**Content**

[1. Intro 3](#_Toc30255103)

[2. Data Engineering 4](#_Toc30255104)

[1. Creating data repositories for ML 4](#_Toc30255105)

[2.1.1. AWS Data Lake 4](#_Toc30255106)

[2.1.2. Amazon FX for Lustre 5](#_Toc30255107)

[2. Identify and implement a data ingestion solutions 5](#_Toc30255108)

[2.2.1. Glue 5](#_Toc30255109)

[2.2.2. Amazon Kinesis Producer Library (KPL) and Amazon Kinesis Client Library (KCL) 6](#_Toc30255110)

[3. Identify and implement a data transformation solution 6](#_Toc30255111)

[2.3.1. EMR – Elastic MapReduce 6](#_Toc30255112)

[2.3.2. Apache Kafka 7](#_Toc30255113)

[3. Exploratory Data Analysis 8](#_Toc30255114)

[1. Sanitize and prepare data for modeling 8](#_Toc30255115)

[3.1.1. Data Augmentation 8](#_Toc30255116)

[2. Perform feature engineering 9](#_Toc30255117)

[3. T-SNE (t-Distributed Stochastic Neighbor Embedding) 9](#_Toc30255118)

[3.3.1. T-SNE vs PCA 9](#_Toc30255119)

[4. Analyze and visualize data for ML 11](#_Toc30255120)

[3.4.1. Correlation and Covariance 11](#_Toc30255121)

[4. Modeling 14](#_Toc30255122)

[1. Frame business problems as ML problems 14](#_Toc30255123)

[2. Select the appropriate model(s) for an ML problem 14](#_Toc30255124)

[4.2.1. XGBoost 14](#_Toc30255125)

[3. Train ML models 15](#_Toc30255126)

[4. Perform hyperparameter optimization 15](#_Toc30255127)

[4.4.1. Regularization 15](#_Toc30255128)

[4.4.2. Momentum 16](#_Toc30255129)

[4.4.3. AdaGrad 17](#_Toc30255130)

[4.4.4. Sample Size and Effective Sample size 17](#_Toc30255131)

[5. ML Implementation and Operations 18](#_Toc30255132)

[1. Build ML solutions for performance, availability, scalability, resiliency, and fault tolerance 18](#_Toc30255133)

[2. Recommend and implement the appropriate ML services and features for a given problem 18](#_Toc30255134)

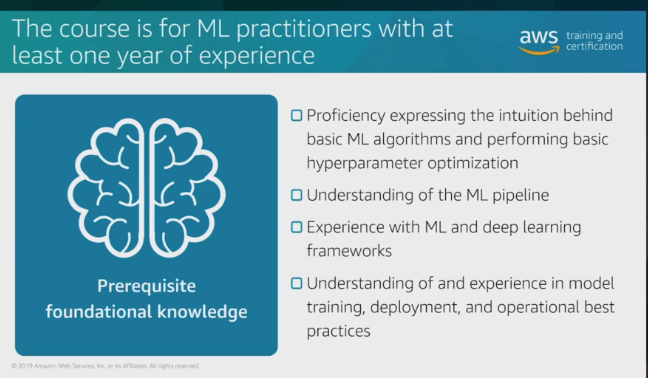
[3. Apply Basic AWS security practices to ML solutions 19](#_Toc30255135)

[4. Deploy and operationalize ML solutions 19](#_Toc30255136)

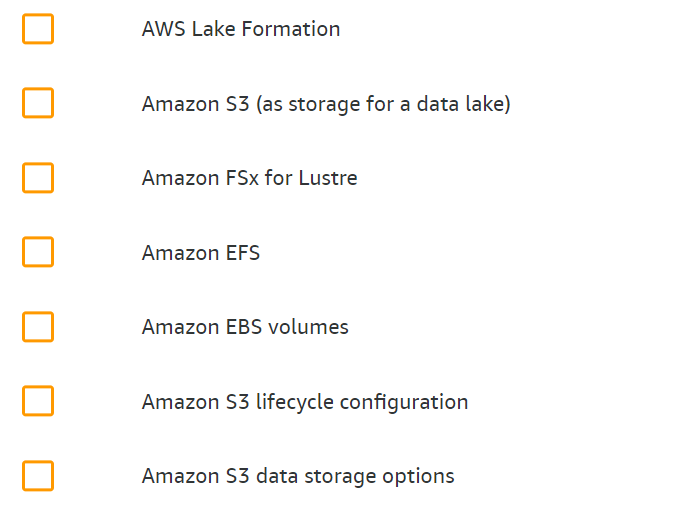
1. Intro

Link:

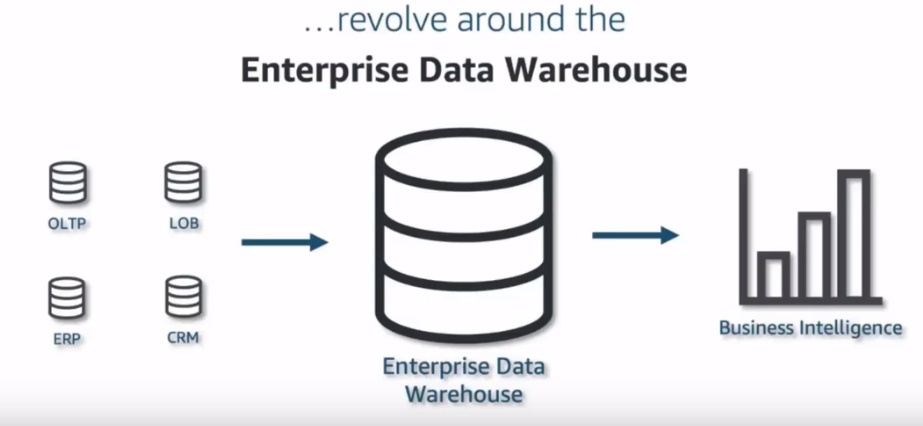
<https://www.aws.training/Account/Transcript/Current>



1. Data Engineering
   1. Creating data repositories for ML



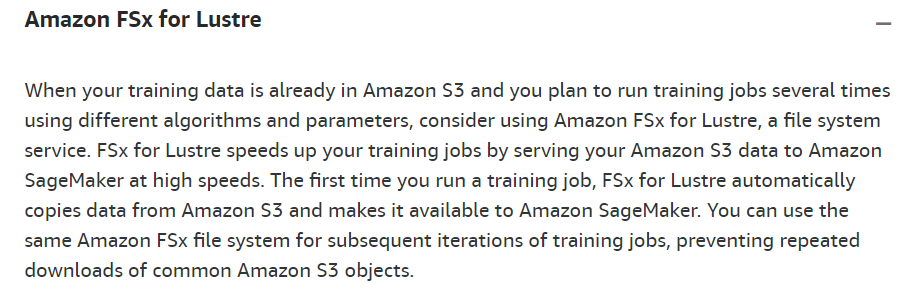
* + 1. AWS Data Lake



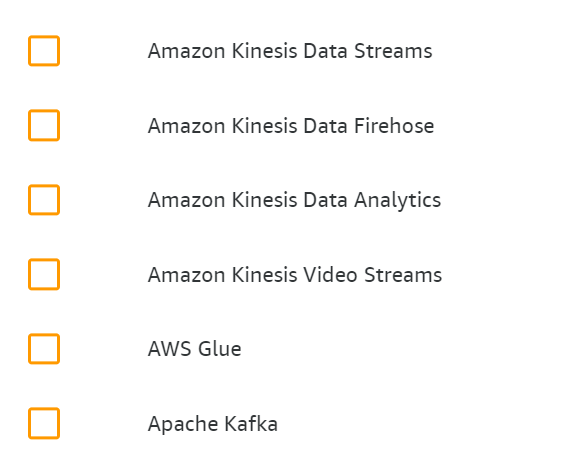
1. OLTP – Online Transaction Processing
2. OLAP – Online Analytics Processing
3. LOB – Line of Business
4. ERP – Enterprise Resource Planning
5. CRM – Customer Relationship Manager

Data Lake: A centralized secure repository that enables you to govern, discover, share, and analyze structured and unstructured data at any scale.

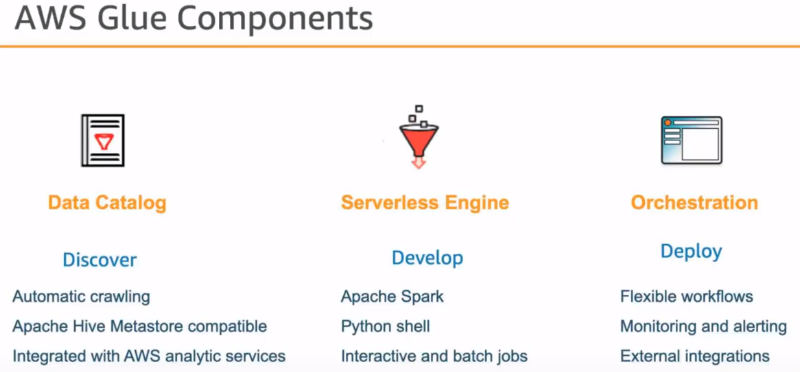
* + 1. Amazon FX for Lustre



* 1. Identify and implement a data ingestion solutions

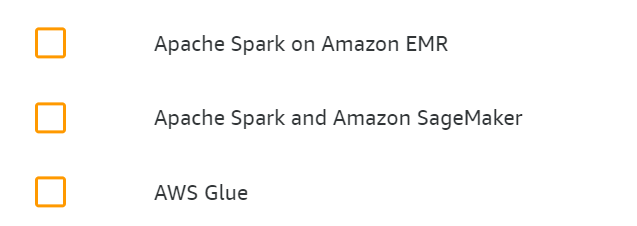


* + 1. Glue



A metadata catalogue, the data catalog stores schemas, columns… Crawl and discover data.

* + 1. Amazon Kinesis Producer Library (KPL) and Amazon Kinesis Client Library (KCL)
  1. Identify and implement a data transformation solution



* + 1. EMR – Elastic MapReduce

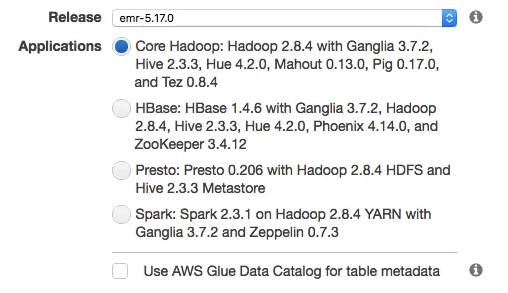
MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

A MapReduce program is composed of a map procedure (or method), which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name), and a reduce method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance

* Hadoop, Ganglia, Hive, Hue, Mahout
* Pig, Tez, HBase
* Phoenix, ZooKeeper, Presto HDFS Metastore, Spark YARN Zeppelin

Frameworks:

Hadoop, Spark, HBase, Flink, Presto

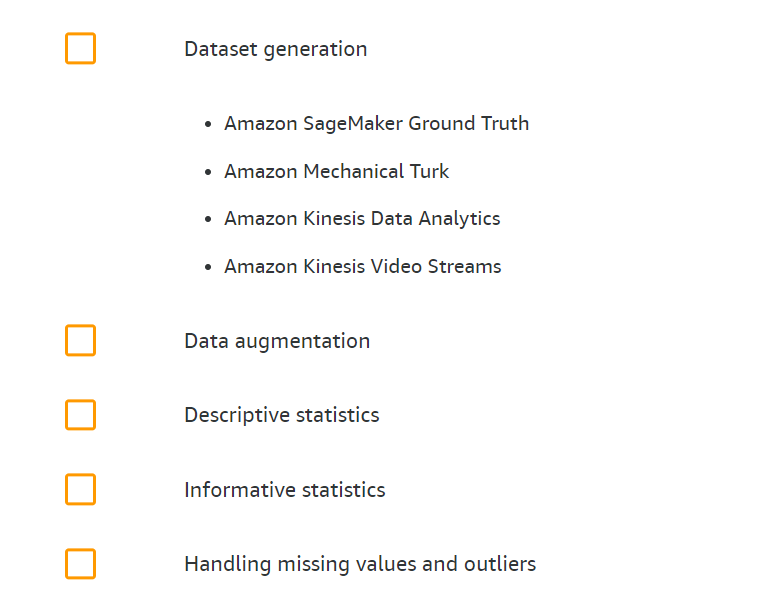


* + 1. Apache Kafka

Originated at LinkedIn. Apache Kafka is publish-subscribe based fault tolerant messaging system. It is fast, scalable and distributed by design.

1. Exploratory Data Analysis
   1. Sanitize and prepare data for modeling

Correlation



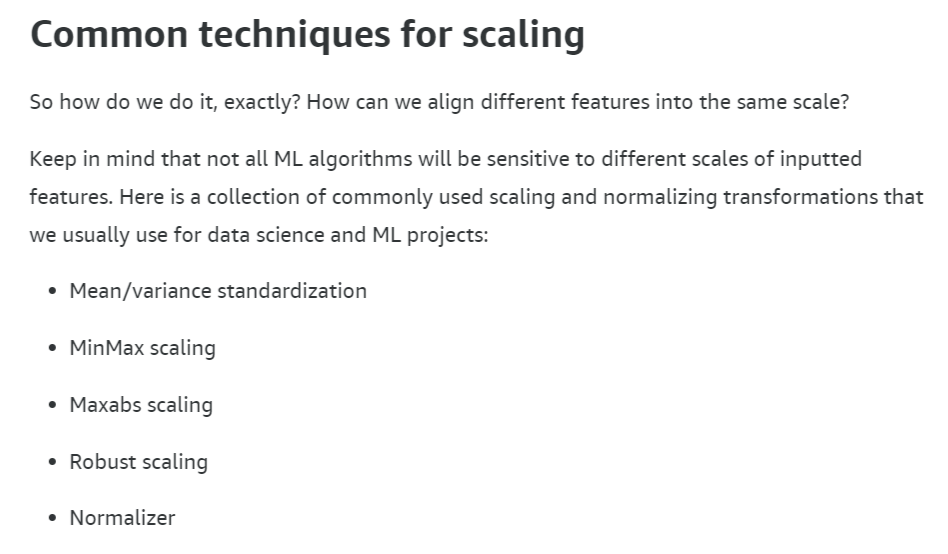
* + 1. Data Augmentation

Data augmentation is an integral process in deep learning, as in deep learning we need large amounts of data and in some cases it is not feasible to collect thousands or millions of images, so data augmentation comes to the rescue.

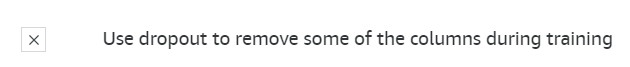
It helps us to increase the size of the dataset and introduce variability in the dataset.

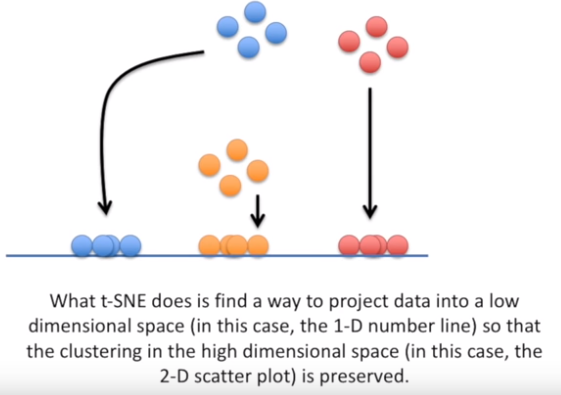
The most commonly used operations are:

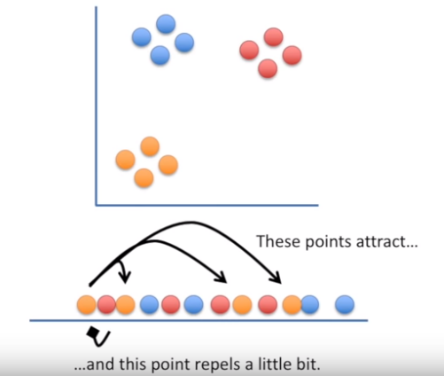
* Rotation
* Shearing
* Zooming
* Cropping
* Flipping
* Changing the brightness level
  1. Perform feature engineering







* 1. T-SNE (t-Distributed Stochastic Neighbor Embedding)

Takes a high dimensional dataset and reduces it to a low dimensional graph while retains the original information.

* + 1. T-SNE vs PCA

PCA and T-SNE:

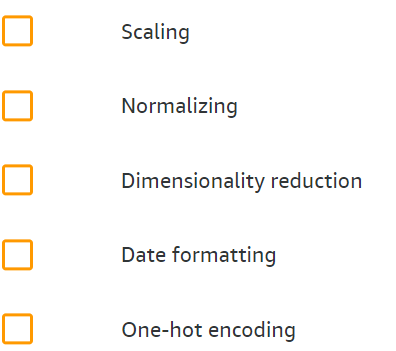
Both principled formulations of dimensionality reduction. Emphasis on principled because even unprincipled formulations like random projection are known to have ϵ bounds on the error under some conditions.

PCA but not T-SNE:

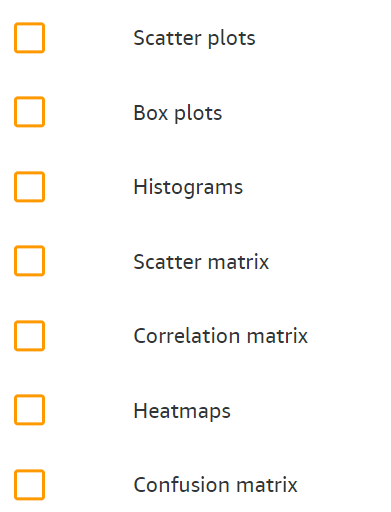
* Can be computed iteratively, so if you’ve already computed k principle components but then you decide you want a (k+1) dimensional representation that is only a little more computation.
* The principle components are an orthogonal basis sorted by amount of variance along the particular dimension. This makes the direction of the principle component vectors tell a story about your original data: which variables account for the most or least variation?
* Once fitted gives you a linear transformation for dimensionality reduction of further points not in the dataset being fitted. The same cannot be said for T-SNE which directly minimizes distance between the dataset and its dimensionality reduction by gradient descent. This gives a correspondence for the known points, but not a function for new points so you’d have to do post-hoc interpolation or start from scratch.

T-SNE but not PCA:

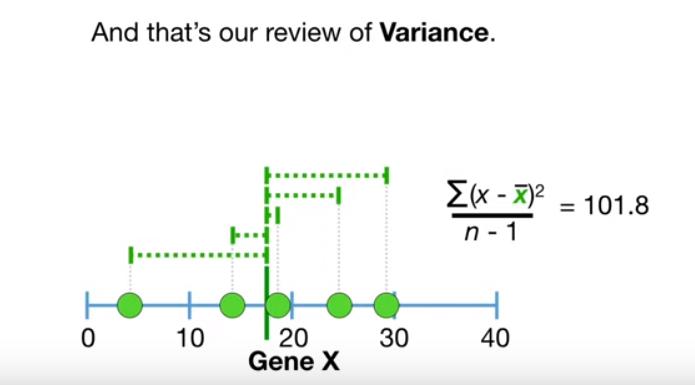
* is non-linear so it can capture the structure of trickier manifolds.
* involves hyperparameters unlike PCA, not that one really needs to worry about it when using reasonable software.

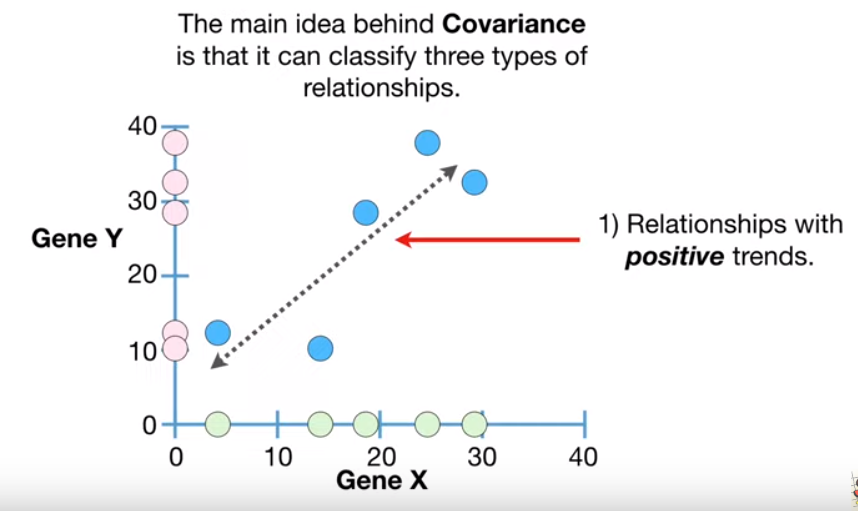


* 1. Analyze and visualize data for ML



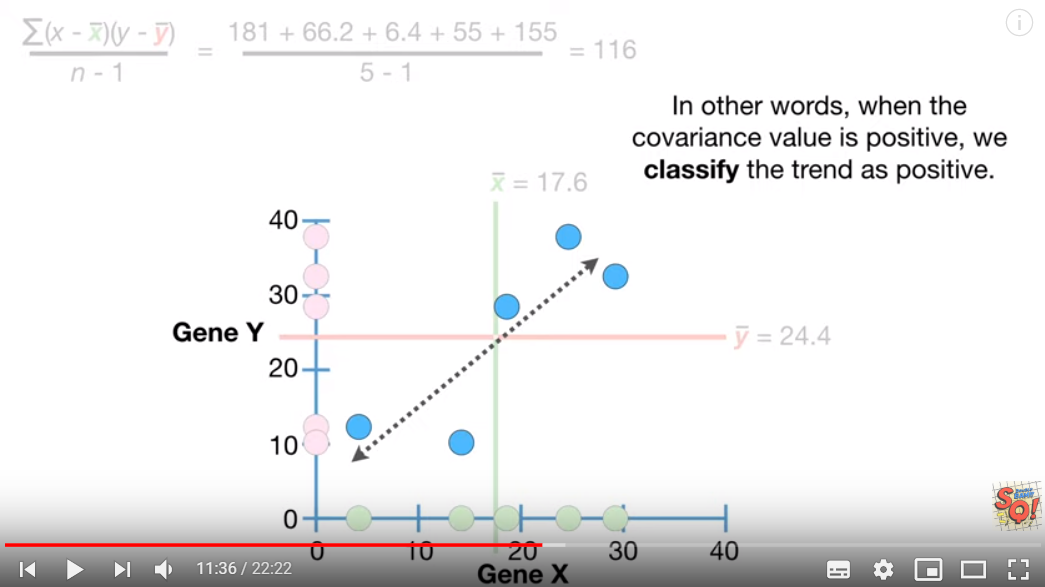
* + 1. Correlation and Covariance

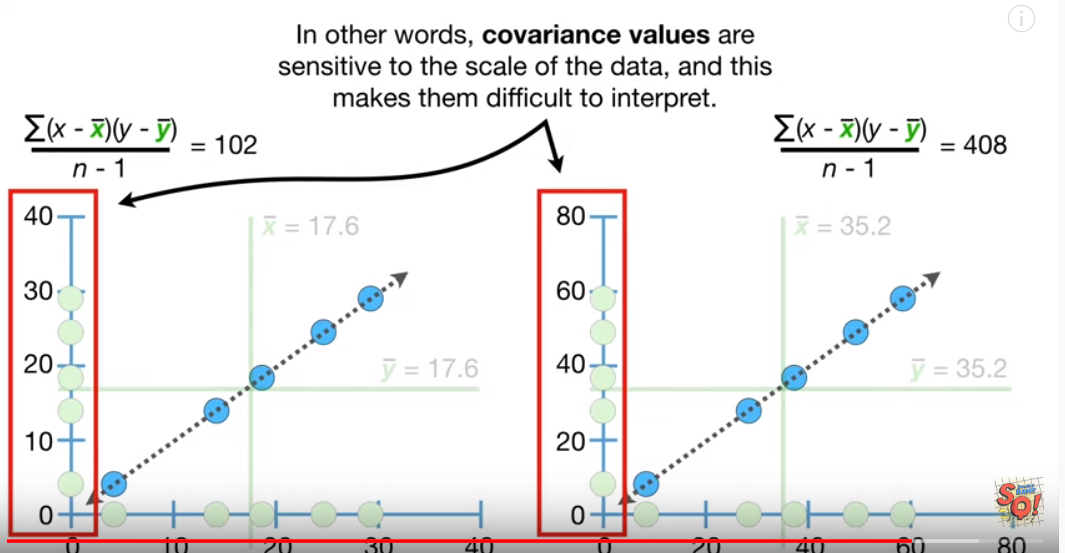




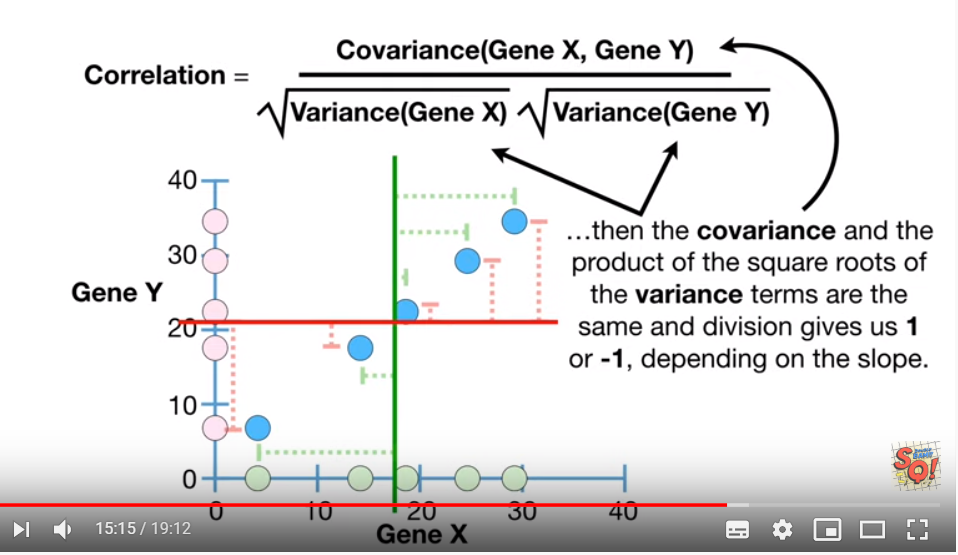
+ Negative + no relationship (e.g. in one line)

Covariance is not really interesting, but it helps you to calculate something really cool: Correlation.

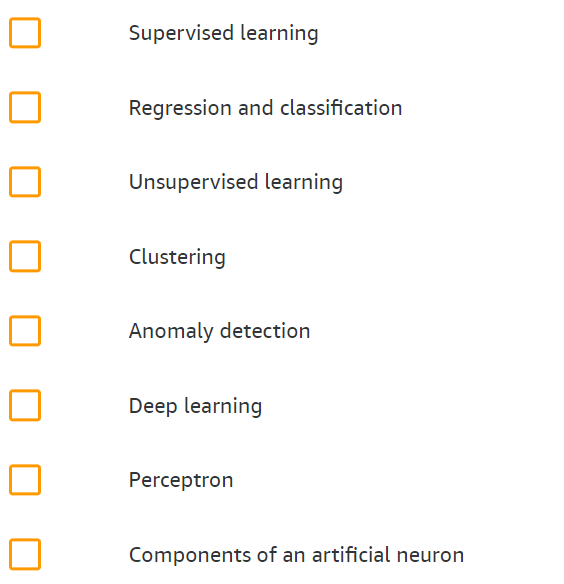


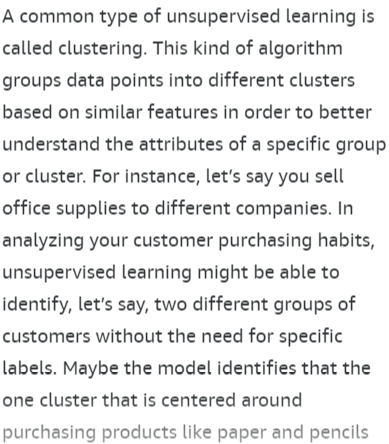


However Correlation describes relationships and it’s not sensitive to the scale of the data. Correlation is used in PCA.

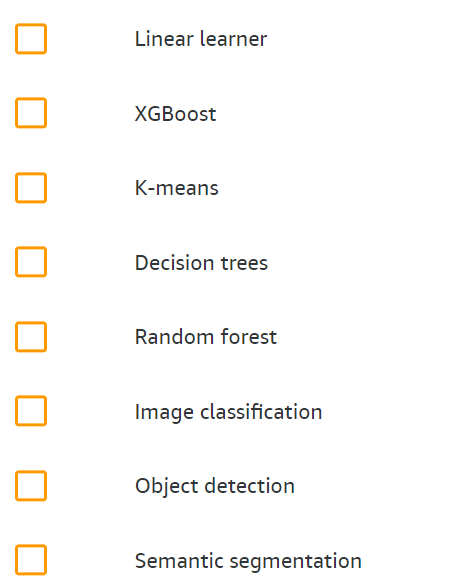


1. Modeling
   1. Frame business problems as ML problems





* 1. Select the appropriate model(s) for an ML problem



* + 1. XGBoost
       1. AdaBoost

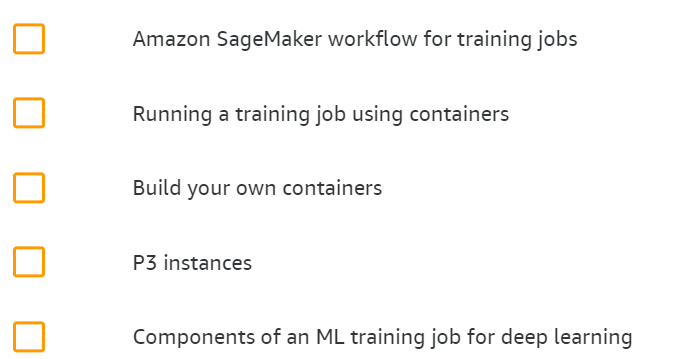
Random Forest with AdaBoost consists of trees only with 1 node and 2 leaves, so called Stumps. AdaBoost combines lot’s of week learners, almost always stumps.

In RF each tree has the same impact on the result, while in AdaBoost not.

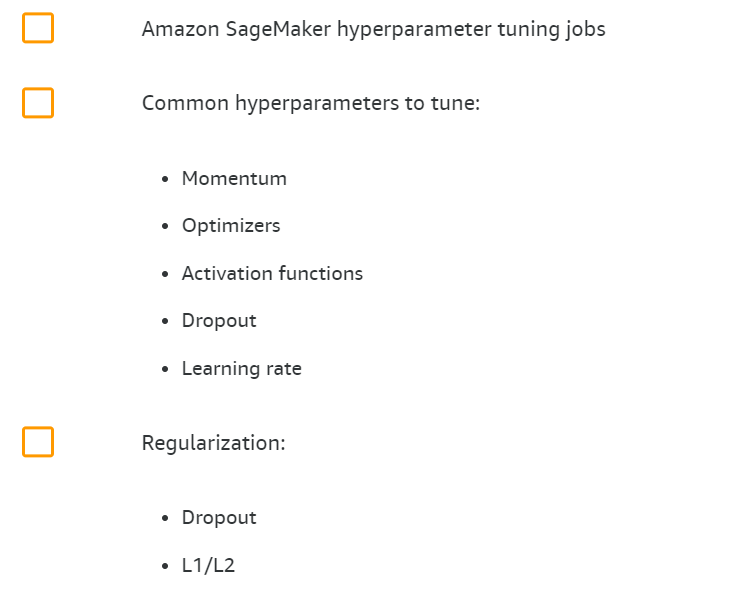
In RF each decision tree is made up independently from the others. So the order does not matter. In AdaBoost it does. The errors that the first stamp make influence how the second stamp is made. And the errors of the 2nd influences how the 3rd is made. Etc…

If we have a Weighted Gini Function, than we use it with the Sample Weights, otherwise we use the Sample Weights to make a new dataset that reflects those weights.

* 1. Train ML models



* 1. Perform hyperparameter optimization



* + 1. Regularization

Regularization basically adds the penalty as model complexity increases.

The key difference between these techniques is that Lasso shrinks the less important feature’s coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

Traditional methods like cross-validation, stepwise regression to handle overfitting and perform feature selection work well with a small set of features but these techniques are a great alternative when we are dealing with a large set of features.

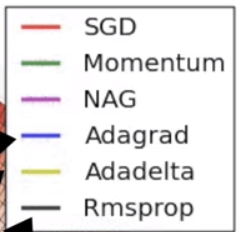
* + - 1. Lasso Regularization – L1
      2. Ridge Regression – L2
      3. Dropout technique

Dropout is a technique where randomly selected neurons are **ignored** during training. They are **“dropped-out” randomly**. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. Dropout is easily implemented by randomly selecting nodes to be dropped-out with **a given probability (e.g. 20%)** each weight update cycle. This is how Dropout is implemented in Keras. Dropout is **only** used **during** the **training** of a model and is not used when evaluating the skill of the model.

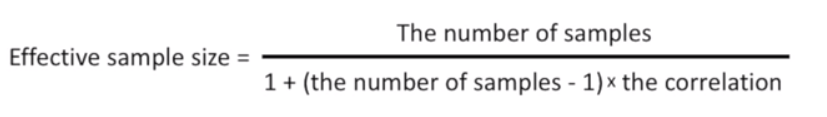
* + 1. Momentum

The basic idea of momentum in ML is to increase the speed of training.

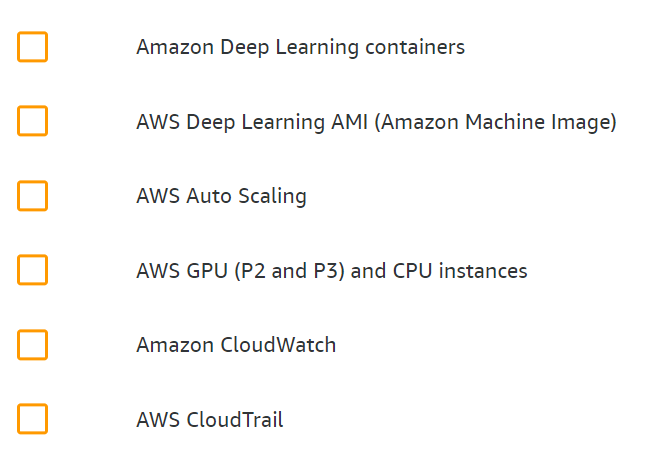
* GD, SGD
* Mini-Batch
* GD Momentum
* Nesterov Accelerated Gradient
* AdaGrad’s Update Rule
* AdaDelta
* **Adam** (Adaptive Moment Estimation)



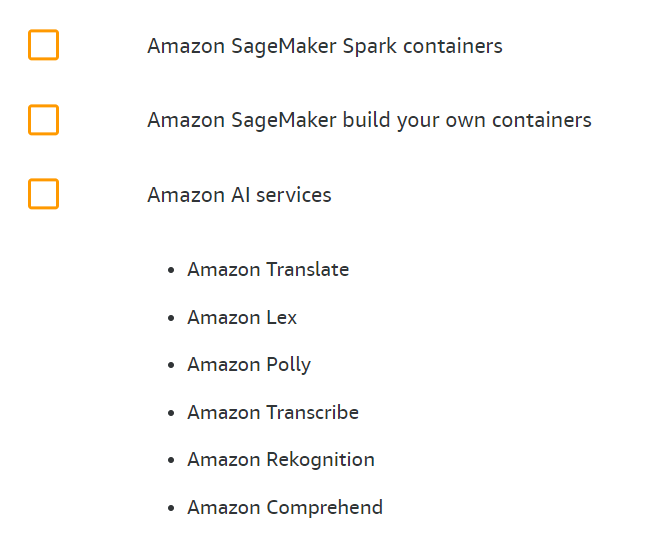
* + 1. AdaGrad
    2. Sample Size and Effective Sample size



1. ML Implementation and Operations
   1. Build ML solutions for performance, availability, scalability, resiliency, and fault tolerance



* 1. Recommend and implement the appropriate ML services and features for a given problem



* 1. Apply Basic AWS security practices to ML solutions



* 1. Deploy and operationalize ML solutions

