**Algorithms**

**Blazing Text**

* Highly optimized implementations of Word2Vec and text classification algorithms
* The Word2Vec algorithm is useful for many downstream NLP tasks
  + Sentiment analysis
  + Named entity recognition
  + Machine translation
* Text classification can be used for
  + Web searches
  + Information retrieval
  + Ranking
  + Document classification
* The Word2Vec algorithm maps words to high-quality distributed vectors
  + The resulting vector representation of a word is called a word embedding
  + Words that are semantically similar correspond to vectors that are closer together
  + Word embeddings capture the semantic relationships between words
* NLP applications learn word embeddings by training on large collections of documents
  + These pretrainied vector representations provide information about semantics and word distributions that typically improves the generalizability of other models that are later trained on a more limited amount of data.
  + Most implementations of the Word2Vec are not optimized for multi-core CPU architectures
    - This makes it difficult to scale to large datasets
  + With BlazingText you can scale to large datasets easily
    - BlazingText provides Skip-gram and continuous bag-of-words (CBOW) training architectures.
    - BalzingText’s implementation of the supervised multi-class, multi-label text classification algorithm extends the fastText text classifier to use GPU acceleration with custom CUDA kernels
    - Can train a model on more than a billion words in a couple of minutes using a multi-core CPU or a GPU.
* Generates meaningful vectors for out-of-vocabulary (OOV) words by representing their vectors as the sum of the character n-gram (subword) vectors.
* Can use A batch\_skipgram mode for the Word2Vec algorithm that allows faster training and distributed computation across multiple CPU nodes.

**DeepAR Forecasting**

* A supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNN).
* Trains a single model jointly over all of the time series
* Can also use the trained model to generate forecasts for new time series that are similar to the ones it has been trained on.
* The training input for the DeepAR algorithm is one or, preferably, more target time series that have been generated by the same process or similar processes.
  + The algorithm trains a model that learns an approximation of this process/processes and uses it to predict how the target time series evolves.

**Factorization Machines**

* A factorization machine is a general-purpose supervised learning algorithm that you can use for both classification and regression tasks.
* It is an extension of a linear model that is designed to capture interactions between features within high dimensional sparse datasets economically.
* For example, in a click prediction system, the factorization machine model can capture click rate patterns observed when ads from a certain ad-category are placed on pages from a certain page-category.
* Factorization machines are a good choice for tasks dealing with high dimensional sparse datasets, such as click prediction and item recommendation.
* The Amazon SageMaker implementation of factorization machines considers only pair-wise (2nd order) interactions between features.

**Image Classification**

* The Amazon SageMaker image classification algorithm is a supervised learning algorithm that supports multi-label classification.
* It takes an image as input and outputs one or more labels assigned to that image.
* It uses a convolutional neural network (ResNet) that can be trained from scratch or trained using transfer learning when a large number of training images are not available.
* The recommended input format for the Amazon SageMaker image classification algorithms is Apache MXNet RecordIO.
  + However, you can also use raw images in .jpg or .png format.
* Image Classification Algorithm is a supervised learning algorithm that analyzes only whole images, classifying them into one of multiple output categories.

**IP Insights**

* Amazon SageMaker IP Insights is an unsupervised learning algorithm that learns the usage patterns for IPv4 addresses.
* It is designed to capture associations between IPv4 addresses and various entities, such as user IDs or account numbers.
  + You can use it to identify a user attempting to log into a web service from an anomalous IP address, for example.
  + You can use it to identify an account that is attempting to create computing resources from an unusual IP address.
* Trained IP Insight models can be hosted at an endpoint for making real-time predictions or used for processing batch transforms.
* Amazon SageMaker IP insights ingests historical data as (entity, IPv4 Address) pairs and learns the IP usage patterns of each entity.
* When queried with an (entity, IPv4 Address) event, an Amazon SageMaker IP Insights model returns a score that infers how anomalous the pattern of the event is.
  + For example, when a user attempts to log in from an IP address, if the IP Insights score is high enough, a web login server might decide to trigger a multi-factor authentication system.
* In more advanced solutions, you can feed the IP Insights score into another machine learning model.
  + For example, you can combine the IP Insight score with other features to rank the findings of another security system, such as those from Amazon GuardDuty.
* The Amazon SageMaker IP Insights algorithm can also learn vector representations of IP addresses, known as embeddings.
* You can use vector-encoded embeddings as features in downstream machine learning tasks that use the information observed in the IP addresses.
  + For example, you can use them in tasks such as measuring similarities between IP addresses in clustering and visualization tasks.

**K-Means**

* K-means is an unsupervised learning algorithm.
* It attempts to find discrete groupings within data, where members of a group are as similar as possible to one another and as different as possible from members of other groups.
* You define the attributes that you want the algorithm to use to determine similarity.
* To improve performance over the open source version, streams mini-batches (small, random subsets) of the training data.
* The k-means algorithm expects tabular data, where rows represent the observations that you want to cluster, and the columns represent attributes of the observations.
* The n attributes in each row represent a point in n-dimensional space.
* The Euclidean distance between these points represents the similarity of the corresponding observations.
* The algorithm groups observations with similar attribute values (the points corresponding to these observations are closer together).

**K-Nearest Neighbors**

* Amazon SageMaker k-nearest neighbors (k-NN) algorithm is an index-based algorithm.
* It uses a non-parametric method for classification or regression.
* For classification problems, the algorithm queries the k points that are closest to the sample point and returns the most frequently used label of their class as the predicted label.
* For regression problems, the algorithm queries the k closest points to the sample point and returns the average of their feature values as the predicted value.
* Training with the k-NN algorithm has three steps: sampling, dimension reduction, and index building.
  + Sampling reduces the size of the initial dataset so that it fits into memory.
  + For dimension reduction, the algorithm decreases the feature dimension of the data to reduce the footprint of the k-NN model in memory and inference latency.
  + AWS provides two methods of dimension reduction methods: random projection and the fast Johnson-Lindenstrauss transform.
  + Typically, you use dimension reduction for high-dimensional (d >1000) datasets to avoid the “curse of dimensionality” that troubles the statistical analysis of data that becomes sparse as dimensionality increases.
  + The main objective of k-NN's training is to construct the index. The index enables efficient lookups of distances between points whose values or class labels have not yet been determined and the k nearest points to use for inference.

**Latent Dirichlet Allocation**

* The Amazon SageMaker Latent Dirichlet Allocation (LDA) algorithm is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories.
* LDA is most commonly used to discover a user-specified number of topics shared by documents within a text corpus.
  + Each observation is a document, the features are the presence (or occurrence count) of each word, and the categories are the topics.
  + Since the method is unsupervised, the topics are not specified up front, and are not guaranteed to align with how a human may naturally categorize documents.
  + The topics are learned as a probability distribution over the words that occur in each document. Each document, in turn, is described as a mixture of topics.

**Linear Learner**

* Linear models are supervised learning algorithms used for solving either classification or regression problems.
* For input, you give the model labeled examples (x, y).
  + x is a high-dimensional vector and y is a numeric label.
* For binary classification problems, the label must be either 0 or 1.
* For multiclass classification problems, the labels must be from 0 to num\_classes - 1.
* For regression problems, y is a real number.
* The algorithm learns a linear function, or, for classification problems, a linear threshold function, and maps a vector x to an approximation of the label y.
* With the Amazon SageMaker algorithm, you can simultaneously explore different training objectives and choose the best solution from a validation set.
* You can also explore a large number of models and choose the best.
* The best model optimizes either of the following:
  + Continuous objectives, such as mean square error, cross entropy loss, absolute error.
  + Discrete objectives suited for classification, such as F1 measure, precision, recall, or accuracy.
* The linear learner algorithm requires a data matrix, with rows representing the observations, and columns representing the dimensions of the features.
* It also requires an additional column that contains the labels that match the data points.
* At a minimum, Amazon SageMaker linear learner requires you to specify input and output data locations, and objective type (classification or regression) as arguments.
* The feature dimension is also required.

**Neural Topic Model**

* Amazon SageMaker NTM is an unsupervised learning algorithm that is used to organize a corpus of documents into topics that contain word groupings based on their statistical distribution.
  + Documents that contain frequent occurrences of words such as "bike", "car", "train", "mileage", and "speed" are likely to share a topic on "transportation" for example.
* Topic modeling can be used to classify or summarize documents based on the topics detected or to retrieve information or recommend content based on topic similarities.
* The topics from documents that NTM learns are characterized as a latent representation because the topics are inferred from the observed word distributions in the corpus.
* The semantics of topics are usually inferred by examining the top ranking words they contain.
* Because the method is unsupervised, only the number of topics, not the topics themselves, are prespecified. In addition, the topics are not guaranteed to align with how a human might naturally categorize documents.
* Topic modeling provides a way to visualize the contents of a large document corpus in terms of the learned topics.
* Documents relevant to each topic might be indexed or searched for based on their soft topic labels.
* The latent representations of documents might also be used to find similar documents in the topic space.
* You can also use the latent representations of documents that the topic model learns for input to another supervised algorithm such as a document classifier.
* Because the latent representations of documents are expected to capture the semantics of the underlying documents, algorithms based in part on these representations are expected to perform better than those based on lexical features alone.

**Object2Vec**

* The Amazon SageMaker Object2Vec algorithm is a general-purpose neural embedding algorithm that is highly customizable.
* It can learn low-dimensional dense embeddings of high-dimensional objects.
* The embeddings are learned in a way that preserves the semantics of the relationship between pairs of objects in the original space in the embedding space.
  + You can use the learned embeddings to efficiently compute nearest neighbors of objects and to visualize natural clusters of related objects in low-dimensional space, for example.
  + You can also use the embeddings as features of the corresponding objects in downstream supervised tasks, such as classification or regression.
* Object2Vec generalizes the well-known Word2Vec embedding technique for words that is optimized in the Amazon SageMaker BlazingText Algorithm.

**Object Detection**

* The Amazon SageMaker Object Detection algorithm detects and classifies objects in images using a single deep neural network.
* It is a supervised learning algorithm that takes images as input and identifies all instances of objects within the image scene.
* The object is categorized into one of the classes in a specified collection with a confidence score that it belongs to the class.
* Its location and scale in the image are indicated by a rectangular bounding box.
* It uses the Single Shot multibox Detector (SSD) framework and supports two base networks: VGG and ResNet
* The network can be trained from scratch, or trained with models that have been pre-trained on the ImageNet dataset.
* A supervised learning algorithm that detects and classifies all instances of an object in an image. It indicates the location and scale of each object in the image with a rectangular bounding box.

**Principal Component Analysis**

* PCA is an unsupervised machine learning algorithm that attempts to reduce the dimensionality (number of features) within a dataset while still retaining as much information as possible.
* This is done by finding a new set of features called components, which are composites of the original features that are uncorrelated with one another.
* They are also constrained so that the first component accounts for the largest possible variability in the data, the second component the second most variability, and so on.
* In Amazon SageMaker, PCA operates in two modes, depending on the scenario:
  + 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. This mode uses an approximation algorithm.
* PCA uses tabular data.
  + The rows represent observations you want to embed in a lower dimensional space.
  + The columns represent features that you want to find a reduced approximation for.
* The algorithm calculates the covariance matrix (or an approximation thereof in a distributed manner), and then performs the singular value decomposition on this summary to produce the principal components.

**Random Cut Forest**

* Amazon SageMaker Random Cut Forest (RCF) is an unsupervised algorithm for detecting anomalous data points within a data set.
* These are observations which diverge from otherwise well-structured or patterned data.
* Anomalies can manifest as unexpected spikes in time series data, breaks in periodicity, or unclassifiable data points.
* They are easy to describe in that, when viewed in a plot, they are often easily distinguishable from the "regular" data.
* Including these anomalies in a data set can drastically increase the complexity of a machine learning task since the "regular" data can often be described with a simple model.
* With each data point, RCF associates an anomaly score.
  + Low score values indicate that the data point is considered "normal."
  + High values indicate the presence of an anomaly in the data.
  + The definitions of "low" and "high" depend on the application but common practice suggests that scores beyond three standard deviations from the mean score are considered anomalous.
* While there are many applications of anomaly detection algorithms to one-dimensional time series data such as traffic volume analysis or sound volume spike detection, RCF is designed to work with arbitrary-dimensional input.
* Amazon SageMaker RCF scales well with respect to number of features, data set size, and number of instances.

**Semantic Segmentation**

* The Amazon SageMaker semantic segmentation algorithm provides a fine-grained, pixel-level approach to developing computer vision applications.
* It tags every pixel in an image with a class label from a predefined set of classes.
* Tagging is fundamental for understanding scenes, which is critical to an increasing number of computer vision applications, such as self-driving vehicles, medical imaging diagnostics, and robot sensing.
* Because the semantic segmentation algorithm classifies every pixel in an image, it also provides information about the shapes of the objects contained in the image.
* Amazon SageMaker semantic segmentation algorithm is built using the MXNet Gluon framework and the Gluon CV toolkit
* It provides you with a choice of three built-in algorithms to train a deep neural network.
  + Fully-Convolutional Network (FCN) algorithm
  + Pyramid Scene Parsing (PSP) algorithm
  + DeepLabV3
* Each of the three algorithms has two distinct components:
  + The *backbone* (or *encoder*)—A network that produces reliable activation maps of features.
  + The *decoder*—A network that constructs the segmentation mask from the encoded activation maps.
* You also have a choice of backbones for the FCN, PSP, and DeepLabV3 algorithms:
  + ResNet50
  + ResNet101
  + These backbones include pretrained artifacts that were originally trained on the ImageNet classification task.
  + You can fine-tune these backbones for segmentation using your own data.
  + Or, you can initialize and train these networks from scratch using only your own data.
* The decoders are never pretrained.
* To deploy the trained model for inference, use the Amazon SageMaker hosting service.
  + During inference, you can request the segmentation mask either as a PNG image or as a set of probabilities for each class for each pixel.
  + You can use these masks as part of a larger pipeline that includes additional downstream image processing or other applications.

**Sequence2Sequence**

* Amazon SageMaker Sequence to Sequence is a supervised learning algorithm where the input is a sequence of tokens (for example, text, audio) and the output generated is another sequence of tokens.
* Example applications include: machine translation (input a sentence from one language and predict what that sentence would be in another language), text summarization (input a longer string of words and predict a shorter string of words that is a summary), speech-to-text (audio clips converted into output sentences in tokens).
* Recently, problems in this domain have been successfully modeled with deep neural networks that show a significant performance boost over previous methodologies.
* Amazon SageMaker seq2seq uses Recurrent Neural Networks (RNNs) and Convolutional Neural Network (CNN) models with attention as encoder-decoder architectures.

**XGBoost**

* The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm.
* Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler, weaker models.
* XGBoost has done remarkably well in machine learning competitions because it robustly handles a variety of data types, relationships, and distributions, and because of the large number of hyperparameters that can be tweaked and tuned for improved fits.
* This flexibility makes XGBoost a solid choice for problems in regression, classification (binary and multiclass), and ranking.
* Provides an XGBoost estimator that executes a training script in a managed XGBoost environment.
* The XGBoost algorithm can be used as a built-in algorithm or as a framework such as TensorFlow.
* Using XGBoost as a framework provides more flexible than using it as a built-in algorithm as it enables more advanced scenarios that allow pre-processing and post-processing scripts to be incorporated into your training script.
* Using XGBoost as a built-in Amazon SageMaker algorithm is how you had to use the original XGBoost Release 0.72 version and nothing changes here except the version of XGBoost that you use.
* Use XGBoost as a framework to run scripts that can incorporate additional data processing into your training jobs.