AWS Certified Machine Learning Recap

Date: 27/12/2019

Version: 0.2

Author: Stefano Priola

Preparazione:

Data Engineering(20%)

focused are S3, EMR,Kinesis, Kinesis FireHose, Kinesis Analytics ,Athena and Glue

Exploratory Data Analysis(24 %)

understanding box plot, line chart, scatter plot, heat map, word cloud, pie chart, pivot table and many more.

Modeling(36%)

Try to pick which machine learning algorithm to use for particular scenario

Try to know list of all built in algorithms supported by AWS Sage maker as you may be tested to pick which SageMaker algorithm to use.

Try to understand list of hyper parameters that can be optimized for each popular machine and deep learning algorithms.

Get to know about all the popular data preprocessing and feature engineering tasks like normalization, different types of imputation, outlier detection and many more.

Get to know use cases for CNN(Convolutional Neural Networks) and RNN(Recurrent Neural Networks)

Get to know how to optimize deep learning models using number of epochs and number of layers

Get to know how to solve the problem using combination of models like first use kmeans clustering and PCA and then feed the data to classification

Machine Learning Implementation and Operations(20 %)

this section touches many AWS ML services. You should have understanding on AWS ML application services like transcribe, translate, comprehend, polly, lex. If there is one AWS service you should understand end to end it is SageMaker. Get to know the different features of sage maker like how to use notebook instances, how to create training jobs, How to deploy endpoints, High availability of endpoints, Blue Green deployment endpoints , suitable instances for model training and deployment.

Data Engineering

EMR Hadoop-based technologies like Apache Hive

Amazon Redshift **You can also use a columnar MPP data warehouse like Amazon Redshift. These systems make it simple to run complex analytic queries with orders of magnitude faster performance for joins and aggregations performed over large datasets. Amazon Redshift, in particular, leverages high-performance local disks, sophisticated query execution. and join-optimized data formats. Because it is just standard SQL, you can keep using your existing ETL and BI tools. But you do have to load data, and you have to provision clusters against the storage and CPU requirements you need.**

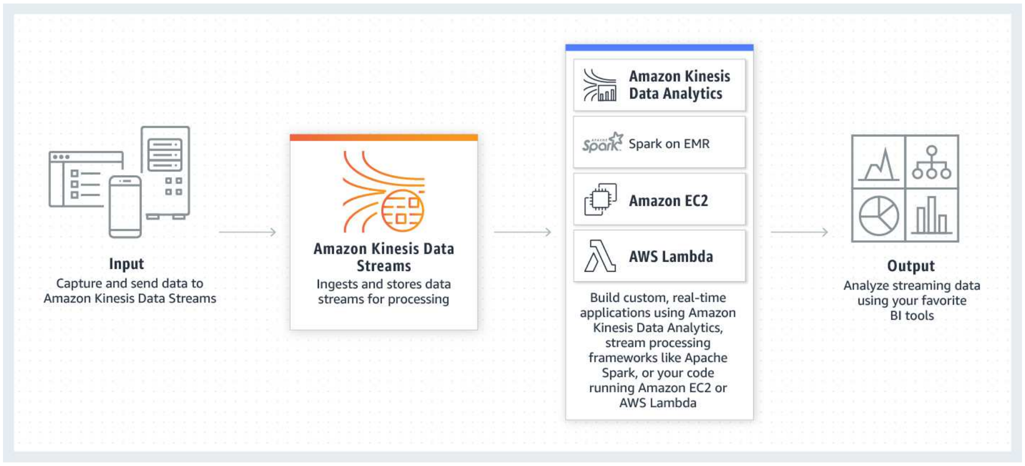
Redshift Spectrum, Amazon Redshift customers can easily query their data in Amazon S3. Spectrum needs a Redshift cluster in place. If you are using Redshift database then it will be wise to use Spectrum along with redshift to get the required performance.

Amazon Athena query **su file di testo su s3** **Amazon Athena è il modo più semplice per consentire ai dipendenti di eseguire query ad hoc sui dati in Amazon S3. Athena è un servizio serverless, Athena Has A REST API**

Amazon Kinesis Data Streams (KDS) INGEST AND STORE DATA is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data collected is available in milliseconds to enable real-time analytics use cases .

Kinesis Streams is capable of capturing large amounts of data (terabytes per hour) from data producers, and streaming it into custom applications for data processing and analysis. Streaming data is replicated by Kinesis across three separate availability zones within AWS to ensure reliability and availability of your data. The more customizable option, Streams is best suited for developers building custom applications or streaming data for specialized needs. The customizability of the approach, however, requires manual scaling and provisioning. Data typically is made available in a stream for 24 hours, but for an additional cost, users can gain data availability for up to seven days.

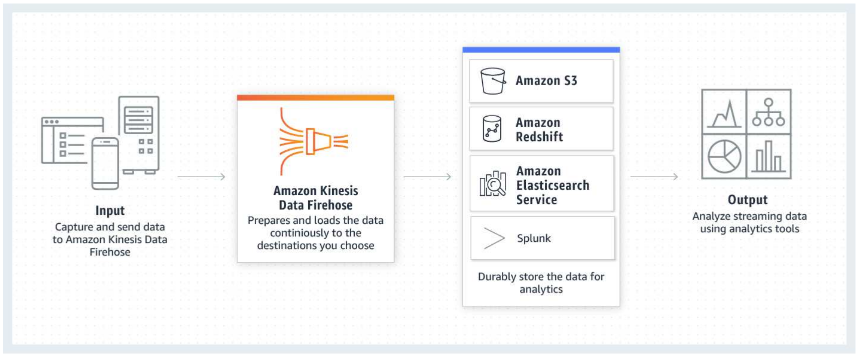
You may use Kinesis Streams if you want to do some custom processing with streaming data. With Kinesis Firehose you are simply ingesting it into S3, Redshift or ElasticSearch.


Input
Capture and send data to Amazon Kinesis Data Streams
Amazon Kinesis Data Streams
Ingests and stores data streams for processing
Amazon Kinesis Data Analytics
Spark on EMR
Amazon EC2
AWS Lambda
Build custom, real-time applications using Amazon Kinesis Data Analytics, stream processing frameworks like Apache Spark, or your code running Amazon EC2 or AWS Lambda
Output
Analyze streaming data using your favorite BI tools

Amazon Kinesis Firehose is a fully managed service for delivering real-time streaming data directly to Amazon S3. Kinesis Firehose encryption supports Amazon S3 server-side encryption with AWS Key Management Service (AWS KMS) for encrypting delivered data in Amazon S3. Finally, Kinesis Firehose can invoke Lambda functions to transform incoming source data and deliver it to Amazon S3. It is used to capture and load streaming data into other Amazon services such as S3, Elasticsearch Service and Redshift (Avro, CSV, Grok, Ion, JSON, ORC, Parquet, RCFile, RegexSerDe, SequenceFile, TextFile e TSV), where data can be copied for processing through additional services. It should be noted that Firehose will automatically scale to meet demand, which is in contrast to Kinesis Streams, for which you must manually provision enough capacity to meet anticipated needs. The simpler approach, Firehose handles loading data streams directly into AWS products for processing.

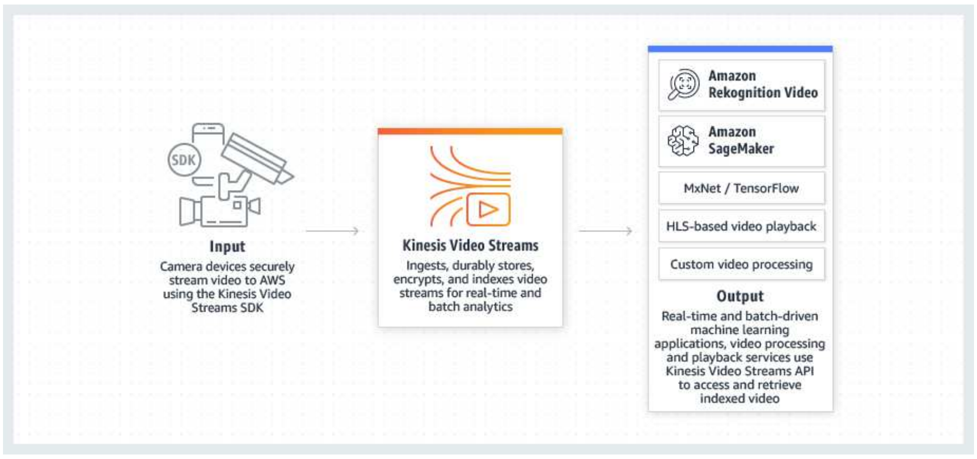
Per aggiungere dati a un flusso di distribuzione Amazon Kinesis Data Firehose, è possibile utilizzare Amazon Kinesis Agent (L'agente monitora determinati file e invia in modo continuo dati nel flusso di distribuzione) oppure i comandi `PutRecord` e `PutRecordBatch` di Firehose.Kinesis Data Firehose si integra inoltre con altre origini dati di AWS quali Kinesis Data Streams, AWS IoT, Amazon CloudWatch Logs e Amazon CloudWatch Events.

You may use Kinesis Streams if you want to do some custom processing with streaming data. With Kinesis Firehose you are simply ingesting it into S3, Redshift or ElasticSearch.


Input
Capture and send data to Amazon Kinesis Data Firehose
Amazon Kinesis Data Firehose
Prepares and loads the data continiously to the destinations you choose
Amazon S3
Amazon Redshift
Amazon Elasticsearch Service
Splunk
Durably store the data for analytics
Output
Analyze streaming data using analytics tools

Kinesis Data Analytics Analisi di flussi di dati con SQL standard (o java flink). Powerful Real-time Processing. No Servers to Manage. three simple steps: setup your streaming data sources, write your queries or streaming applications, and setup your destination for processed data.

Kinesis Video Streams automatically provisions and elastically scales all the infrastructure needed to ingest streaming video data from millions of devices. It also durably stores, encrypts, and indexes video data in your streams, and allows you to access your data through easy-to-use APIs.


Input
Camera devices securely stream video to AWS using the Kinesis Video Streams SDK
SDK
Kinesis Video Streams
Ingests, durably stores, encrypts, and indexes video streams for real-time and batch analytics
Amazon Rekognition Video
Amazon SageMaker
MxNet / TensorFlow
HLS-based video playback
Custom video processing
Output
Real-time and batch-driven machine learning applications, video processing and playback services use Kinesis Video Streams API to access and retrieve indexed video

Modeling(36%)

SAGEMAKER Algoritms

For training and hosting Amazon SageMaker algorithms, we recommend using the following EC2 instance types:

- ml.m4.xlarge, ml.m4.4xlarge, and ml.m4.10xlarge Standard cpu
- ml.c4.xlarge, ml.c4.2xlarge, and ml.c4.8xlarge Compute Optimized (CPU ma high net)
- ml.p2.xlarge, ml.p2.8xlarge, and ml.p2.16xlarge Accelerated Computing (gpu)

PCA

PCA is an unsupervised machine learning (like clustering) algorithm that attempts to reduce the dimensionality (number of features) within a dataset while still retaining as much information as possible. PCA supports both GPU and CPU computation.

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.

Sequence to Sequence

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). SageMaker seq2seq is only supported on GPU instance

BlazingText

The Amazon SageMaker BlazingText supervised algorithm provides highly optimized implementations of the Word2vec and text classification algorithms. The Word2vec algorithm is useful for many downstream natural language processing (NLP) tasks, such as sentiment analysis, named entity recognition, machine translation, etc. Text classification is an important task for applications that perform web searches, information retrieval, ranking, and document classification.

DeepAR Forecasting

The Amazon SageMaker DeepAR forecasting algorithm is a supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNN).

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. such as click prediction and item recommendation

Image Classification Algorithm

The Amazon SageMaker image classification algorithm is a supervised learning algorithm that takes an image as input and classifies it into one of multiple output categories. It uses a CNN 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. Image classification in Amazon SageMaker can be run in two modes: full training and transfer learning. In full training mode, the network is initialized with random weights and trained on user data from scratch. In transfer learning mode, the network is initialized with pre-trained weights and just the top fully connected layer is initialized with random weights. Then, the whole network is fine-tuned with new data. In this mode, training can be achieved even with a smaller dataset. This is because the network is already trained and therefore can be used in cases without sufficient training data.

Amazon SageMaker visual search

You can apply several different algorithms to perform visual search. For this blog post, we'll extract features from images using a Convolutional Neural Net (CNN) model pretrained on the well-known ImageNet dataset of over 11 million images. The idea of using a pretrained CNN to extract features from. The CNN extracts features from images by detecting colors, textures, shapes, etc. With the resulting features defined in a much lower dimensional space than the original images, it becomes easier to make image-to-image comparisons to find the most visually similar matches. Normally a pretrained CNN outputs labels for classifying images into different categories such as car, table, vacuum cleaner, etc. To make use of a pretrained CNN, we'll need to modify it. Specifically, we'll modify it to extract features from images. A CNN typically is modified so the layer before the last fully connected layer is used as the output. The output of this layer can be considered as a "feature vector" for an image that summarizes the image's features. To perform visual search given an image of a query item, a query feature vector is produced and then a lookup is performed against an index of feature vectors of reference items. A key advantage of using a pretrained model is avoiding the time and costs associated with gathering labeled training data and training a new model. The featurization process is much faster than model training.

IP Insights Algorithm

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. 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.

K-Means Algorithm

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.

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.

Latent Dirichlet Allocation (LDA)

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. This allows LDA to discover these word groups and use them to form topics. ou can use LDA for a variety of tasks, from clustering customers based on product purchases to automatic harmonic analysis in music. However, it is most commonly associated with topic modeling in text corpuses. Observations are referred to as documents. The feature set is referred to as vocabulary. A feature is referred to as a word. And the resulting categories are referred to as 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.

Neural Topic Model (NTM)

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. Although you can use both the Amazon SageMaker NTM and LDA algorithms for topic modeling, they are distinct algorithms and can be expected to produce different results on the same input data.

Object2Vec

Object2Vec 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 such a way that the semantics of the relationship between pairs of objects in the original space are preserved in the embedding space. You can use the learned embeddings, for example, to efficiently compute nearest neighbors of objects and to visualize natural clusters of related objects in low-dimensional space. You can also use the embeddings as features of the corresponding objects in downstream supervised tasks, such as classification or regression. Embeddings are an important feature engineering technique in machine learning (ML). They convert high dimensional vectors into low-dimensional space to make it easier to do machine learning with large sparse vector inputs. It has been widely used in many use cases, such as sentiment analysis, document classification, and natural language understanding.

Object Detection Algorithm

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.

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. 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. Although the algorithm could technically run on GPU instance types it does not take advantage of GPU hardware.

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. 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.

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. The Amazon SageMaker semantic segmentation algorithm only supports GPU instances for training, and we recommend using GPU instances with more memory for training with large batch sizes.

XGBoost Algorithm

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 the estimates of a set of simpler, weaker models. This flexibility makes XGBoost a solid choice for problems in regression, classification (binary and multiclass), and ranking. Amazon SageMaker XGBoost currently only trains using CPUs. It is a memory-bound (as opposed to compute-bound) algorithm. So, a general-purpose compute instance (for example, M4) is a better choice than a compute-optimized instance (for example, C4)

Multi-GPU and distributed training with Gluon

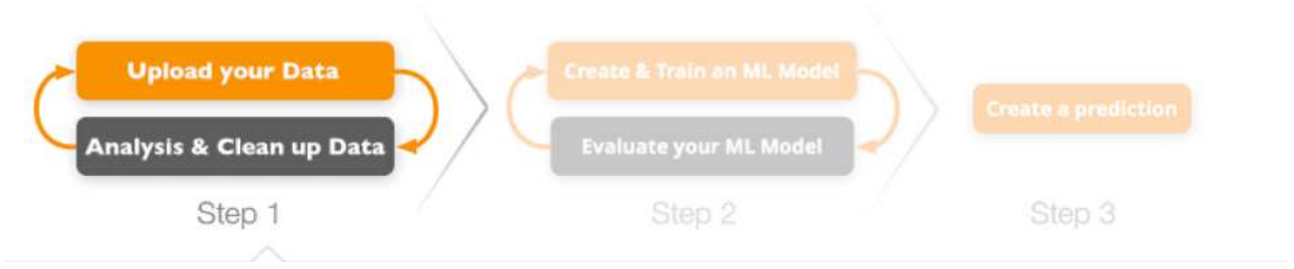
Consequences of increasing mini-batch size

When moving from training on a single GPU to training on multiple GPUs, a good heuristic is to increase the mini-batch size by multiplying by the number of GPUs to keep the mini-batch size per GPU constant. For example, if a mini-batch size of 128 keeps a single GPU fully utilized, you should increase to a mini-batch size of 512 when using four GPUs. To increase the rate of convergence with larger mini-batch size, you must increase the learning rate of the SGD optimizer.

epochs The number of passes done over the training data.

datasource?

A datasource is an Amazon ML object that contains information about your input data, including its location, attribute names and types, and descriptive statistics for each attribute. Note that a datasource does not contain your input data, it only points to its location.



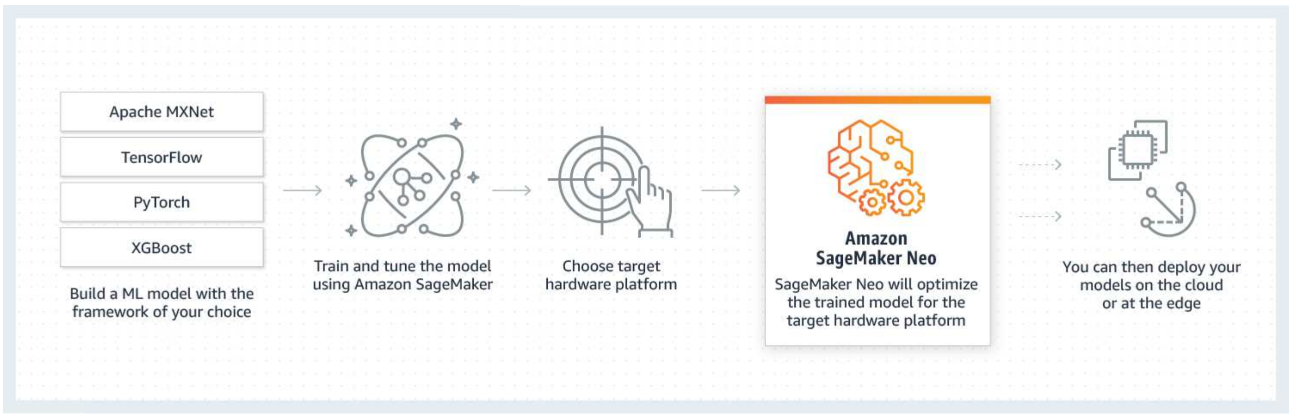
Algoritm

- Supervised
- Unsupervised
- Reinforcement – Agent -> environment with action and reward/sensation

XGBoost is an optimized distributed gradient boosting library designed to be highly ***efficient***, ***flexible*** and ***portable***. It implements machine learning algorithms under the Gradient Boosting

Apache MXNet is a flexible and efficient deep learning platform. It's particularly suitable for multi-GPU and distributed training across multiple hosts. The Gluon library in Apache MXNet provides a clear, concise, and simple API for deep learning. Gluon is an imperative and dynamic Python interface for MXNet, which enables rapid model development, while maintaining MXNet performance

Amazon SageMaker Neo enables developers to train machine learning models once and run them anywhere in the cloud and at the edge.Amazon SageMaker Neo converts the framework-specific functions and operations for TensorFlow, MXNet, and PyTorch into a single compiled executable that can be run anywhere.


Apache MXNet
TensorFlow
PyTorch
XGBoost
Build a ML model with the framework of your choice
Train and tune the model using Amazon SageMaker
Choose target hardware platform
Amazon SageMaker Neo
SageMaker Neo will optimize the trained model for the target hardware platform
You can then deploy your models on the cloud or at the edge

Gradient descent is an optimization algorithm often used for finding the weights or coefficients The goal of the algorithm is to find model parameters (e.g. coefficients or weights) that minimize the error of the model on the training dataset

Amazon Elastic Inference, a new service that lets you attach just the right amount of GPU-powered inference acceleration to any Amazon EC2 Here are the high-level steps required to use the service with an Amazon EC2 instance.

1. Create a security group for the instance allowing only incoming SSH traffic.
2. Create an IAM role for the instance, allowing it to connect to the Amazon Elastic Inference service.
3. Create a VPC endpoint for Amazon Elastic Inference in the VPC where the instance will run, attaching a security group allowing only incoming HTTPS traffic from the instance. Please note that you'll only have to do this once per VPC and that charges for the endpoint are included in the cost of the accelerator.

Machine learning (ML) is split in two distinct phases: training and inference. raining deals with building the model, Inference deals with using the model, i.e. predicting results

Sagemaker

```
from sagemaker import get_execution_role
from sagemaker.session import Session

role = get_execution_role()
bucket = Session().default_bucket()
```

security

create an IAM role that you attach to the EC2 instance to give applications running on the instance temporary security credentials. The credentials have the permissions specified in the policies attached to the role.

The policy {

```
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "AllowCreate-Describe-Delete-Models",
      "Effect": "Allow",
      "Action": [
         "sagemaker:CreateModel",
         "sagemaker:DescribeModel",
         "sagemaker:DeleteModel"],
      "Resource": "*"
      }
   ],
     "Statement": [{
      "Sid": "AdditionalIamPermission",
      "Effect": "Allow",
      "Action": [
         "iam:PassRole"],
      "Resource": "arn:aws:iam::account-id:role/role-name"
      }
   ]
```

}

has two statements:

- The first statement grants permission for three Amazon SageMaker actions (`sagemaker:CreateModel`, `sagemaker:DescribeModel`, and `sagemaker:DeleteModel`) within an Amazon SageMaker notebook instance.

Sagemaker Groud True – make easy label dataset. With Ground Truth, you can use workers from either Amazon Mechanical Turk, a vendor company that you choose, or an internal, private workforce along with machine learning to enable you to create a labeled dataset.

Linear Learner = linear + logistic regression- solving either classification or regression problems

Decision tree – non linear supervised algoritm

XGboost - non linear supervised algoritm

Random forest – ottimo per anomalie

K-means – unsupervised

Anomaly detection

CNN – convolutional – Image Classification

RNN – recurrent output messo come feedback – Sequence to sequence

Time Series - DeepAR

A well-defined machine learning problem, including the following:

- A dataset
- An understanding of the type of algorithm you need to train
- A clear understanding of how you measure success

MONITORING

Amazon CloudWatch monitors your AWS resource

AWS Snowball

to securely and efficiently migrate bulk data from on-premises storage platforms and Hadoop clusters to S3 buckets. . PETABYTE. Snowball appliance will be automatically shipped to you. AES-256-bit encryption.

AWS Storage Gateway

AWS Storage Gateway can be used to integrate legacy on-premises data processing platforms with an Amazon S3-based data lake. he File Gateway configuration of Storage Gateway offers on-premises devices and applications a network file share via an NFS connection. Files written to this mount point are converted to objects stored in Amazon S3 in their original format without any proprietary modification. This means that you can easily integrate applications and platforms that don't have native Amazon S3 capabilities

DistCP, which is a standard Apache Hadoop data transfer mechanism. This allows you to run DistCP jobs to transfer data from an on-premises Hadoop cluster to an S3 bucket.

CRISP DM cross industry standard process data mining. Business understanding, data understanding, data preparation, modeling, data evaluation , deployment

Data Transformations

Recipe Format Reference: When you create a new datasource in Amazon ML and statistics are computed for that datasource, Amazon ML will also create a suggested recipe that can be used to create a new ML model from the datasource. Groups You can define groups of variables in order to collectively transform all variables within the groups, Assignments You can assign one or more transformations to an intermediate variable, for convenience and readability. Outputs The outputs section controls which input variables will be used for the learning process, and which transformations apply to them. The simplest outputs section simply includes the predefined **ALL_INPUTS**

"no_punct(var1)"

"lowercase(var1)"

"osb(var1, 4)" To illustrate, consider the string "The quick brown fox jumps over the lazy dog", and OSBs of size 4.

```
"The quick brown fox", {The_quick, The__brown, The___fox}

"quick brown fox jumps", {quick_brown, quick__fox, quick___jumps}

"brown fox jumps over", {brown_fox, brown__jumps, brown___over}

"fox jumps over the", {fox_jumps, fox__over, fox___the}

"jumps over the lazy", {jumps_over, jumps__the, jumps___lazy}

"over the lazy dog", {over_the, over__lazy, over___dog}

"the lazy dog", {the_lazy, the__dog}

"lazy dog", {lazy_dog}
```

```
"ngram(var1, 3)" {"I really enjoyed", "really enjoyed reading", "enjoyed reading this",
"reading this book", "I really", "really enjoyed", "enjoyed reading",
"reading this", "this book", "I", "really", "enjoyed", "reading",
"this", "book"}
```

"quantile_bin(var1, 50)" The quantile binning processor takes two inputs, a numerical variable and a parameter called *bin number*, and outputs a categorical variable.

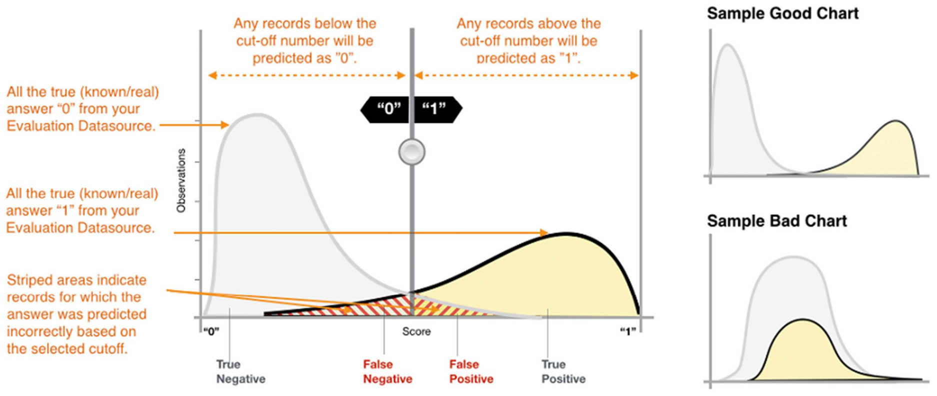
cartesian(var1, var2)

In statistics, a **categorical variable** is a variable that can take on one of a limited, and usually fixed number of possible values, assigning each individual or other unit of observation to a particular group or nominal category on the basis of some qualitative property.[1] In computer science and some branches of mathematics, categorical variables are referred to as enumerations or enumerated types

AWS Glue will generate ETL code in Scala or Python to extract data from the source, transform the data to match the target schema, and load it into the target. You can edit, debug and test this code via the Console, in your favorite IDE, or any notebook.

Model Accuracy EVALUATION

Amazon ML provides an industry-standard accuracy metric for binary classification models called Area Under the (Receiver Operating Characteristic) Curve (AUC). AUC measures the ability of the model to predict a higher score for positive examples as compared to negative examples. The AUC metric returns a decimal value from 0 to 1. AUC values near 1 indicate an ML model that is highly accurate. Values near 0.5 indicate an ML model that is no better than guessing at random.



Any records below the cut-off number will be predicted as "0".
Any records above the cut-off number will be predicted as "1".
All the true (known/real) answer "0" from your Evaluation Datasource.
All the true (known/real) answer "1" from your Evaluation Datasource.
Striped areas indicate records for which the answer was predicted incorrectly based on the selected cutoff.
"0"
"1"
Observations
Score
True Negative
False Negative
False Positive
True Positive
Sample Good Chart
Sample Bad Chart

Correct Predictions

- True positive (TP): Amazon ML predicted the value as 1, and the true value is 1.
- True negative (TN): Amazon ML predicted the value as 0, and the true value is 0.

Erroneous Predictions

- False positive (FP): Amazon ML predicted the value as 1, but the true value is 0.
- False negative (FN): Amazon ML predicted the value as 0, but the true value is 1.

BINARY classification

$$Accuracy = \frac{\mathrm{TP + TN}}{\mathrm{TP + FP + FN + TN}}$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP + FP}}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP + FN}}$$

$$FPR = \frac{\mathrm{FP}}{\mathrm{FP + TN}}$$

MUTICLASS

F1 score is a binary classification metric that considers both binary metrics precision and recall. It is the harmonic mean between precision and recall. The range is 0 to 1. A larger value indicates better predictive accuracy:

$$F1\ score = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

REGRESSION

$$RMSE = \sqrt{1/N \sum_{i=1}^{N} (actual\ target - predicted\ target)^2}$$

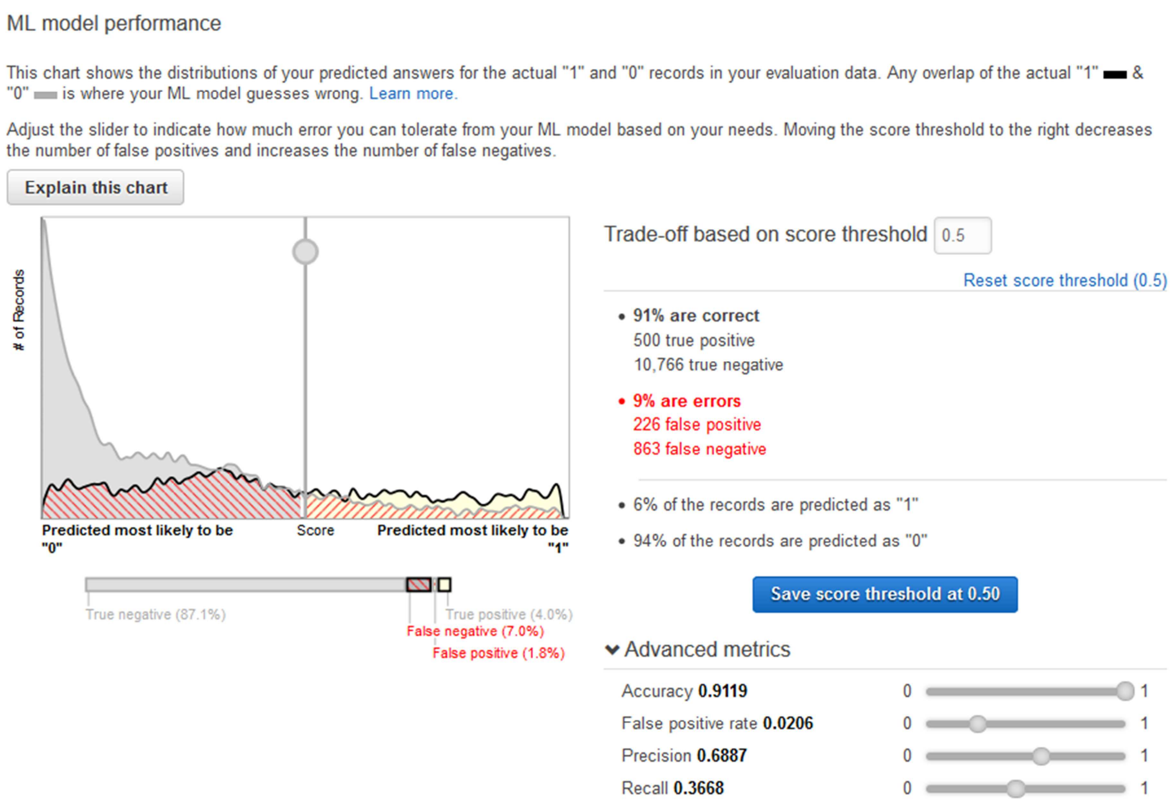
It is a distance measure between the predicted numeric target and the actual numeric answer (ground truth). The smaller the value of the RMSE, the better is the predictive accuracy of the model. A model with perfectly correct predictions would have an RMSE of 0

CROSSVALIDATION

In Amazon ML, you can use the k-fold cross-validation method to perform cross-validation. In k-fold cross-validation, you split the input data into k subsets of data (also known as folds). You train an ML model on all but one (k-1) of the subsets, and then evaluate the model on the subset that was not used for training. T

Adjust Score Threshold (CUTOFF)

You can control the cutoff for what the model considers a positive prediction by increasing the score threshold until it considers only the predictions with the highest likelihood of being true positives as positive. You can also reduce the score threshold until you no longer have any false negatives.



ML model performance

This chart shows the distributions of your predicted answers for the actual "1" and "0" records in your evaluation data. Any overlap of the actual "1" ▬ & "0" ▬ is where your ML model guesses wrong. Learn more.

Adjust the slider to indicate how much error you can tolerate from your ML model based on your needs. Moving the score threshold to the right decreases the number of false positives and increases the number of false negatives.

Trade-off based on score threshold 0.5

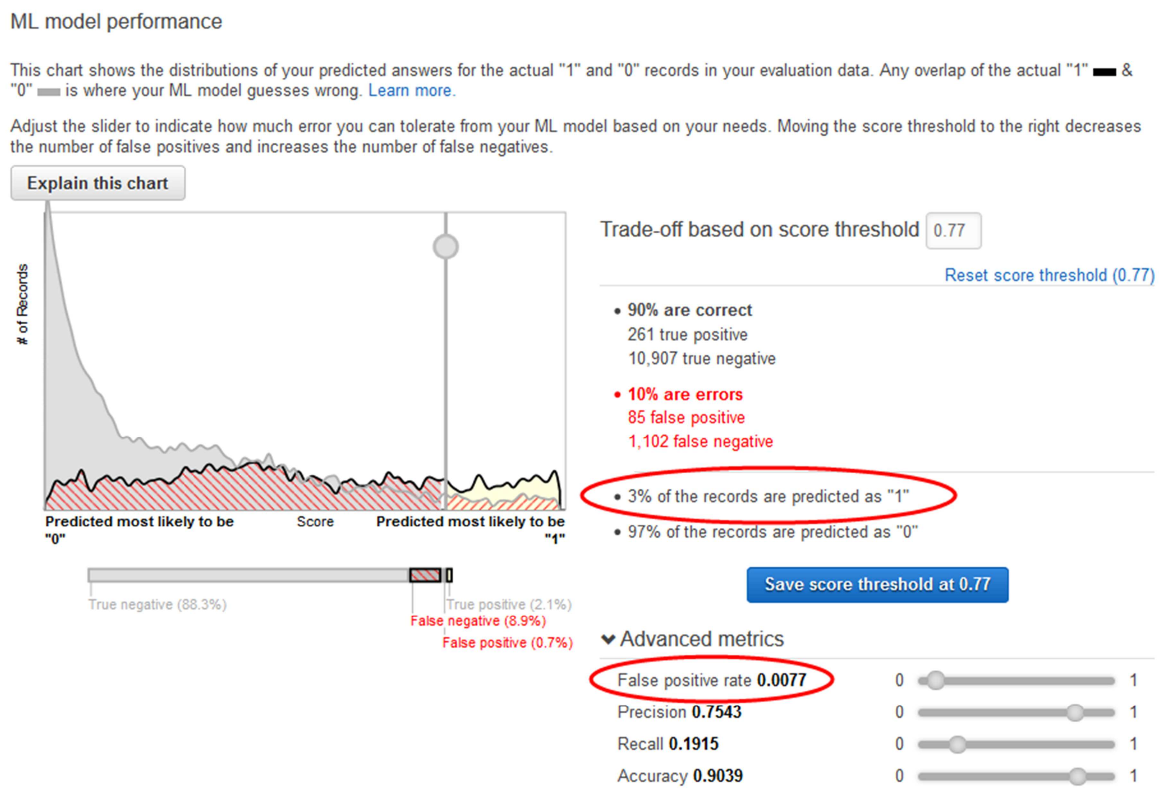
- **91% are correct**
  500 true positive
  10,766 true negative
- **9% are errors**
  226 false positive
  863 false negative

- 6% of the records are predicted as "1"
- 94% of the records are predicted as "0"

Advanced metrics

Accuracy **0.9119**
False positive rate **0.0206**
Precision **0.6887**
Recall **0.3668**

Choose your cutoff to reflect your business needs. For this tutorial, each false positive costs campaign money, so we want a high ratio of true positives to false positives.



ML model performance

This chart shows the distributions of your predicted answers for the actual "1" and "0" records in your evaluation data. Any overlap of the actual "1" ▬ & "0" ▬ is where your ML model guesses wrong. Learn more.

Adjust the slider to indicate how much error you can tolerate from your ML model based on your needs. Moving the score threshold to the right decreases the number of false positives and increases the number of false negatives.

Explain this chart
# of Records
Predicted most likely to be "0"
Score
Predicted most likely to be "1"
True negative (88.3%)
True positive (2.1%)
False negative (8.9%)
False positive (0.7%)
Save score threshold at 0.77
Reset score threshold (0.77)

Trade-off based on score threshold 0.77

- **90% are correct**
  261 true positive
  10,907 true negative
- **10% are errors**
  85 false positive
  1,102 false negative

- 3% of the records are predicted as "1"
- 97% of the records are predicted as "0"

Advanced metrics

False positive rate **0.0077**
Precision **0.7543**
Recall **0.1915**
Accuracy **0.9039**

hyperparameters or training parameters,

Learning Rate Learning rate affects the speed at which the algorithm reaches (converges to) the optimal weights

Model Size you can reduce the model size by using L1 regularization or by specifically restricting the model size by specifying the maximum size.

Number of Passes The Number of passesparameter controls the number of passes that the algorithm makes over the training data.

Data Shuffling huffling mixes up the order of your data so that the SGD algorithm doesn't encounter one type of data for too many observations in succession

Regularization Regularization helps prevent linear models from overfitting training data examples (that is, memorizing patterns instead of generalizing them) by penalizing extreme weight value. L1 regularization has the effect of reducing the number of features used in the model by pushing to zero the weights of features that would otherwise have small weights. As a result, L1 regularization results in sparse models and reduces the amount of noise in the model. L2 regularization results in smaller overall weight values, and stabilizes the weights when there is high correlation between the input features

You can then retrain the model periodically with a larger, improved training dataset.

Use the Amazon SageMaker local mode to train on your notebook instance

using the pre-built TensorFlow and MXNet containers. The Amazon SageMaker deep learning containers allow you to write TensorFlow or MXNet scripts as you typically would. However, now you deploy them to pre-built containers in a managed, production-grade environment for both training and hosting. They've recently been open sourced, which means you can pull the containers into your working environment and use custom code built into the Amazon SageMaker Python SDK to test your algorithm locally, just by changing a single line of code.

SageMaker Hosting Services

Amazon SageMaker also provides model hosting services for model deployment. Amazon SageMaker provides an HTTPS endpoint where your machine learning model is available to provide inferences.

Deploying a model using Amazon SageMaker hosting services is a three-step process:

1. **Create a model in Amazon SageMaker**—By creating a model, you tell Amazon SageMaker where it can find the model components. This includes the S3 path where the model artifacts are stored and the Docker registry path for the image that contains the inference code. In subsequent deployment steps, you specify the model by name
2. **Create an endpoint configuration for an HTTPS endpoint**—You specify the name of one or more models in production variants and the ML compute instances that you want Amazon SageMaker to launch to host them. When hosting models in production, you can configure the endpoint to elastically scale the deployed ML compute instances.
3. **Create an HTTPS endpoint**—Provide the endpoint configuration to Amazon SageMaker. The service launches the ML compute instances and deploys the model or models as specified in the configuration.

You can deploy multiple variants of a model to the same Amazon SageMaker HTTPS endpoint. This is useful for testing variations of a model in production. You want to test a variation of the model by directing a small amount of traffic, say 5%, to the new model. To do this, create an endpoint configuration that describes both variants of the model. You specify the `ProductionVariant` in your request to the `CreateEndPointConfig`.

Typically, a client application sends requests to the Amazon SageMaker HTTPS endpoint to obtain inferences from a deployed model. You can also send requests to this endpoint from your Jupyter notebook during testing.

You can configure a `ProductionVariant` to use Application Auto Scaling

Getting Inferences by Using Amazon SageMaker Batch Transform

Get inferences for an entire dataset by using Amazon SageMaker batch transform. Batch transform uses a trained model to get inferences on a dataset that is stored in Amazon S3, and saves the inferences in an S3 bucket that you specify when you create a batch transform job. Batch transform manages all compute resources necessary to get inferences. This includes launching instances and deleting them after the transform job completes.

Batch transform is ideal for situations where:

- You want to get inferences for an entire dataset and store them online.
- You don't need a persistent endpoint that applications (for example, web or mobile apps) can call to get inferences.
- You don't need the sub-second latency that Amazon SageMaker hosted endpoints provide.
- You want to preprocess your data before using the data to train a new model or generate inferences.

Validating Machine Learning Models

Offline testing—Use historical, not live, data to send requests to the model for inferences. Deploy your trained model to an alpha endpoint, and use historical data to send inference requests to it. To send the requests, use a Jupyter notebook in your Amazon SageMaker notebook instance and either the AWS SDK for Python (Boto) or the high-level Python library provided by Amazon SageMaker.

Online testing with live data—Amazon SageMaker supports deploying multiple models (called production variants) to a single Amazon SageMaker endpoint. You configure the production variants so that a small portion of the live traffic goes to the model that you want to validate. For example, you might choose to send 10% of the traffic to a model variant for evaluation. After you are satisfied with the model's performance, you can route 100% traffic to the updated model.

The Amazon SageMaker Programming Model

Use the Amazon SageMaker console—With the console, you don't write any code. You use the console UI to start model training or deploy a model. The console works well for simple jobs, where you use a built-in training algorithm and you don't need to preprocess training data.

Modify the example Jupyter notebooks—Amazon SageMaker provides several Jupyter notebooks that train and deploy models using specific algorithms and datasets. Start with a notebook that has a suitable algorithm and modify it to accommodate your data source and specific needs.

Write model training and inference code from scratch—Amazon SageMaker provides both an AWS SDK (Boto) and a high-level Python library that you can use in your code to start model training jobs and deploy the resulting models.

Integrate Amazon SageMaker into your Apache Spark workflow—Amazon SageMaker provides a library for calling its APIs from Apache Spark. With it, you can use Amazon SageMaker-based estimators in an Apache Spark pipeline.

Monitoring Amazon SageMaker

Amazon CloudWatch monitors your AWS resources and the applications that you run on AWS in real time.

Amazon CloudWatch monitors your AWS resources and the applications that you run on AWS in real time.

AWS CloudTrail captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred

Glossary

Boto3 Boto is the Amazon Web Services (AWS) SDK for Python, which allows Python developers to write software that makes use of Amazon services like S3 and EC2. Boto provides an easy to use, object-oriented API as well as low-level direct service access.