| ML Stack | 1) Infra services: support ML frameworks like EC2 instance types w specialist CPUs, ECS, DL AMIs, Docker images  2) SageMaker: remove heavy lifting involved in building and training models, deploy to prod and running models at scale. Remove managing infra  3) AI services: augmented w AI/ML, like Textract, Transcribe, Translate, Lex, Polly, Rekognition, Comprehend | | | | | |
| AWS Service Quotas | | | Service Quotas/Limits: limit num of services you can provision on a per-account basis  - Centrally manage across multiple services in AWS. – Quota increases (request in AWS console)  AWS Trusted Advisor: to run service limit checks (will warn if > 80% of limit) | | | |
| AI services | | 1) Transcribe: Speech-to-text. – Input can be audio files (MP3) or streamed audio  2) Comprehend: NLP to process text. Discover insights, meaning, connections in text data  - Sentiment analysis, Identify language, Topic modeling, Part of Speech tagging, Key phrase extraction  3) Translate: language translation, 70+ language supported, can customize w own terminology  4) Lex: chatbot, uses NLU (natural language understanding) and has integration w Lambda for executing logic. – Multi-platform compatibility (phones, web apps, chat services)  - Auto speech recognition, voice and text conversation  5) Rekognition: analyze images and videos  - Object detection, image labelling, analyze image properties, image moderation, facial analysis, face comparison, face liveness (ensure real person), text in image, PPE detection  6) Textract: extracts text & data from documents  - Identify, understand and extract specific data from scanned docs (PDFs, images, tables, forms)  - Can be used for handwritten text through OCR  7) Polly: generate speech from text. – Can provide text in variety of languages  - Resulting audio can be streamed, saved or downloaded  8) Forecast: performing forecasting for time-series data  - Input time-series data (+ related data) -> Forecast -> Train custom model -> Provide predictions  9) Fraud Detector: Identify fraudulent activities customized on your data  - S3 data -> Fraud Detector -> Customized model -> Prediction API (real-time) -> Fraud score (0 – 1000) | | | | |
| Identity Access Management (IAM) | | | | Centralized control of AWS acct, providing Granular permissions  - Supports security compliance frameworks like PCI DSS, FedRAMP, SOC, ISO | | |
| 1) Users. 2) Groups: collection of users. 3) Roles: can be assigned to users, apps, services | | |
| IAM Policy: document that defines ≥ 1 permissions and can be attached to user, group or role  - Can add condition to limit access to actions and resources  - Effect (allow, deny), Action (create, upload, delete), Resource, Condition | | |
| S3 bucket policies | | | All newly created buckets are private, no public access by default  - Only bucket owner can upload, read or delete files  - Define S3 bucket policies (JSON) to define access to grant  - Applied at bucket level: applies to all objects within bucket. – Can be used to deny access  - Effect, Principal (user, group or role), Action, Resource, Condition | | | |
| VPCs | | | Uses Network access control lists (NACLs) and security groups (SG) to control access to EC2 instances in subnets. NACL is subnet level, SG at resource level  VPC sits within a single region. At VPC level, can configure Internet gateway (inbound & outbound) or NAT gateway (outbound only)  Internet gateway <-> Route table <-> Network ACL/Public subnet <-> Security Group/Resource  NAT gateway <- Route table <- NACL/Private subnet <- SG/Resource | | | |
| Security Groups | | | Controls traffic allowed to reach and leave resources it is associated with.  All traffic denied by default. You only define the allowed traffic (stateful).  All rules get evaluated before logic is applied, most restrictive match wins  SG Components: - Protocol (TCP, UDP, ICMP). – Port range (22, 443, 1024-6535). – Src or dest IP ranges OR other security groups | | | |
| Encryption | | | Encryption in Transit: SSL/TLS or HTTPS  Encryption at Rest: SSE-S3 by default, SSE-KMS, SSE-C or Client-Side Encryption | | | |
| KMS is a manged service to create and control keys to encrypt data  Integrates w S3, EBS, EFS, FSx, RDS, DynamoDB, Athena, Redshift, Kinesis, EMR, ECR, EKS, CloudWatch Logs, OpenSearch, CloudTrail | | | |
| CMK (Customer Master Key): encrypt/decrypt data up to 4KB. For envelope encryption (2 encryption)  - Generate, encrypt, decrypt data key. – Data key is used to encrypt/decrypt data | | | |
| Anonymizing Data | | | | | | Athena -> analyze data in S3 -> Perform anonymization -> Store in another S3 bucket |
| Deploy to endpoint | | | create\_model(name=model\_1, role=role, container\_defs={“Image”: image\_uri1, “ModelDataUrl”: })  variant1 = production\_variant(model\_name=model\_1, instance\_type, initial\_instance\_count…)  session.endpoint\_from\_production\_variants(name=endpoint\_name, production\_variants=[variant1, variant2])  session.invoke\_endpoint(EndpointName=endpoint\_name, ContentType=”text/csv””, Body=payload)  invoke\_endpoint(…, TargetVariant=variant1[“VariantName”]) # To call specific variant  Can specify weight to route traffic | | | |
| Retraining Pipelines w Step Functions | | | Serverless orchestration tool to manage logic of workflow  - Create and visualize ML workflows or pipelines, like auto retraining when new data is available  E.g. Lambda -> CreateProcessingJob -> CreateTrainingJob -> CreateModel -> CreateTransformJob (Batch transform) | | | |
| Monitor Model Performance | | | SageMaker Model Monitor: monitor quality of ML models  1) Continuous monitoring of real-time endpoints and batch transform jobs  2) Analyze model predictions to determine model performance and quality  3) Compare predictions w known ground truth data | | | |
| 1) Drift in data quality. 2) Drift in model quality metrics (accuracy, RMSE)  3) Bias in model predictions. 4) Model explainability (feature attribution drift)  - Set alerts to notify you (integrates w CloudWatch) | | | |
| Create Baseline -> Set up monitoring schedule -> Compare w Predictions -> Check if predictions differs from baseline | | | |

Blue-green deployment: running 2 identical envs (blue, green) and switching traffic btw them

Canary deployment: deploy new model to small subset of users and monitoring its performance before rolling out to all

Rolling deployment: gradually replace old model w new one across all instances

A/B testing more for evaluation of diff versions and not for initial deployment

SageMaker Variant Invocations per Instance setting = target value for average num of invocations per instance per min for model variant. – Used by auto scaling policy to add or remove instances to keep metric close to specified value

- Formula = MAX\_RPS \* SAFETY\_FACTOR \* 60

Amazon Personalize: product recommendation

- Use event tracker in Personalize to include real-time interactions -> allow model to learn from customers feedback

AWS DeepLens: video camera optimized for DL in AWS

AWS Rekognition to identify staff members from image stored in S3:

- Create & populate a Rekognition Collection w existing profile pics. Call API SearchFacesByImage operation passing in both Collection ID and S3 location of image

Athena Federated Query feature: enables data from a variety of sources to be queried in place using SQL

Launch EC2 w DL AMI, to access from local and run Jupyter notebooks:

- Start Jupyter Notebook service on EC2 + add rule to security group to allow inbound access on port 22 + create an SSH local port forwarding to default port 8888 that Jupyter listens on

SageMaker Elastic Inference (EI): accelerate throughput and decrease latency

- Reduce cost by allowing users to attach desired amt of GPU-powered INFERENCE acceleration to an instance w/o code changes. Supports TensorFlow, PyTorch, MXNet

For SageMaker to use model trained locally:

- Build Docker image w inference code. Tag Docker image w registry hostname and upload to ECR

AWS Deep Learning AMI supports MXNet, CNTK, Keras vs AWS DL Containers support TensorFlow, PyTorch, MXNet

VPC Peering connections: allows cross-VPC connectivity btw AWS resources.

- Route tables also need to be configured to ensure instance subnets can route to each other

Amazon Augmented AI (Amazon A2I): allows you to conduct a human review of ML systems to guarantee precision.

Amazon Forecast algos: ARIMA, ETS for baseline, Prophet slightly better. CNN-QR, DeepAR+ best

SageMaker notebook inside VPC w interface endpoints. All connections contained in VPC and secured

- Still can access from internet as need to: Create IAM policy that allows sagemaker:CreatePresignedNotebookInstanceUrl and sagemaker:DesribeNotebookInstances actions from only the VPC endpoints. Apply policy to all IAM users, groups and roles used to access the instances

SageMaker notebook instances are based on EC2 instances running within AWS service account (inside aws vpc)