Practical Data Science Coursera Specialization

by DeepLearning.AI

Course #1 Analyze Datasets and Train ML Models using AutoML

Course Site

Made By: Matias Borghi

Table of Contents

Summary

In the first course of the Practical Data Science Specialization, you will learn foundational concepts for exploratory data analysis (EDA), automated machine learning (AutoML), and text classification algorithms. With Amazon SageMaker Clarify and Amazon SageMaker Data Wrangler, you will analyze a dataset for statistical bias, transform the dataset into machine-readable features, and select the most important features to train a multi-class text classifier. You will then perform automated machine learning (AutoML) to automatically train, tune, and deploy the best text-classification algorithm for the given dataset using Amazon SageMaker Autopilot. Next, you will work with Amazon SageMaker BlazingText, a highly optimized and scalable implementation of the popular FastText algorithm, to train a text classifier with very little code.

Practical data science is geared towards handling massive datasets that do not fit in your local hardware and could originate from multiple sources. One of the biggest benefits of developing and running data science projects in the cloud is the agility and elasticity that the cloud offers to scale up and out at a minimum cost.

The Practical Data Science Specialization helps you develop the practical skills to effectively deploy your data science projects and overcome challenges at each step of the ML workflow using Amazon SageMaker. This Specialization is designed for data-focused developers, scientists, and analysts familiar with the Python and SQL programming languages and want to learn how to build, train, and deploy scalable, end-to-end ML pipelines - both automated and human-in-the-loop - in the AWS cloud.

Week 1: Explore the Use Case and Analyze the Dataset

Ingest, explore, and visualize a product review data set for multi-class text classification.

Learning Objectives

- **Describe the discipline of practical data science in the cloud**
- **Define the task and use case: Multi-class classification for sentiment analysis of product reviews**
- **Ingest and explore data**
- **Analyze data using visualizations**

Week 2: Data Bias and Feature Importance

Determine the most important features in a data set and detect statistical biases.

Learning Objectives

- **Describe the concept of data bias and compare popular bias metrics**
- **Demonstrate how to detect data bias**
- **Understand feature importance**

Week 3: Use Automated Machine Learning to train a Text Classifier

Inspect and compare models generated with automated machine learning (AutoML).

Learning Objectives

- **Describe the concept of Automated Machine Learning (AutoML)**
- **Discuss how SageMaker Autopilot uniquely implements AutoML**
- **Demonstrate how to train a text classifier with AutoML**

Week 4: Built-in algorithms

Train a text classifier with BlazingText and deploy the classifier as a real-time inference endpoint to serve predictions.

Learning Objectives

- **Summarize why and when to choose built-in algorithms**
- **Describe use case and algorithms**
- **Understand the evolution of text analysis algorithms**
- **Discuss word2vec, FastText and BlazingText algorithms**
- **Transform raw review data into features to train a text classifier**
- **Apply the Amazon SageMaker built-in BlazingText algorithm to train a text classifier**
- **Deploy the text classifier and make predictions**

Week 1: Explore the Use Case and Analyze the Dataset

Introduction to Practical Data Science

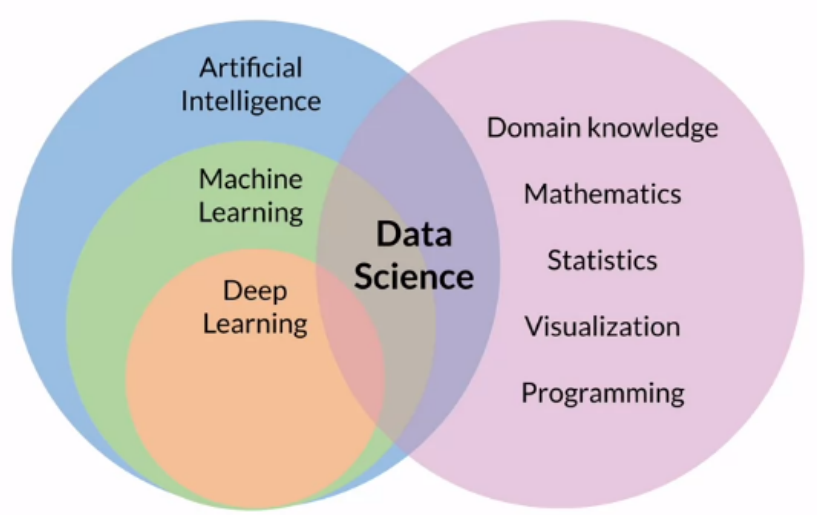
Welcome

I will start this week with a brief introduction to the discipline of practical data science and discuss the benefits of performing data science in the Cloud. Practical data science helps you to improve your data science and machine learning skills, work with almost any amount of data and implement your use cases in the most efficient way for your use case. By moving your data science projects into the Cloud, you are no longer bound by resource limitations, such as your laptop, CPU, or memory. You can run data analysis on virtually any size of data. You can slice your data and run data transformations in parallel. You can switch from CPU to GPU if you want to speed up your model training and you can do much of this in just a few clicks. In the course of this week, I will give you a short overview of the data science and machine learning concepts, you will learn to apply and the toolkits you will use. I will then move on to define the task and the use case you will focus on. To give you our first hint, you will learn how to work with text data. Specifically, you will focus on a multi-class classification for sentiment analysis of product reviews. As always, data science starts with data. I will introduce you this week to the dataset you will work with to implement this text classification task. You will learn how easy it is to ingest the data into a central repository and explore the data using various tools from our practical data science and machine learning toolbox. You will learn how to analyze the data further using interactive queries and learn how to visualize the results. The analysis will reveal important information for future tasks in the model development process, as you will see.

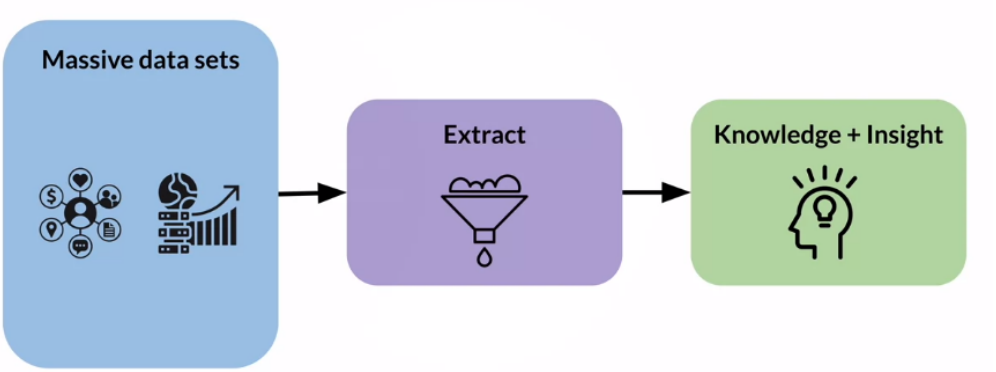
Practical Data Science

I want to briefly define what's behind practical data science in the Cloud and introduce you to the data science and machine learning concepts which you will learn to apply and the toolkits you will learn to use.

Let me start by comparing the terms artificial intelligence, machine learning, deep learning, and data science.



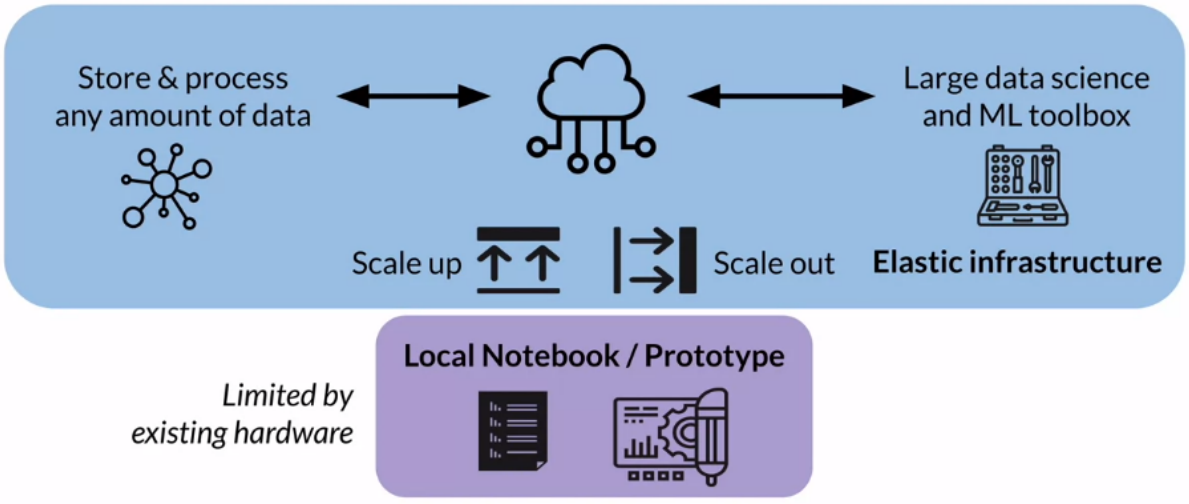
Artificial intelligence or AI is generally described as a technique that lets machines mimic human behavior. **Machine learning** or ML is a subset of AI that uses statistical methods and algorithms that are able to learn from data without being explicitly programmed. Then finally, **deep learning** is yet another subset of machine learning that uses artificial neural networks to learn from data. If you're new at data science, you see that this discipline touches all fields. **Data science** truly is an interdisciplinary field that combines business and domain knowledge with mathematics, statistics, data visualization, and programming skills.



Now, what's practical data science? **Practical data science** helps you to improve your data science and machine learning skills, work with almost any amount of data and implement their use cases in the most efficient way. It's different from working on a local development environment, such as your laptop, with small curated datasets. Practical data science is geared towards handling massive datasets that could originate from social media channels, mobile and web applications, public or company internal data sources, and much more, depending on the use case you're working on. This data is often messy, potentially error written, or even poorly documented. Practical data science tackles these issues by providing tools to analyze and clean

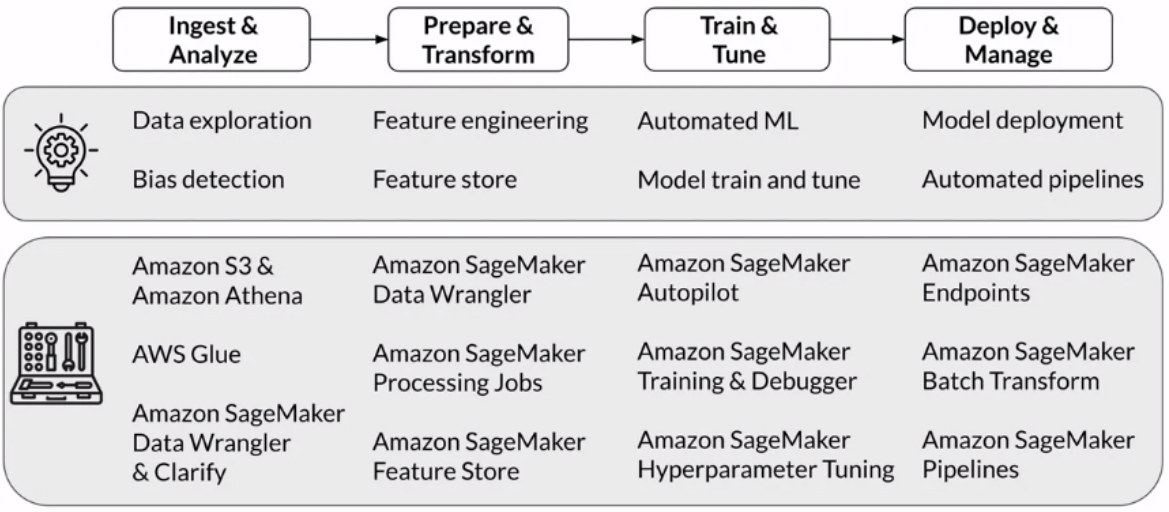
the data and to extract relevant features. This process leads to the ultimate goal of data science, which is knowledge distillation and gaining insight from those large datasets.

What's different about **practical data science in the Cloud**? If you develop data science projects in a local notebook or IDE environment for example, hosted on your laptop or company owned server pool you are limited by the existing hardware resources. For example, you have to carefully watch how much data you process and potentially lodge into memory. How much CPU processing power you have available to train and tune your model. If you need more, you need to buy additional computer resources. This process doesn't allow you to move and develop quickly. One of the biggest benefits of developing and running data science projects in the Cloud is the agility and elasticity that the Cloud offers. Maybe your model training takes too long because it consumes all of the CPU resources of the compute instance you have chosen. You can switch to using a compute instance that has more CPU resources, or even switch up to a GPU based compute instance. This is referred to as scaling up or instead of training your model on a single CPU instance, say you want to perform distributed model training in parallel across various compute instances. This is referred to as *scaling out*. Both scenarios can be accomplished in the Cloud within a few seconds. When the model training is completed, the instances are terminated as well. This means you only pay for what you actually use. This elastic infrastructure allows you to store and process almost any amount of data because the infrastructure scales too much the required resources. You can also innovate faster because you can try new datasets, new models, new code, or new machine learning libraries quickly and without any upfront investments. The Cloud also comes with a large data science and machine learning toolbox, you can choose from to perform your tasks as fast and efficiently as possible.

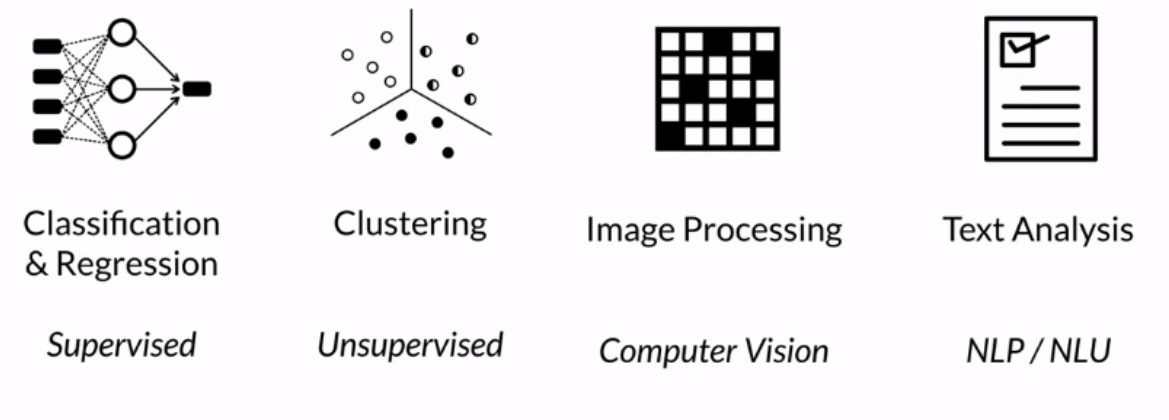


Let's have a look at a typical **machine learning workflow** and of course, every workflow starts with data. In the *ingest and analyze step*, you explore the data and analyze the data for potential statistical bias. Looking at the toolbox for this step, you will use Amazon Simple Storage Service or Amazon S3 and Amazon Athena to ingest, store and query your data. With AWS Glue, you will catalog the data in its schema. For statistical bias detection in data, you will learn how to work with Amazon SageMaker Data Wrangler and Amazon SageMaker Clarify, and don't worry right now, the tools will be explained in much more detail as you move throughout this course and the following courses of this specialization, as you learn how to apply each concept to implement the text classification use case. In the following weeks, you will step into the model development stage, where you'll start with *preparing and transforming* the data for model training. You will learn how to perform feature engineering and learn about the concept and

benefits of a feature story. Again, you will work with a powerful set of tools that are part of the Amazon SageMaker service. In the *model training and tuning stage*, you will learn how to use automated machine learning, cheek rate of first baseline, and a set of best model candidates for the use case. You will also perform custom model training and tuning in a later week. Again, you will work with additional tools part of the Amazon SageMaker service. In the *model deployment and management stage*, you will learn to discuss different model deployment options and strategies and how to orchestrate the model development as an automated pipeline. Similarly, your toolkit will be based on Amazon SageMaker.

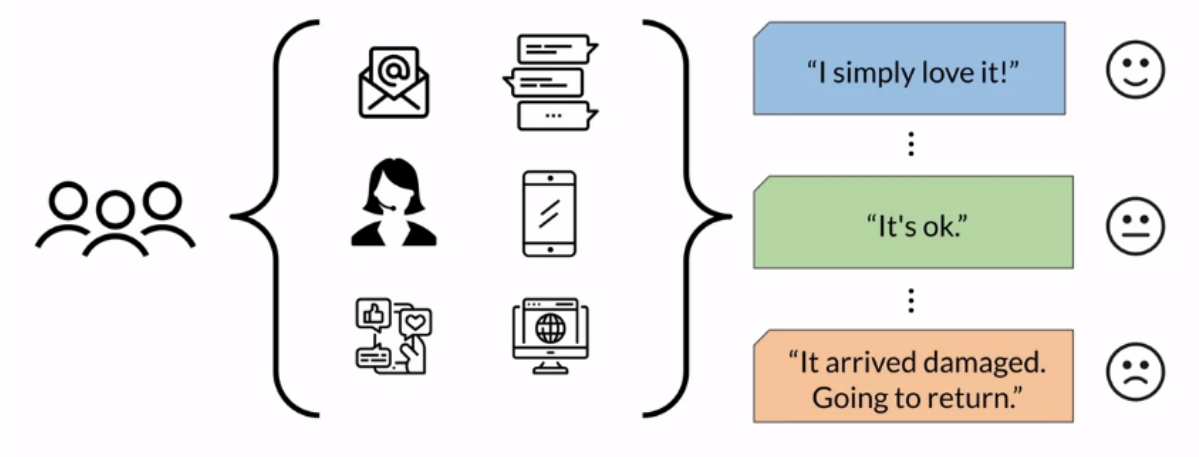


Use case and data set



Many use cases can be classified into one of the major machine learning tasks and learning paradigms. Popular machine learning tasks are *classification and regression problems*, which are examples of **supervised learning**. In supervised learning, you learn by example by providing the algorithm with labeled data. With classification, the goal is to assign the input sample to a defined class. For example, is this email I received spam or not spam? In contrast, regression applies statistical methods to predict a continuous value, such as a house price given a set of related and non related input variables. Another popular task is *clustering* and clustering is an example of **unsupervised learning**. Here the data is not labelled, hence doesn't provide any examples. The clustering algorithm tries to find patterns in the data and starts grouping the data points into

distinct clusters. A clustering use case could be identifying different customer segments for marketing purposes. If you look more at the fields of AI applications, *image processing* is a major task as part of the broader scientific field of **computer vision**. You might need to classify images into pictures of dogs and cats, identify segments in the image to help your car's driver assistance systems to distinguish between speed signs and trees or to detect brand labels in an image. Following computer vision, *text analysis* has regained popularity and research and the industry in recent years. The field of **Natural Language Processing** or NLP or Natural Language Understanding NLU have been studied since the 1950s, but thanks to advances in deep learning and neural network architectures. You can see more advanced NLP tasks such as neural machine translations, sentiment analysis, question answering and others being implemented. And you will learn much more about the field of NLP and text analysis because this is the task you will focus on in this course.



More specifically, you will perform multi-class classification for sentiment analysis of product reviews. Assume you work at an e-commerce company selling many different products online. Your customers are leaving product feedback across all the online channels. Whether it is through sending email, writing chat FAQ messages on your website, maybe calling into your support center or posting messages on your company's mobile app, popular social networks or partner websites. And as a business, you want to be able to capture this customer feedback as quickly as possible to spot any change in market trains or customer behavior and then be alerted about potential product issues. Your 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. A great resource to explore

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)

product reviews are e-commerce sites. You can use the review text as the input feature for the model training and the sentiment as a label for model training. The sentiment class is usually expressed as an integer value for model training such as 1 for positive sentiment, 0 for neutral sentiment and -1 for negative sentiment.

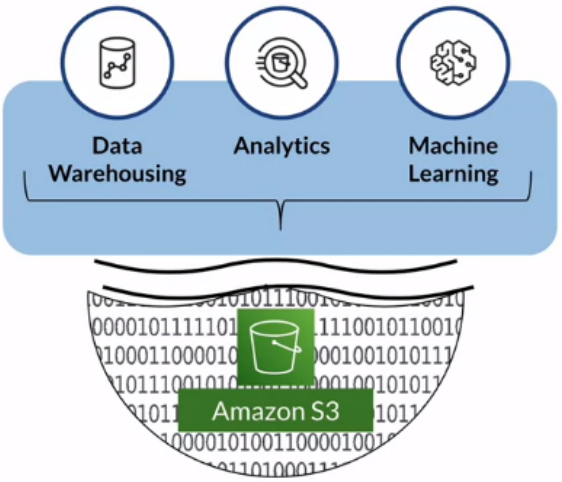
Working with Data

Data ingestion and exploration

One of the largest advantages of performing data science in the cloud is that you can store and process virtually any amount of data. The infrastructure scales elastically with the size of your data. Just think of the product reviews use case you will be working on. Imagine your e-commerce company is collecting all the customer feedback across all online channels. You need to capture sudden customer feedback streaming from social media channels, feedback captured and transcribed through supports under calls, incoming emails, mobile apps, and website data, and much more. To do that, you need a flexible and elastic repository that can start not only the different file formats, such as dealing with structured data, CSV files, as well as unstructured data such as support center call audio files. But it also needs to elastically scale the storage capacity as new data arrives.

- Centralized and secure repository
- Store, discover and share data at any scale
  - structured relational data
  - semi-structured data
  - unstructured data
  - streaming data
- Governance

Cloud based **data lakes** address this problem. Think of a data lake as this centralized and secure repository that can store, discover, and share virtually any amount and any type of data. You can ingest data in its raw format without any prior data transformation. Whether it's structured relational data in the form of CSV or TSV files, semi-structured data such as JSON or XML files, or unstructured data such as images, audio, and video files. You can also ingest streaming data, such as an application delivering a continuous feed of log files or feeds from social media channels into your data lake. A data lake needs to be governed. With new data arriving at any point in time you need to implement ways to discover and catalog the new data. You also need to secure and control access to the data to comply with the political data security, privacy, and governance regulations. With this governance in place, you can now give data signs and machine learning teams access to large and diverse datasets.



- Amazon Simple Storage Service (Amazon S3)
- Object storage
- Durable, available, exabyte scale
- Secure, compliant, auditable

Data lakes are often built on top of objects storage such as **Amazon S3**. You're probably familiar with file and block storage. 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. You can ingest virtually anything from just appear dataset files to exabytes of data. AWS also provides additional tools and services to assist you in building a secure, compliant, and auditable data lake on top of S3. With a data lake in place, you can now use this centralized data repository to enable data warehousing analytics and also machine learning.

AWS Data Wrangler

- Open source Python library
- Connects pandas DataFrames and AWS data services
- Load/unload data from
  - data lakes
  - data warehouses
  - databases

```
!pip install awswrangler

import awswrangler as wr
import pandas as pd

# Retrieving the data directly from Amazon S3
df = wr.s3.read_csv(
     path='s3://bucket/prefix/')
```

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 awswrangler"* command. Here's a sample code snippet that shows how you can work with AWS Data Wrangler. First, you import the library together with Pandas. If you want to read, for example, CSV data from your S3 data lake into a Pandas DataFrame, you can run this command shown here using the S3 read CSV function, providing the S3 path to your data.

AWS Glue Data Catalog

Another tool you will be using this week is the **AWS Glue Data Catalog**. This data catalog service is used to register or catalog the data stored in S3. Similar to taking inventory in a shop, you need to know what data is stored in your S3 data lake or bucket as an individual container for objects is called. Using the Data Catalog Service, you create a reference to the data basically S3 to table mapping. The AWS Glue table which is created inside an AWS Glue database only contains the metadata information such as the data schema. *It's important to note that no data is moved. All the data remains in your S3 location*. You catalog where to find the data and which schema should be used to query the data. Instead of manually registering the data, you can also use **AWS Glue Crawler**. A Crawler can be used and set up to run on a schedule or to automatically find new data which includes inferring the data schema and also to update the data catalog.

AWS Glue
Data Catalog

Name	reviews
Database	dsoaws_deep_learning
Classification	csv
Location	s3://<bucket>/<prefix>

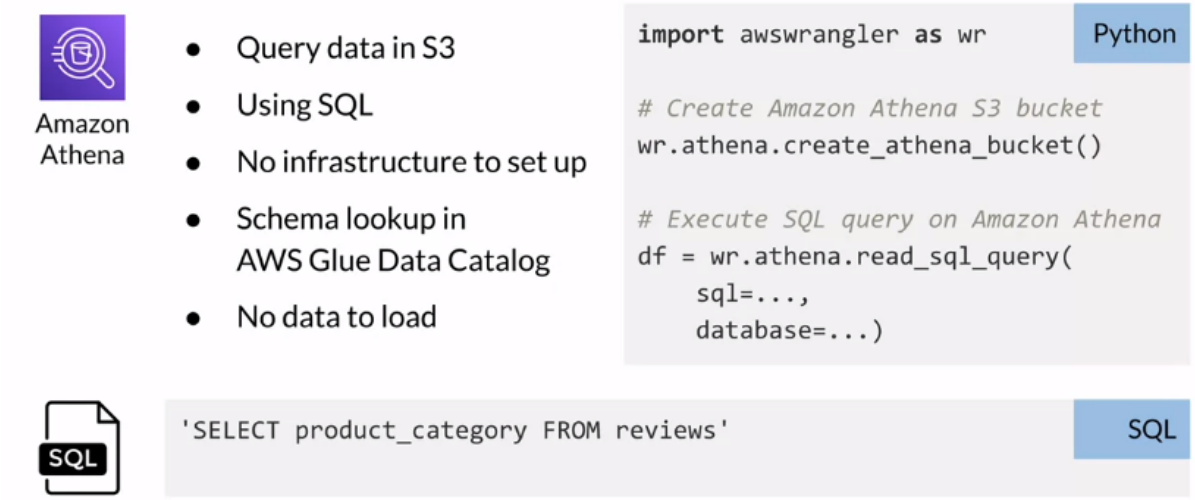
- Creates reference to data ("S3-to-table" mapping)
- Just metadata / schema stored in tables
- No data is moved
- *AWS Glue Crawlers* can be set up to automatically
  - infer data schema
  - update data catalog

Now how can you register the data? You can use the AWS Data Wrangler tool just as I introduced. The first step is to create an AWS Glue Data Catalog database. To do that, import the AWS Wrangler Python library as shown here, and then call the *catalog.create_database* function, providing a name for the database to create. AWS Data Wrangler also offers a convenience function called *catalog.create_CSV_table* that you can use to register the CSV data with the AWS Glue Data Catalog. The function will only store the schema and the metadata in the AWS Glue Data Catalog table that you specify. The actual data again remains in your S3 bucket.

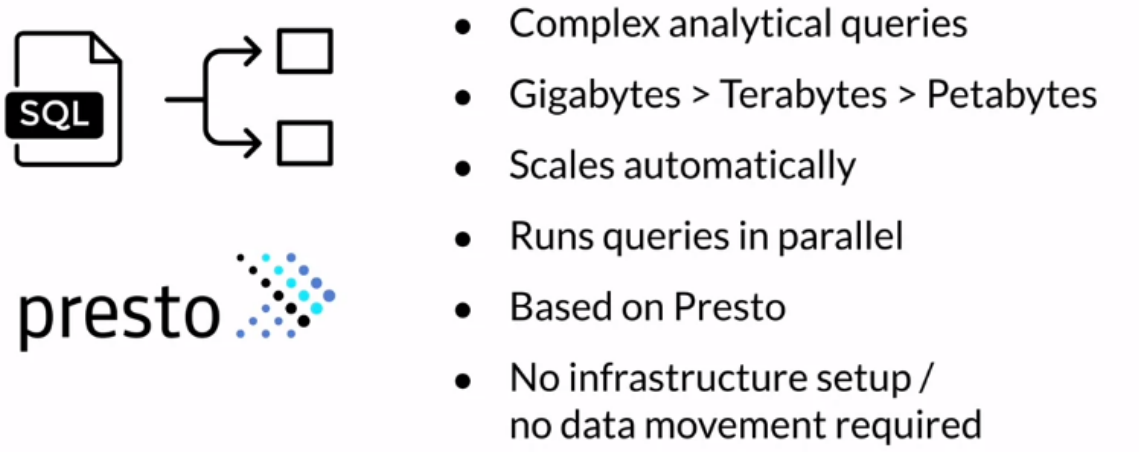
```
import awswrangler as wr

# Create a database in the
# AWS Glue Data Catalog
wr.catalog.create_database(
     name=...)

# Create CSV table (metadata only) in the
# AWS Glue Data Catalog
wr.catalog.create_csv_table(
     table=...,
     column_types=...,
     ...)
```



Now you can query the data stored in S3 using a tool called **Amazon Athena**. Athena is an interactive queries service that lets you run standard SQL queries to explore your data. Athena is serverless which means you don't need to set up any infrastructure to run those queries, and no matter how large the data is that you want to query, you can simply type your SQL query referencing the dataset schema you provided in the AWS Glue Data Catalog. No data is loaded or moved, and here is a sample SQL query. Let's list all product categories from the AWS Glue table called reviews, and imagine this table points to the dataset stored in S3. To run this query, you can use the previously introduced AWS Data Wrangler tool again. 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.



Given this simplicity, you might wonder what is so special about this, and I have to admit the SQL query I've shown here was fairly simple. But just imagine building highly complex analytical queries to run against not just gigabytes, but potentially terabytes or petabytes of data. Using Athena, you don't have to worry about any compute and memory resources to support this query, because Athena will automatically scale out and split the query into simpler queries to run in parallel against your data. Athena is based on Presto, an open source distributed SQL engine developed for this exact use case, running interactive queries against data sources of all sizes. Remember, no installation or infrastructure setup is needed, and no data movement is required. Just register your data with AWS Glue and use Amazon Athena to explore your datasets from the comfort of your Python environment.

Data visualization

Visualizing your data is one of the most effective ways of exploring your data across multiple dimensions at once. Depending on what kind of data you are exploring and what kind of relationships in the data you're looking for. The type of visualizations you use might be different. Let's take a look at a few of those visualizations you will use this week, and the tools you will work with.

Pandas, as introduced earlier, is used for data analysis and data manipulation. **NumPy is** used to perform scientific computing in python. Similarly, matplotlib and Seaborn, are popular python libraries for creating visualizations. **Matplotlib** helps to create static animated and interactive visualizations. **Seaborn** is based on matplotlib, and adds statistical data visualizations.

You will use these tools to visualize the product reviews.

The purpose of exploring and visualizing the data set is to better understand data set characteristics, from simple things such as understanding the number of samples you have available for model training to answering more complex business questions. Let me start with the first one. To get an understanding of how many samples the data set contains, let's run this SQL query using amazon Athena, which will return the number of reviews in each sentiment class. The query result is best suited to be visualized in the bar chart.

SQL Query

```
SELECT sentiment, COUNT(*) AS count_sentiment
FROM dsoaws_deep_learning.reviews
GROUP BY sentiment
ORDER BY sentiment DESC, count_sentiment
```

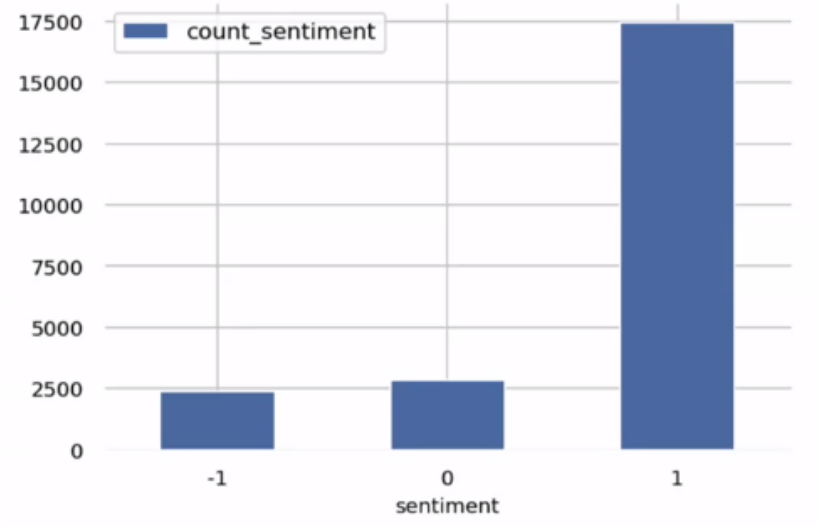
Here is the visualization code. I import the matplotlib library, for a simple bar chart, you can use the pandas data frame that holds the query results, and then call the plot bar function directly. You define the data frame columns which are used as the X and Y axis data, and at any additional data such as title to the plot. You can also use matplotlib to enrich the plot, let's say with labels for the X and Y axis. And when you're done, you can call the show function, and here can see the result in a sample bar chart.

Python visualization code

```
import matplotlib.pyplot as plt
chart = df.plot.bar(
     x="sentiment",
     y="count_sentiment")

plt.xlabel("sentiment")
plt.show(chart)
```

The bar chart also makes it easy to see that the positive sentiment class here, numbered with one, contains many more samples than any other of the classes. You basically have an unbalanced distribution of samples across the sentiment classes. You will learn how to address this in later week.



Here is another interesting visualization, this time I want to show you how to calculate percentiles and visualize data distributions. For this purpose, I will calculate the distribution of the review length or the number of words per review. The SQL query here is pretty straightforward. I split the review text in the column review body by a space character which gives me a list of the individual words, and then I calculate the cardinality which gives me the number of words.

SQL Query

```
SELECT CARDINALITY(SPLIT(review_body, ' ')) as num_words
FROM dsoaws_deep_learning.reviews
```

Before I plot the distribution, I want to calculate the percentiles. You can use the described function on the panda's data frame that contains the review length, and you specify the percentiles to calculate. To visualize the distribution of review length, I choose a histogram. Histograms represent frequency distributions. They show how often each different value occurs.

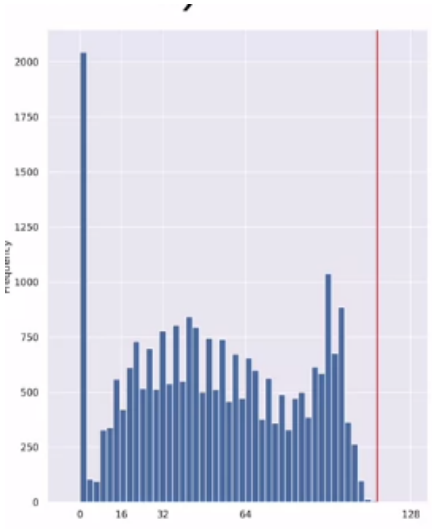
In this case, I want to group the review length in 100 bins, and I add a red marker to highlight the 100th percentile.

Python visualization code

```
summary = df["num_words"].describe(
    percentiles=[0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00])

df["num_words"].plot.hist(
    xticks=[0, 16, 32, 64, 128, 256], bins=100,
    range=[0, 256]).axvline(x=summary["100%"], c="red")
```

Let's have a look at the result, and here we go. On the right, you can see sample results of the percentile calculations. You can see that the shortest review had only one word, and if you remember I grouped all review length into the 100 bins, which are represented here by the blue bars in the histogram. The fewer bins you specify, the fewer bars show up here. The X axis shows the length and the Y axis represents their frequency. And if you look at the 100 percentile highlighted and right here, you can see that all of the reviews have 115 words fewer in this sample. And this information will be very helpful later in the course when you start building the text classifier model.



mean	52.51
std	31.38
min	1.00
10%	10.00
20%	22.00
30%	32.00
40%	41.00
50%	51.00
60%	61.00
70%	73.00
80%	88.00
90%	97.00
100%	**115.00**

Additional reading material

If you wish to dive more deeply into the topics covered this week, feel free to check out these optional references. (You won't have to read these to complete this week's practice quizzes.)

- AWS Data Wrangler
- AWS Glue
- Amazon Athena
- Matplotlib
- Seaborn
- Pandas
- Numpy

Week 2: Data Bias and Feature Importance

Statistical bias and feature importance

Statistical Bias

What is **statistical bias**? A data set is considered to be biased if *it cannot completely and accurately represent the underlying problem space*. For those of you familiar with statistics, you know that statistical bias is a tendency of a statistic to either overestimate or underestimate a parameter. In this course, you will learn about statistical biases in training data sets, which are imbalances in these training data sets. In these biased data sets, *some elements of a data set are more heavily represented than others*. Let's say for example, you are trying to build a financial services model that can be used to ***detect fraud***. The training data that you will use to build this model is a previous set of credit card transactions that a business has access to. Now for the majority of the time, credit card transactions are not fraudulent, which is really good for the business. But if you're using that biased data set to train a model to detect fraud, then your model is very unlikely to detect fraudulent transactions because it has not seen that many fraudulent transactions before. One way to address this problem is to *add more examples* of how fraud transactions would look like to your training data set. Biased data sets typically lead to biased models, and the biased models could have both business and regulatory consequences for the businesses that use these models.

- Training data does not comprehensively represent the problem space
- Some elements of a dataset are more heavily weighted or represented

Fraud Detection

Biased models

- Imbalances in product review dataset

Let's take another example. Let's construct a ***product's review*** data set. Such a data set could be biased if it contains disproportionately large number of reviews for, let's say, one product category called A and fewer number of reviews for other categories like product category B and C. When you build a product sentiment prediction model with this biased data set, the resulting model could very well detect sentiment of new products that belong to product category A. But for newer products that belong to other categories such as product category B and C, your sentiment model is not going to be really accurate.

Statistical bias causes

What causes **statistical bias**? How does bias even get introduced into your datasets? There could be multiple reasons for that.

The first one that we see here is **activity bias**. This is biases that exist in *human-generated content*, especially on social media. Think about all the data that has been collected over these social media platforms over the last several years. Reality is that a very small percentage of the population is actively participating on these social media platforms. So the data that has been collected over the years on these platforms is not representative of the entire population.

The second one is very similar but slightly different. This is **societal bias**. This is once again, biases in *data that is generated by humans*, but maybe not just on social media. These biases could be introduced because of preconceived notions that exist in society. Data generated by humans can be biased because all of us have unconscious bias.

Sometimes bias can be introduced by the machine learning system itself. Let's say, for example, a machine learning application gives users a few options to select from, and once the user selects an option, the user selection is used as training data to further train and improve the model. This introduces *feedback loops*. Take, for example, a streaming service. You want to watch a movie on the streaming service and the streaming service makes a few recommendations for you and you decide to watch Dancing with Wolves. You like the movie and you rate it high. From then on, the streaming service is recommending you the movies that have wolves in them. It's partly because of the feedback you provided to the service. But in reality, maybe you watched that movie because you like the actress in the movie and you don't even particularly like wolves. Situations like this could result in **selection bias** that includes a feedback loop that involves both the model consumers and the Machine Learning model itself.

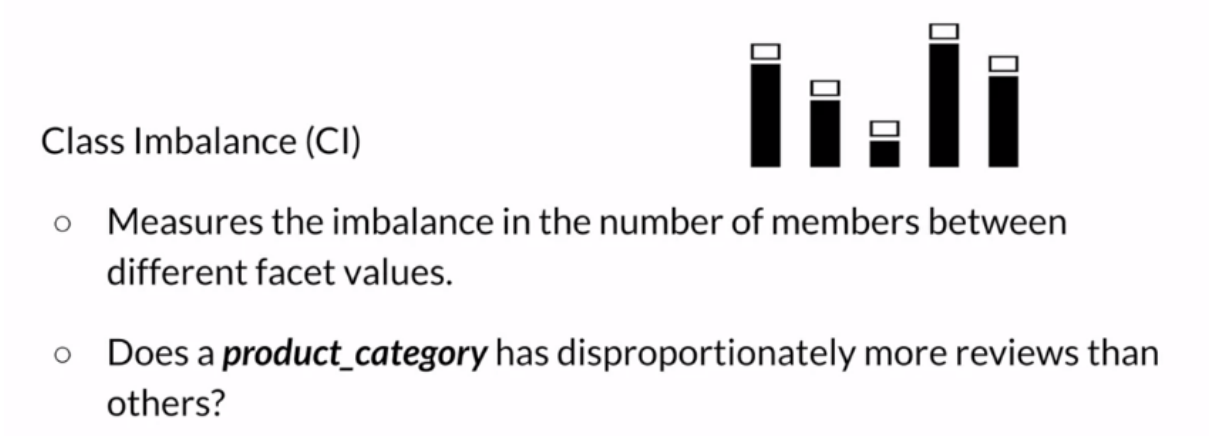
Now, even if you detect some of the statistical biases in your dataset prior to training your model, once the model is trained and deployed, drift can still happen. **Data drift** happens, especially when the *data distribution significantly varies from the distribution of the training data that was used to initially train the model*. This is called data drift and also data shift. There are several different variations of data drift. Sometimes the distribution of the independent variables or the features that make up your dataset can change. That's called *covariant drift*. Sometimes the data distribution of your labels or the targeted variables might change. That's the second one, which is *prior probability drift*. Sometimes the relationship between the two, that is the relationship between the features and the labels can change as well. That's called *concept drift*. Concept drift also called concept shift can happen when the definition of the label itself changes based on a

particular feature like age or geographical location. Take, for example, my experience. Last time when we traveled a few years ago across US on a road trip, we quickly found out that the soft drinks are not called the same across US. So when we stopped for meals and ordered soft drinks, we realized that soda is not called soda across the US. In some areas, it's called pop, and in some areas, it's called soda. Now, if you think about all the geographies across the world, you can only imagine the interesting combinations, different labels you can come up with. With all these issues that could potentially happen with your datasets, it becomes really important to continuously monitor and detect various biases that could be prevalent in your training datasets before and after you train your models.

Measuring statistical bias

Now, you're ready to measure the imbalances and the statistical bias in your dataset using specific metrics. When I talk about these metrics, it's important to understand that these metrics are applicable to a particular facet of your dataset. A **facet** is a sensitive feature in your dataset that you want to analyze for these imbalances. Let's take, for example, the product review dataset. In that dataset or product category could be a facet or a feature interest for you that you want to analyze for imbalances.

The first metric that I will introduce is **class imbalance**. It is very easy intuitively to understand this. Class imbalance, or CI, measures the imbalance in the number of examples that are provided for different facet values in your dataset. When you apply this to the product review dataset, it answers this particular question, does a particular product category, such as product category A, have a disproportionately larger number of total reviews than any other category in the dataset.



The next metric that I will introduce is DPL, this is the difference in proportions of labels. This metric measures the imbalance of positive outcomes between the different facet values. When applied to the product review dataset, what this metric is measuring is if a particular product category, say product category A, has disproportionately higher ratings than other categories. While CI, the metric that we just saw as measuring if a particular category has a total number of reviews higher than any other categories, DPL is actually looking for higher ratings than any other product categories. Now, I'm introducing only a couple of different metrics here, but there are several other metrics that can be used to measure different portions of bias across your data-sets.

Difference in Proportions of Labels (DPL) ✓ ✗

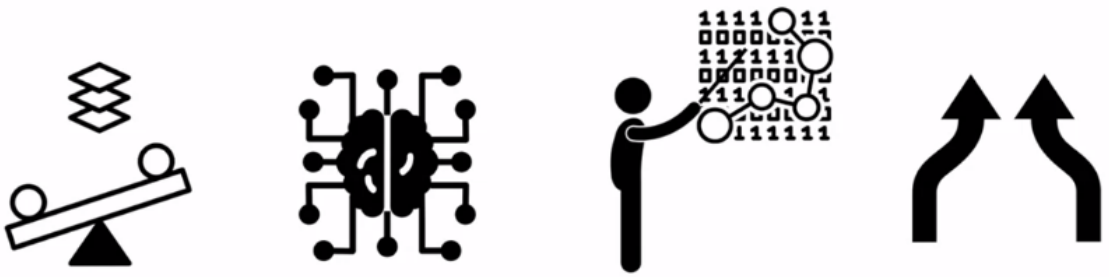
- Measures the imbalance of positive outcomes between different facet values.
- Does a ***product_category*** has disproportionately higher ratings than others

Detecting statistical bias

Next I will talk about two different tools that can help with detection of statistical bias in your training data sets. The two tools are **SageMaker Data Wrangler** and **SageMaker Clarify**. First, I will introduce Data Wrangler. Data Wrangler provides you with capabilities to connect to various different *sources* for your data, *visualize* the data and transform the data by applying any number of *transformations* in the Data Wrangler environment. And detect statistical bias in your data sets and *generate reports* about the bias detected in those data sets. It also provides capabilities to provide *feature importance* calculations on your training data set. You will have a chance to explore the various capabilities of Data Wrangler in different parts of the course. For this section, I will be focused on the ability to detect statistical bias and generate bias reports on the training data sets.

Detect statistical bias with Amazon SageMaker Clarify

Next, I will introduce **Amazon SageMaker Clarify** as a tool to perform statistical bias detection on your datasets. SageMaker Clarify can perform *statistical bias detection* and *generate bias reports* on your training datasets. Additionally, it can also perform bias detection in trained and deployed models. It further provides capabilities for *machine learning explainability*, as well as *detecting drift in data and models*. For now, I'm going to focus on the statistical bias detection and report generation capabilities of Clarify.



Statistical Bias Report | Model Bias Report | Explainability | Drift

To start using Clarify APIs, start by importing the Clarify library from the SageMaker SDK. Once you have the Clarify library, construct the object, SageMaker Clarify Processor using the library. **SageMaker Clarify Processor** is a construct that allows you to scale the bias detection process into a distributed cluster. By using two parameters, *instance type* and *instance count*, you can scale up the distributed cluster to the capacity that you need. 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. The processing capacity is measured by the node's compute capacity, memory, and the network [inaudible]. Once you have configured the distributor cluster, next, you specify an *S3 location* where you want the bias report to be saved to. That's the parameter bias report output path.

```
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

Once this step is done, the next step is to configure the data config object on the Clarify library. The **data config object** represents the details about your data. As you can expect, it has the *input and output location of your data*, an S3, as well as the label that you're trying to predict using that dataset. In this case here, the *label* that we are trying to predict is sentiment.

```
bias_data_config = clarify.DataConfig(
    s3_data_input_path=...,
    s3_output_path=...,
    label='sentiment',
    headers=df_balanced.columns.to_list(),
    dataset_type='text/csv')
```

Data Configuration

Next, you configure the **bias config object** on the Clarify library. The bias config object captures the *facet or the featured name* that you are trying to evaluate for bias or imbalances. In this case, you're trying to find out imbalances in the product category feature. The parameter *label values* or threshold defines the desired values for the labels. If the sentiment feature is your label, what is the desired value for that label? That value goes into the parameter label values or threshold.

```
bias_config = clarify.BiasConfig(
    label_values_or_threshold=[...],
    facet_name='product_category')
```

Bias Configuration

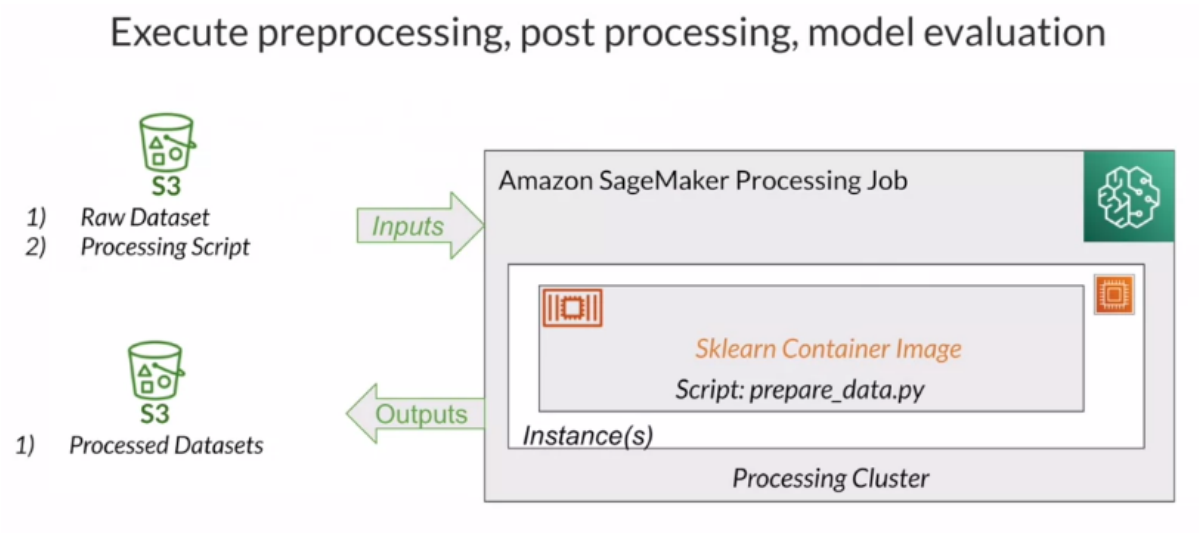
Once you have configured those three objects, you are ready to run the **pre-training bias method** on the Clarify processor. In addition to specifying the *data config* and the *data bias config* that you already configured, you can also specify the *methods* that you want to evaluate for bias. These methods are basically the metrics that you've already learned about to detect bias. The metrics here are the CI, the class imbalance, and the DPL. You can also specify a few other methods here as well. The *wait* parameter specifies whether this bias detection job should block the rest of your code or should it be executed in the background. Similarly, the *logs* parameter specify whether you want to capture the logs or not.

```
clarify_processor.run_pre_training_bias(
    data_config=...,
    data_bias_config=...,
    methods=["CI", "DPL", ...],
    wait=<<False/True>>,
    logs=<<False/True>>)
```

Pre training bias job

Once the configuration of the pre-training bias method is done, you launch this job. In the background, SageMaker Clarify is using a construct called **SageMaker Processing Job** to execute the bias detection at scale. SageMaker Processing Jobs is a construct that allows you to perform any data-related tasks at scale. These tasks could be executing pre-processing, or post-processing tasks, or even using data to evaluate your model. As you can see in the figure here, the SageMaker Processing Job expects the data to be in an S3 bucket. The data is collected from the S3 bucket and processed on this processing cluster which contains a variety of containers in the cluster. By default, containers for scikit-learn, Python, and a few others are supported. You can also have the opportunity to bring your own custom container as well. Once

the processing cluster has processed the data, the transformed data or the processed data is put back in the S3 bucket.



Now, you understand how to use the Clarify APIs and what happens behind the scenes as well. What do you think happens when you execute this run pre-training bias? What do you think the result is going to be? *The result will actually be a very detailed report on the bias on your dataset that has persisted in S3 bucket.* You can download the report and review in detail to understand the behavior of your data.

Approaches to statistical bias detection

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.

Feature importance: SHAP

Next, I will talk about feature importance and the open source framework SHAP that is behind feature importance. In this section, you will also learn about using SageMaker Data Wrangler to calculate feature importance on the product review data set.

Feature importance is the idea of explaining the individual features that make up your training data set using a score called important score. Some features from your data set could be more relevant or more important to your final model than others. Using feature importance, you can rank the individual features in the order of their importance and contribution to the final model. Feature importance allows you to evaluate how useful or valuable a feature is in relation to the other features that exist in the same data set. Let's take for example, the product review data set. It consists of multiple different features and you are trying to build a product sentiment prediction model out of that data set. You would be interested in understanding what features will play a role towards that final model, that is where feature importance comes into picture.

- Explains the features that make up the training data using a score (importance).

- How useful or valuable the feature is relative to other features.
- Predict the sentiment for a product → Which features play a role?

Feature importance is based on a very popular open source framework called **SHAP**, SHAP stands for *shapley additive explanations*.

- **Open Source Framework - SHAP**
  - Shapley values based on game theory.
  - Explain predictions of a ML model
    - Each feature value of training data instance is a player in a game
    - ML prediction is the payout
  - Local vs global explanations
  - SHAP can guarantee consistency and local accuracy.

The framework itself is based on *shapley values*, which in turn is based on game theory. To understand how SHAP works, consider a play or a game in which multiple players are involved and there is a very specific outcome to the play that could be either a win or a loss. Shapley values allow you to attribute the outcome of the game to the individual players involved in the

game. Translating that into the machine learning world, you can use the same concept to *explain the predictions* made by the machine learning model. In this case, the individual players would be the individual features that make up the data set and the outcome of the play would be the machine learning model prediction. So using the same concept, you can explain how the predictions will correlate to the individual feature values that make up your training data set. Using the SHAP framework, you can provide both *local and global explanations*. While the local explanation focuses on indicating how an individual feature contributes to the final model. The global explanation takes a much more comprehensive view in trying to understand how the data in its entirety contributes to the final outcome from the machine learning model. SHAP framework is also very extensive in nature in that it considers all possible combinations of feature values along with all possible outcomes for your machine learning model. Because of this extensive nature, the SHAP framework could be very time intensive, but also because of this extensive nature, SHAP can provide you with guarantees in terms of *consistency and local accuracy*.

Additional reading material

If you wish to dive more deeply into the topics covered this week, feel free to check out these optional references. (You won't have to read these to complete this week's practice quizzes.)

- Measure Pretraining Bias - Amazon SageMaker
- SHAP

Week 3: Use Automated Machine Learning to train a Text Classifier

Automated Machine Learning