C1 → Analyze Datasets and Train ML Models using AutoML

Created	@February 16, 2022 2:27 PM
Tag	Code Trainning

⚠️ **First Course of Practical Data Science Specialization**

Index

Reading Material

Specialization Link

https://www.deeplearning.ai/program/practical-data-science-specialization/

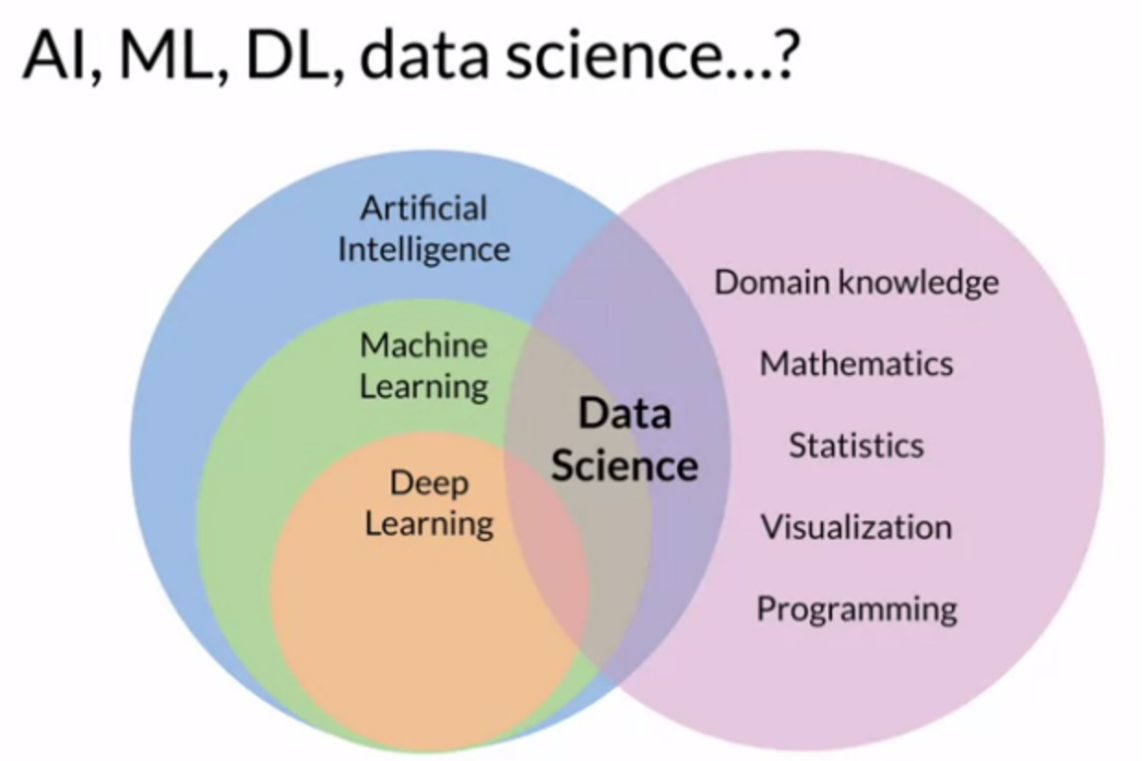
https://www.coursera.org/learn/automl-datasets-ml-models?specialization=practical-data-science

W1 - Explore the Use Case and Analyze the Dataset

▼ Introduction to Practical Data Science

AI ML DL and DS

- **AI** : is generally described as a technique that lets machines mimic human behavior.
- **ML**: s a subset of AI that uses statistical methods and algorithms that are able to learn from data without being explicitly programmed
- **DL:** yet another subset of machine learning, that uses artificial neural networks to learn from data.
- **data Science**: ruly is an interdisciplinary field that combines business and domain knowledge with mathematics, statistics, data visualization, and programming skills



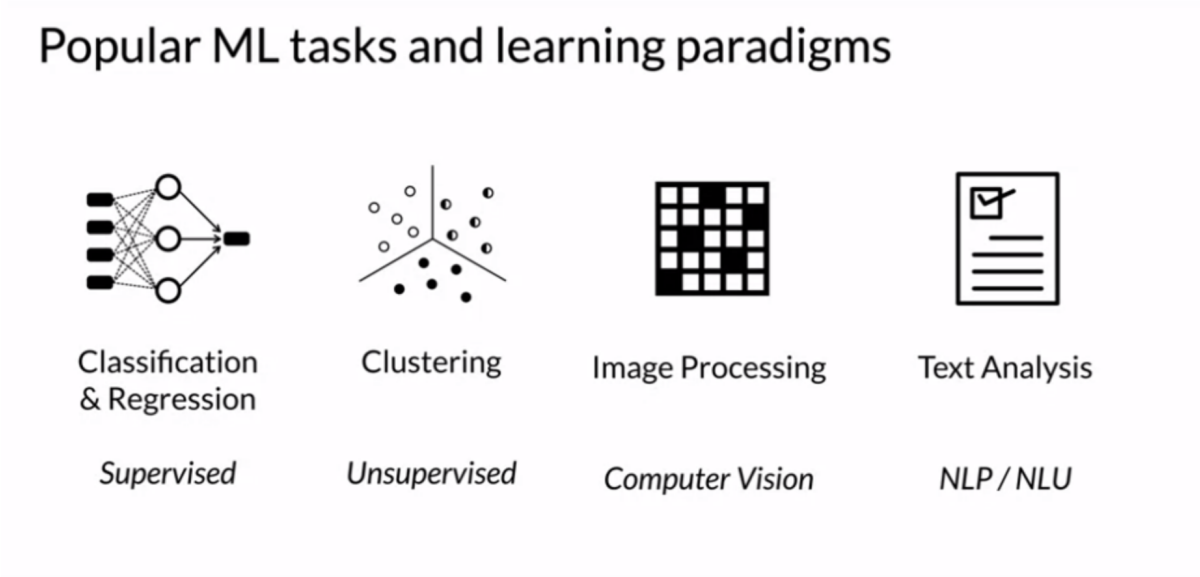
Practical DS: gaining insigth for large datasets, the advantage of working on the cloud is getting bigger resources when needed

Running DS projects in the cloud it allows more scability and freedmom , models turn more scalable. also you can store any amounts of data

Scaling up: bigger CPU

Scaling out: multiple machines simultaneously (computaçaõ distribuida)

▼ Use case and data set



Computer Vision also includes image classification

NLP - Natural Language Processing

NLU - Natural Language Understaing

Task of the course: our task is to build an NLP model that will take those product reviews as input. You will then use the model to classify the sentiment of the reviews into the three classes of positive, neutral and negative. For example, a review such as I simply Love It, should be classified into the positive class. Multi-class classification is a supervised learning task hence you need to provide your tax classifier model with examples how to correctly learn to classify the products and the product reviews into the right sentiment classes.

Input feature for model training	Label for model training
Review Text	Sentiment
I simply love it!	1 (positive)
It's ok.	0 (neutral)
It arrived damaged, going to return	-1 (negative)

▼ Working with Data

Cloud DL allow you to using any kind of data:

- structure data
- semi-structured: jason....
- unstructure: images , audio etc
- streaming data

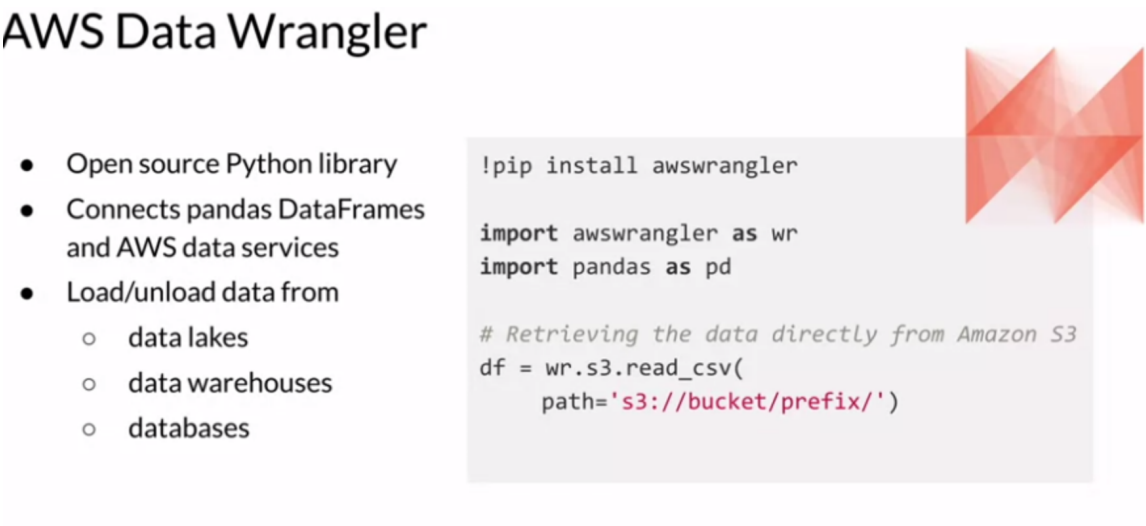
Amazon simple Storage Service - S3:

File storage stores and manages data as individual files organized in hierarchical file folder structures. In contrast, **block storage** stores and manages data as individual chunks called the blocks. Each block receives a unique identifier, but no additional metadata is stored with that block. With object storage, data is stored and managed as objects, which consists of the data itself, any relevant metadata, such

as when the object was last modified and a unique identifier. **Object storage** is particularly helpful for storing and retrieving growing amounts of data of any type hence it's the perfect foundation for data lakes. **Amazon S3 gives you access to durable and high available object storage in the cloud.**

AWS Data Wrangler

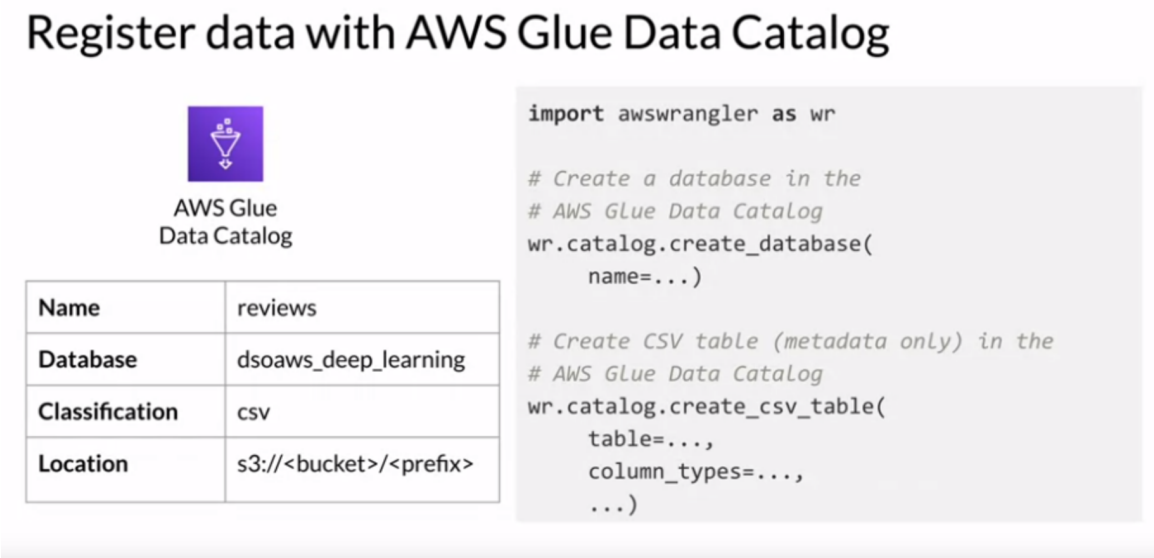
Now, let me introduce some additional toolbox items you will be working on this week. One of them is AWS Data Wrangler. AWS Data Wrangler is an open source Python library developed by members of the AWS professional services team. The library connects Pandas DataFrame with AWS data-related Services. Pandas is a very popular Python data analysis and manipulation tool. AWS Data Wrangler offers abstracted functions to load or unload data from data lakes, data warehouses or databases on AWS. You can install the library through the PIP install AWS wrangler command. Here's a sample code snippet that shows how you can work with AWS Data Wrangler.



AWS Glue Data Catalog

used to catalog the data in S3 data lake/bucket

just used for metadata,

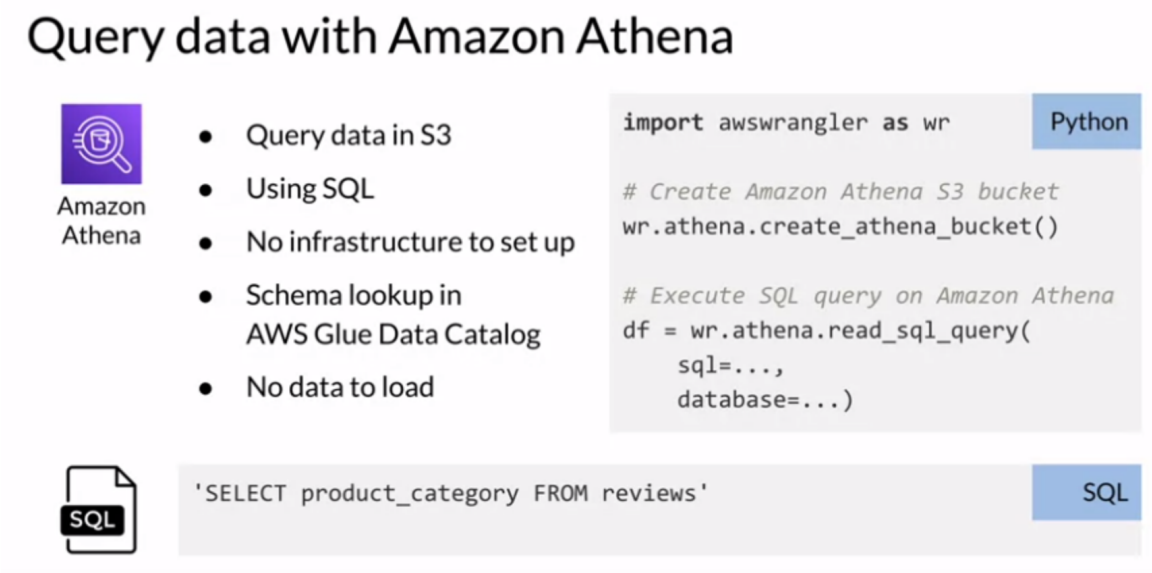


AWS Athena

The data resides in S3, all you store in Wrangler is the metadata ,

From the Python environment you're working in, just use the Data Wrangler, Athena, read_SQL_query function. Pass in the SQL statement you have written and point to the AWS Glue database which contains the table you'll reference here in the SQL query. Again, this database and table only contains the metadata of your data. The data still resides in S3, and when you run this Python command, AWS Data Wrangler will send this SQL query to Amazon Athena. Athena then runs the query on the specified dataset and stores the results in S3, and it also returns the results in a Pandas DataFrame as specified in the command shown here.

athena automatically scales out, using parallel computing for getting the data if needed



Reading Material

AWS Data Wrangler

AWS Glue

Amazon Athena

Matplotlib

Seaborn

Pandas

Numpy

W2 - Data Bias and Feature Importance

Statiscal Bias:

- data is not representative
- imbalaced dataset

Causes:

- Activity bias (social media content): the population in social media does not represent the entire population
- societal bias (human generated bias): Data generated by humans can be biased because all of us have unconscious bias.
- Selection Bias: feedback loop, better recomendations will be more presented and therefore being more viewd and so on
- Data drift: data starts to drift/shift from the training data
    - Covariante drift: the distribution of the independent variables or the features that make up your dataset can change
    - Piror prob drift: data distribution of your labels or the targeted variables might change
    - concept drif: defination on levels might change, the relationship between the two, that is the relationship between the features and the labels can change as well.

Measure bias

- Class imbalance (measures the imbalance in the number of examples that are provided for different facet values in your dataset
- DPL: is actually looking for higher ratings than any other product categories.

▼ SageMaker Data Wrangler:

it allows to generate Bias reports on the datasets, it also peforms this reports on deployed models

It only used a sample of the data to perform the analisys

▼ SageMaker Clarify Processor

is a construct that allows you to scale the bias detection process into a distributed cluster.

```
from sagemaker import clarify

clarify_processor = clarify.SageMakerClarifyProcessor(
    role=role,
    instance_count=1,
    instance_type='ml.c5.2xlarge',
    sagemaker_session=sess)

bias_report_output_path = << Define S3 path >>
```

Distributed cluster size

Type of each instance

S3 location to store bias report

> 💡 **Instant count represents the number of nodes that are included in the cluster, and instance type represents the processing capacity of each individual node in the cluster.**

You need to specify the 3 objects before running the pre trained model to detect the bias. You can select **different measures:** class imbalance or DPL

SageMaker Clarify is using a construct called **SageMaker Processing Job** to execute the bias detection at scale (scale-out). SageMaker Processing Jobs is a construct that allows you to perform any data-related tasks at scale

the SageMaker Processing Job expects the data to be in an S3 bucket.

the report generated is stored in the S3 bucket

SM Data Wrangler vs SM Clarify

You learned about two tools to detect statistical bias in your data sets. SageMaker Data Wrangler and SageMaker Clarify. Now the question becomes, which one of these tools should you use, in which situation? The first option, Data Wrangler, provides you with more of a UI based visual experience. So, if you would like to connect to multiple data sources and explore your data in more visual format and configure what goes into your bias reports by making selections from drop down boxes and option buttons. And finally, launch the bias detection job using a button click, Data Wrangler is the tool for you. Keep in mind that Data Wrangler is only using a subset of your data to detect bias in that data set. On the other hand, SageMaker Clarify provides you with more of an API based approach. Additionally, Clarify also provides you with the ability to scale out the bias detection process. SageMaker Clarify uses a construct called processing jobs that allow you to configure a distributed cluster to execute your bias detection job at scale. So, if you're thinking of large volumes of data, for example, millions of millions of rows of product reviews and you want to explore that data set for bias. Then, SageMaker Clarify is the tool for you, so that you can take advantage of the scale and capacity offered by Cloud.

▼ SHAP → shapley additive explanations

Feature importance

based on shapey values theory

Reading Material

Measure Pretraining Bias - Amazon SageMaker

SHAP

https://aws.amazon.com/sagemaker/pricing/

W3 - Automated Machine Learning (AutoML)

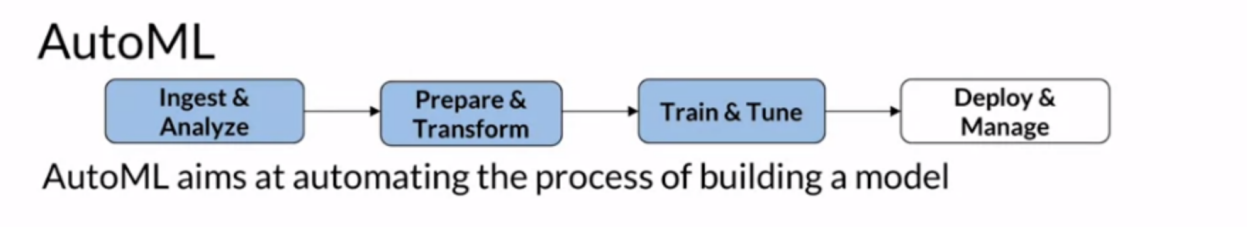
Tool: SagemakerAutopilot

AutoML → allows to reduce time, iterate quickly, allows to tune models in parrallell when used with could computing and

AutoML Flow

Basic steps of Data preparation flow: ingest, feature enngeneering, train-validation split etc etc etc

Auto ML uses Ml to automate some ML steps:



autoML it receives data and does analysis to understand which Ml problem it's facing: classification, multi-classification or regression. Experiments different algorithms,

transforms data /data engineer, train with different hyper-parameters

As a reminder, **AutoML** is a capability that applies machine learning and Automation to model-building task by automatically analyzing your data, automatically selecting the right algorithm or algorithms for your experiments, automatically preparing and applying data transformations, and then finally, automatically training and tuning your model through a number of iterations, until the most performant model is identified

Sagemaker Autopilot

auto pilot generates all the code in jupyter notebooks that allows you to reproduce all the steps, is fully transparent

when creating autopilot experiment you must say which variable is the target, it works for regression, binary classification and multi class

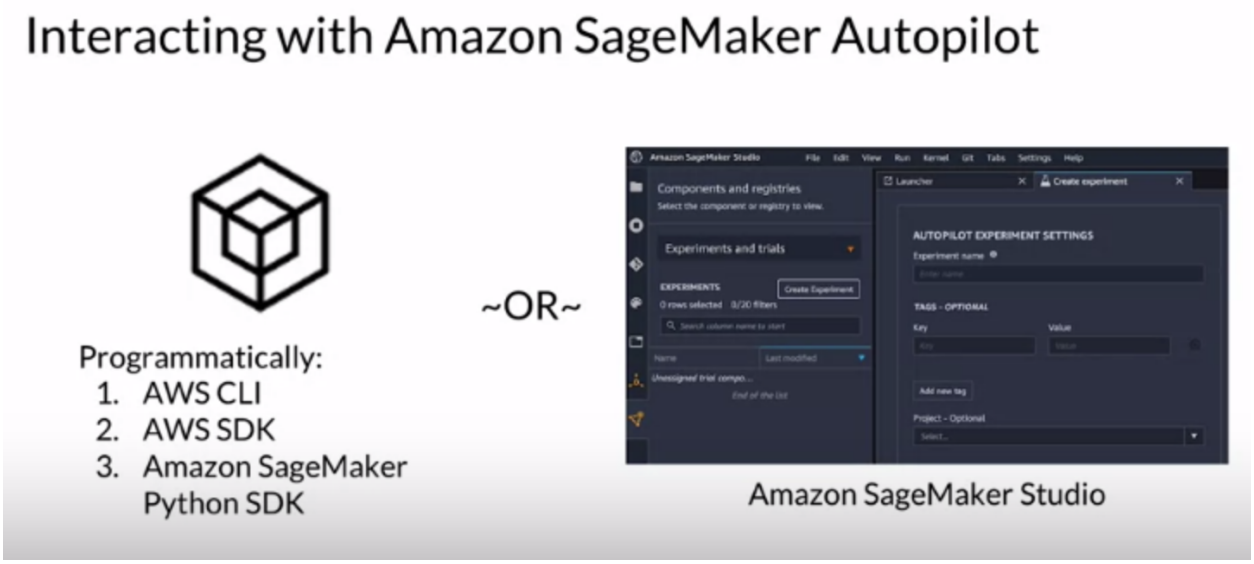
it creates two types of notebooks:

- data exploration notebook, identifies potential issues in the data
- candidate generated notebook: contains each suggested data pre-processing step. The algorithm and the hyper-parameter ranges that will be used for the tuning job

It allows you to choose which metric to use to evaluate the model and allows to tune how much of “auto” it is, you can select some steps to perform manually.

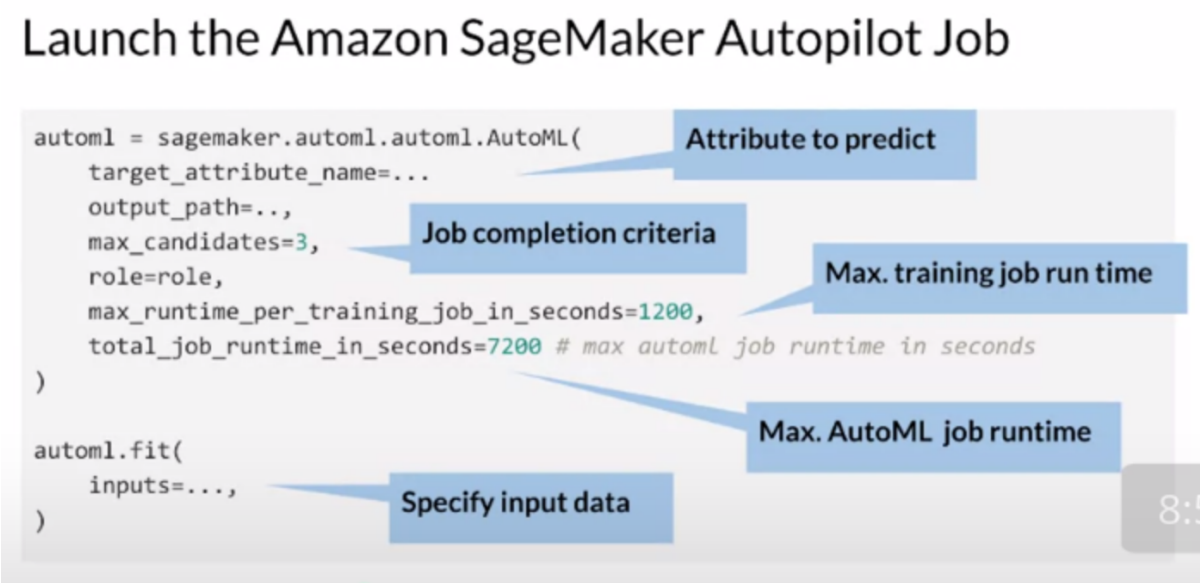
experience = job

Interact with AutoPilot:



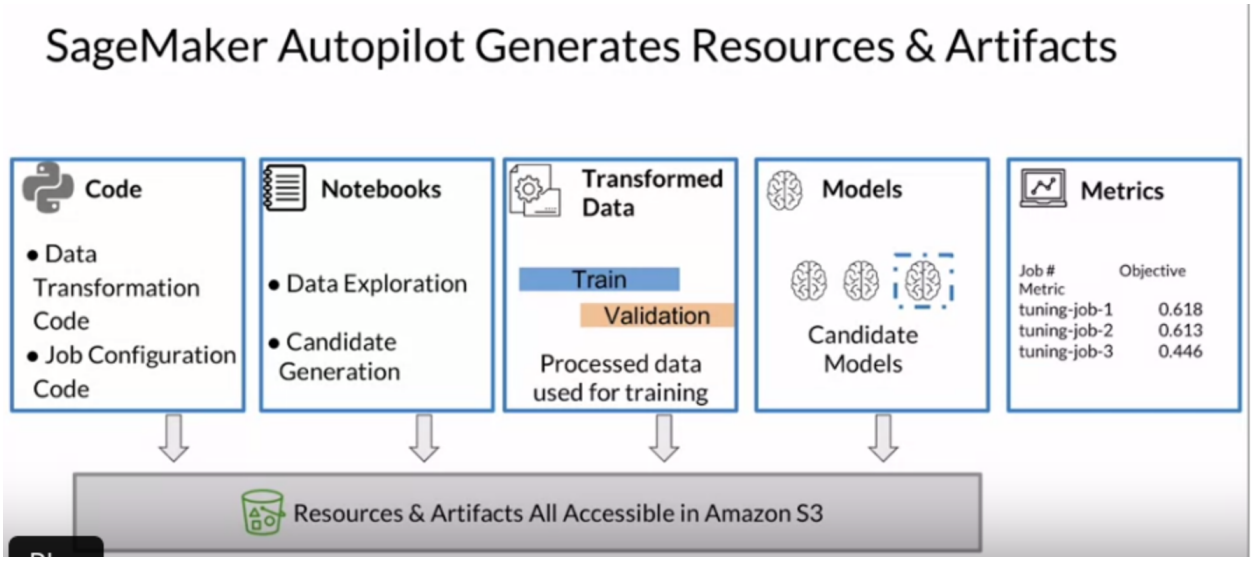
Programatically:

Among other specifications that you can configure:



SM Autopilot outputs:

are stored in S3 bucket specified in the config , also accessible through the SM studio



Model Deploy

Sagemaker are supports both **batch** and **real time deployments**. In this case, you need the model to be persistently available to be able to serve real time request for prediction. Serving your predictions in real time requires a model serving stack that not only has your trained model but also a hosting stack to be able to serve those predictions. In this typically involves some type of a proxy, a web server that can interact with your loaded serving code and you're trained model. Your model can then be consumed by client applications through real time, invoke employment API requests with Sagemaker model hosting, you simply choose the instance type as well as the count combined with the doctor container image that you want to use for inference and then Sagemaker takes care of creating the endpoint. In deploying that model to the endpoint, you can also configure automatic scaling to scale your endpoint to meet the demands of your workload by taking advantage of on demand capacity when it's needed.

When you choose to deploy a candidate pipeline generated by autopilot, it gets deployed using a Sagemaker hosting feature called inference pipeline.

Deploy inference pipeline contains 3 containers:

- **Data Transformation Container:** his container will perform the same transformations on your data set that were used for training so that you can ensure your prediction request data is in the correct format for inference

- **Algorithm Container**: his is the container that contains the train model artifact that was selected as the best performing model based on your hyper parameter tuning jobs
- **Inverse Label Container**: used to post process your prediction into a readable value by your application that consumes the output

With inference pipeline, you're able to host your data transformation model, your product classification model and your inverse label transformer behind the same endpoint. This allows you to keep your training and inference code in sync and allows you to abstract those transformations away from your consuming applications. When an inference request comes in, the request is sent to the first data transformation model and then the remaining models are sequentially run with that final model. In this case, the inverse label transformer sending the final influence result back to your client application. In this section, I briefly covered model hosting on Sagemaker, specifically focusing on real time persistent endpoints and the ability for you to deploy the candidate pipeline model generated by autopilot with a simple configuration. This allows you to host your model using Sagemaker managed endpoints, managed endpoints mean you don't have to manage the underlying infrastructure that's hosting your model and you can focus on machine learning.

Reading material:

Amazon SageMaker Autopilot

W4 - Built-in Algorithms

Built-in algorithms: built-in algorithms to train and deploy your models help you to run experiments quickly because you don't need to write any custom model code

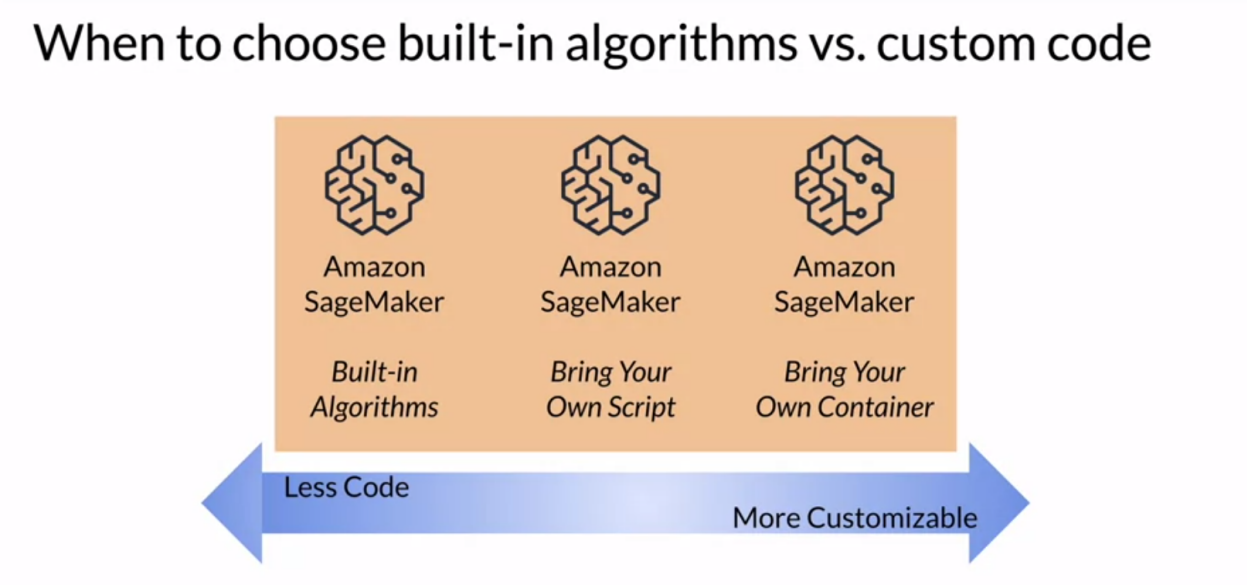
Advantages:

- really scalable, you can swich fom CPU to GPU by simpling change the instance type and also allow parallell computing just by increasing the instance count
- allows to save time (easier to implement)

- allows to download train models and use them elsewhere

When to use:

- **Built in Algorithms:** if the bult-in can solve the task why not use it? keep it simple
- **Bring your own code:** make some changes to built in algorithms
- **Bring your own Container**: writing algorithms in docker containers.



Examples of built-in algorithms for each problem:

Classification & regression

Example problems and use cases	Problem types	Input format	Built-in algorithms
Predict if an item belongs to a category: an email spam filter	Binary/multi-class classification	Tabular	XGBoost, K-Nearest Neighbors (k-NN)
Predict a numeric/continuous value: estimate the value of a house	Regression	Tabular	Linear Learner, XGBoost
Predict sales on a new product based on previous sales data	Time-series forecasting	Tabular	DeepAR Forecasting

Linear Learner: extends up on typical linear models by actually training many models in parallel, each with slightly different type of parameters and then returns the one with the best fit

DeepAR Forecasting: is a supervised learning algorithm for forecasting scalar, meaning one-dimensional time series, using recurrent neural networks or RNNs

Clustering

Example problems and use cases	Problem types	Input format	Built-in algorithms
Drop weak features such as the color of a car when predicting its mileage.	Feature engineering: reduce dimensions	Tabular	Principal Component Analysis (PCA)
Detect abnormal behavior	Anomaly detection	Tabular	Random Cut Forest (RCF)
Group high/medium/low-spending customers from transaction histories	Clustering / grouping	Tabular	K-Means
Organize a set of documents into topics based on words and phrases	Topic modeling	Text	Latent Dirichlet Allocation (LDA), Neural Topic Model (NTM)

RCF: is an unsupervised algorithm for detecting anomalous data points within a dataset. RCF associates an anomaly score with each data point. Low score values indicate that the data point is considered normal. However, high values indicate the presence often anomaly in the data

LDA: is a generative probability model, which means it attempts to provide a model for the distribution of outputs and inputs based on latent variables. In statistics, latent variables are variables that are not directly observed, but are inferred from other variables in the training dataset. This is opposed to the discriminative models, which attempt to learn how inputs map to the outputs.

NTM: uses a deep learning model rather than a pure statistical model

Image processing

Example problems and use cases	Problem types	Input format	Built-in algorithms
Content moderation	Image classification	Image	Image Classification
Detect people and objects in an image	Object detection	Image	Object Detection
Self-driving cars identify objects in their path	Computer vision	Image	Semantic Segmentation

image classification: algorithm can be run in two modes; full training or transfer learning. In full training, the network is initialized with random weights and trained on user data from scratch. In transfer learning mode, network is initialized with pre-trained weights, and just the top fully connected layer is initialized with the 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

object detection algorithm detects all instances of predefined objects within the images categorized as the object, and also adds a bounding box indicating the location and scale of the object in given images.

Semantic segmentation is different from the image classification and object detection in that it classifies every pixel in an image. This leads to information about the shapes of the objects contained in the image. The segmentation output is represented as a grayscale image called a segmentation mask that has the same shape as the input image. Classifying each pixel is fundamental for understanding scenes, which is critical to an increasing number of Compute Edition applications, such as self-driving vehicles, but also medical imaging diagnostics, and robot sensing

Text analysis

Example problems and use cases	Problem types	Input format	Built-in algorithms
Convert Spanish to English	Machine translation	Text	Sequence-to-Sequence
Summarize a research paper	Text summarization	Text	Sequence-to-Sequence
Transcribe call center conversations	Speech-to-text	Text	Sequence-to-Sequence
Classify reviews into categories	Text classification	Text	BlazingText

sequence to sequence algorithm is a supervised learning algorithm where the input is a sequence of tokens, for example, text or audio, and the output is generated as another sequence of tokens

blazing text: optimized implementations of the Word2Vec and text classification algorithms; Bazing text creates character and graham and embeddings using the continuous bag of words and skip gram training architectures, blazing text also allows you to save money by stopping your model training early.

Reading Material

Word2Vec algorithm

GloVe algorithm

FastText algorithm

Transformer architecture, "Attention Is All You Need"

BlazingText algorithm

ELMo algorithm

GPT model architecture

BERT model architecture

Built-in algorithms

Amazon SageMaker BlazingText