Create a Data Context

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

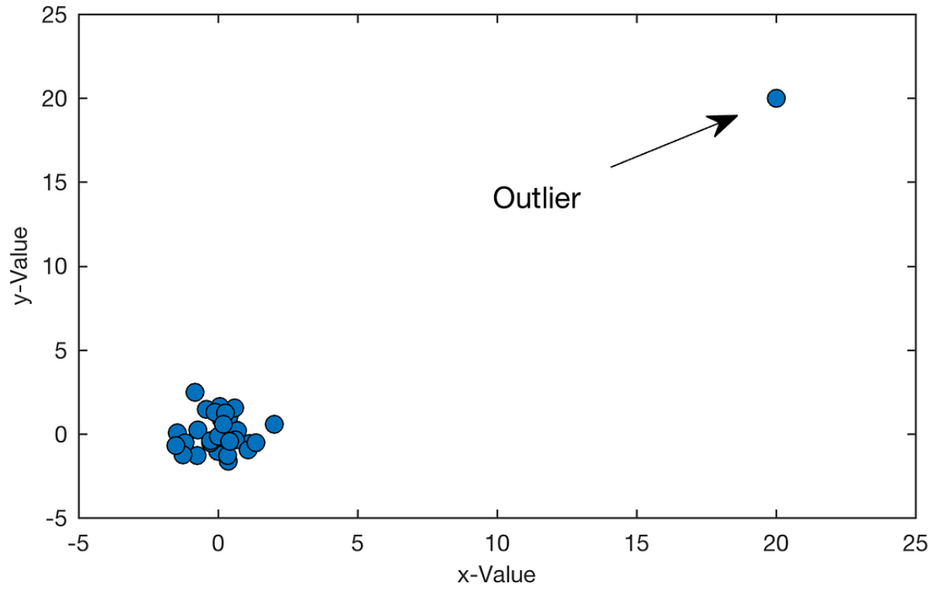
Davina Moossazadeh
Data Scientist

datacamp

What is data quality?

How fit a dataset is for its intended purpose

- Completeness
- Accuracy
- Validity
- Uniqueness
- Timeliness
- Integrity
- Consistency
- etc.



[1] https://nitin9809.medium.com/outlier-detection-and-treatment-part-1-aa0b09f60e50

Why is data quality important?

A model can only be as good as the the data going in!

What is Great Expectations?

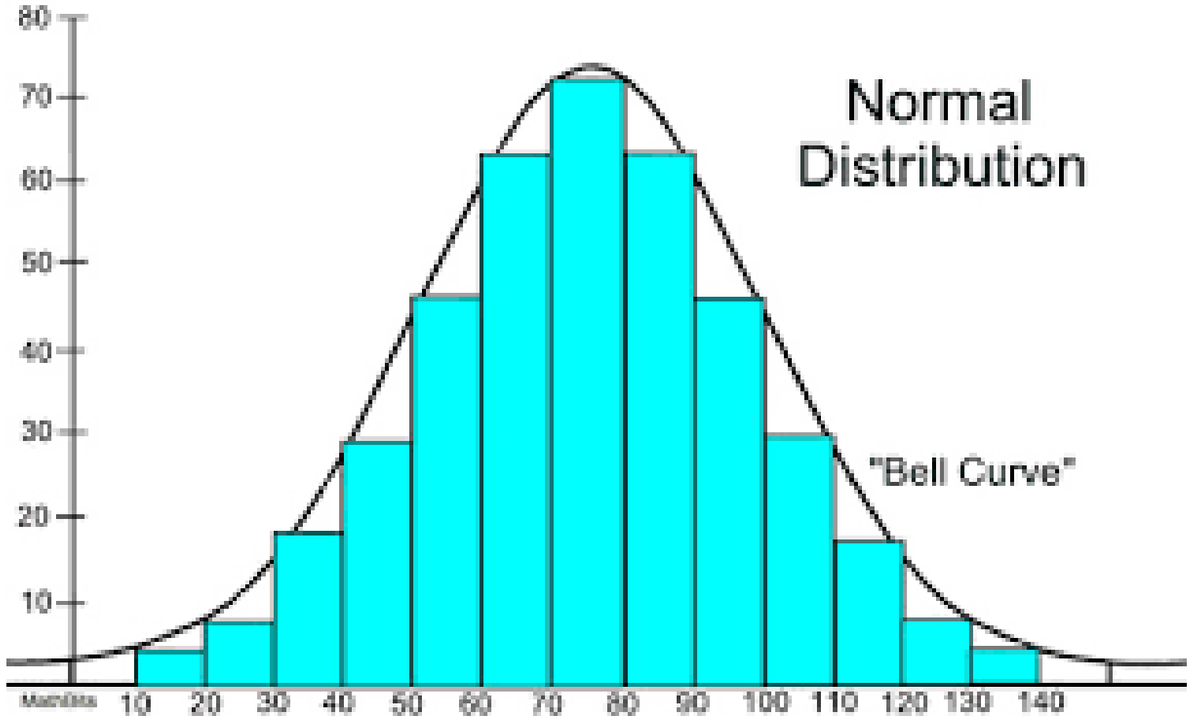
Great Expectations (GX) - A platform for managing data quality

- **GX Cloud** - web-based UI
- **GX Core** - Python package

Expectations

Expectation - Verifiable assertion about data

- Dataset shape
- Null values
- Duplicates
- Value sets/ranges
- String formatting
- Data distributions
- Data quality issues
- etc.



[1] https://docs.greatexpectations.io/docs/core/define_expectations/create_an_expectation/
https://mathbitsnotebook.com/Algebra2/Statistics/STnormalDistribution.html

Data Contexts

Data Context - The primary entry point for a GX deployment

- Configurations and methods for all supporting GX components
  - Data Sources
  - Expectation Suites
  - Checkpoints
  - Data Docs
  - Validation Results
  - Metrics

[1] https://docs.greatexpectations.io/docs/core/set_up_a_gx_environment/create_a_data_context/

Importing GX

Import Great Expectations with the alias `gx` :

```
import great_expectations as gx
```

Creating a Data Context

Use `get_context()` to create the Data Context:

```
context = gx.get_context()
print(context)
```

```
{ "analytics_enabled": true,
  "checkpoint_store_name": "default_checkpoint_store",
  "config_variables_file_path": "uncommitted/config_variables.yml",
  "config_version": 4.0,
  "data_context_id": "5b407294-b17c-43e3-aa5f-4f8a4741e772",
  "expectations_store_name": "default_expectations_store",
  "fluent_datasources": {},
  "plugins_directory": "plugins/",
  "stores": {},
  "validation_results_store_name": "default_validations_store" }
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Connect to Data

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Components

GX components - Python classes that represent data and data validation entities

- Data Context
- **Data Sources** & **Data Assets** (✔)
- Batch Definitions & Batches ☐
- Expectations ☐
- Expectation Suites ☐
- Validation Definitions ☐
- Checkpoints & Actions ☐
- Data Docs ☐

[1] https://docs.greatexpectations.io/docs/core/introduction/gx_overview

Data Sources

Data Source - An object that tells GX how to connect to a specific source of external data

[1] https://docs.greatexpectations.io/docs/core/connect_to_data/dataframes/

Data Sources

Data Source - An object that tells GX how to connect to a specific source of external data

[1] https://docs.greatexpectations.io/docs/core/connect_to_data/dataframes/

Creating a Data Source

Manage Data Sources with the `.data_sources` attribute, using the `.add_pandas()` method:

```
data_source = context.data_sources.add_pandas(
    name="my_pandas_datasource"
)
```

Note: The `name` parameter in GX is different from the Python variable name

- You can assign them different values, e.g., `"my_pandas_datasource"` vs. `data_source`

[1] https://docs.greatexpectations.io/docs/core/connect_to_data/dataframes/

Data Assets

Data Asset - A collection of records within a Data Source

```
data_asset = data_source.add_dataframe_asset(
    name="my_dataframe_asset"
)
```

[1] https://docs.greatexpectations.io/docs/core/connect_to_data/dataframes/

Cheat sheet

Create Data Source from Data Context:

```
data_source = context.data_sources.add_pandas(
    name: str
)
```

Create Data Asset from Data Source:

```
data_asset = data_source.add_dataframe_asset(
    name: str
)
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Read Data in Batches

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

Kaggle Weather Data

	Location	Date_Time	Temperature_C	Humidity_pct	Precipitation_mm	Wind_Speed_kmh
0	Chicago	2024-01-06 02:59:46	26.786811	31.513614	0.496024	22.980095
1	Chicago	2024-04-16 00:07:29	17.587820	32.817923	0.128803	0.234146
2	Chicago	2024-04-01 03:12:48	-2.562660	30.356593	2.624328	2.601357
3	Chicago	2024-04-03 12:07:38	7.166150	50.377273	4.669553	11.841165
4	Chicago	2024-04-05 13:22:42	38.386233	74.049712	6.792913	3.292467
...	...	...	...	...	...	...
87114	Chicago	2024-03-23 15:27:39	30.363594	42.566362	6.472597	3.940268
87115	Chicago	2024-01-22 01:18:42	31.870160	71.849950	0.842965	22.321245
87116	Chicago	2024-01-31 15:53:16	-9.061359	72.270546	6.224749	22.943680
87117	Chicago	2024-01-25 01:43:57	16.432551	41.331437	4.395345	0.209104
87118	Chicago	2024-03-02 07:20:55	-6.500893	49.713982	3.050012	7.596484

Batch Definitions

Batch Definition - A configuration for how a Data Asset should be divided for testing

```
batch_definition = data_asset.add_batch_definition_whole_dataframe(
    name="my_batch_definition"
)
print(batch_definition)
```

```
id='69e2a81d-1c28-4d1a-b66e-52cdc1198266'
name='my_batch_definition'
partitioner=None
```

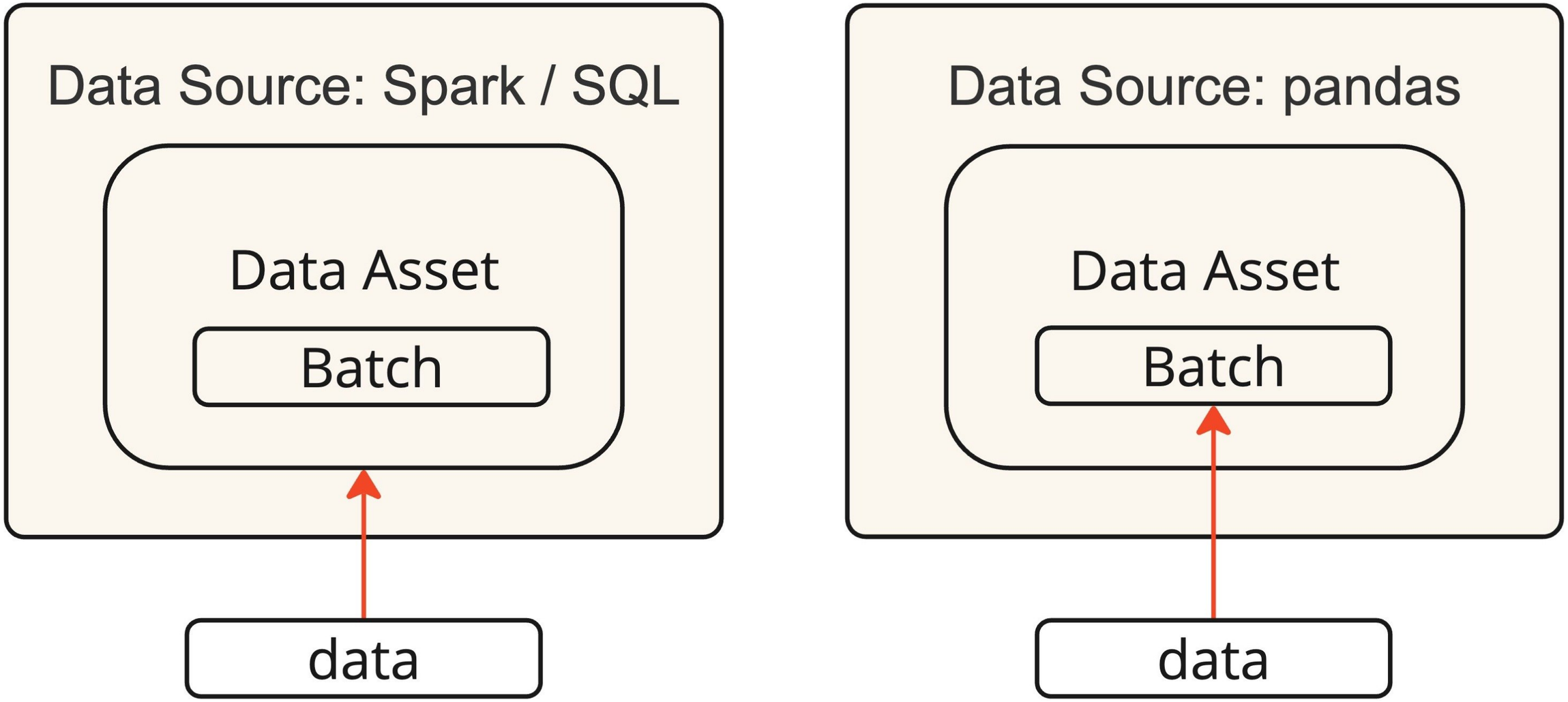
[1] https://docs.greatexpectations.io/docs/core/connect_to_data/dataframes/

Batches

Batch - A group of records that validations can be run on

```
batch = batch_definition.get_batch(
    batch_parameters={"dataframe": dataframe}
)
```

Batches


Data Source: Spark / SQL
Data Asset
Batch
data
Data Source: pandas
Data Asset
Batch
data

The Batch object

We can use `.head()` as with pandas:

```
print(batch.head())
```

Calculating Metrics: 100% 1/1 [00:00<00:00, 88.32it/s]

	Location	Date_Time	Temperature_C	Humidity_pct	Precipitation_mm	Wind_Speed_kmh
0	Chicago	2024-01-06 02:59:46	26.786811	31.513614	0.496024	22.980095
1	Chicago	2024-04-16 00:07:29	17.587820	32.817923	0.128803	0.234146
2	Chicago	2024-04-01 03:12:48	-2.562660	30.356593	2.624328	2.601357
3	Chicago	2024-04-03 12:07:38	7.166150	50.377273	4.669553	11.841165
4	Chicago	2024-04-05 13:22:42	38.386233	74.049712	6.792913	3.292467

[1] Table adapted from https://www.kaggle.com/datasets/prasad22/weather-data

The Batch object

```
print(batch.head(fetch_all=True))
```

	Location	Date_Time	Temperature_C	Humidity_pct	Precipitation_mm	Wind_Speed_kmh
0	Chicago	2024-01-06 02:59:46	26.786811	31.513614	0.496024	22.980095
1	Chicago	2024-04-16 00:07:29	17.587820	32.817923	0.128803	0.234146
2	Chicago	2024-04-01 03:12:48	-2.562660	30.356593	2.624328	2.601357
3	Chicago	2024-04-03 12:07:38	7.166150	50.377273	4.669553	11.841165
4	Chicago	2024-04-05 13:22:42	38.386233	74.049712	6.792913	3.292467
...	...	...	...	...	...	...
87114	Chicago	2024-03-23 15:27:39	30.363594	42.566362	6.472597	3.940268
87115	Chicago	2024-01-22 01:18:42	31.870160	71.849950	0.842965	22.321245
87116	Chicago	2024-01-31 15:53:16	-9.061359	72.270546	6.224749	22.943680
87117	Chicago	2024-01-25 01:43:57	16.432551	41.331437	4.395345	0.209104
87118	Chicago	2024-03-02 07:20:55	-6.500893	49.713982	3.050012	7.596484

87119 rows × 6 columns

The Batch object

`.columns()` shows all DataFrame columns (note the `()`)

```
print(batch.columns())
```

```
['Location',
 'Date_Time',
 'Temperature_C',
 'Humidity_pct',
 'Precipitation_mm',
 'Wind_Speed_kmh']
```

Cheat sheet

Create Batch Definition from Data Asset:

```
batch_definition = data_asset. \
add_batch_definition_whole_dataframe(
  name: str
)
```

Create Batch from Batch Definition:

```
batch = batch_definition.get_batch(
  batch_parameters={"dataframe": dataframe}
)
```

Get Batch DataFrame rows:

```
batch.head(fetch_all: bool)
```

Get Batch DataFrame column list:

```
batch.columns()
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS