

Basic Transformations

Making Data **Machine Learning Ready**

Charles Parker

VP ML Algorithms, BigML

In a Perfect World...



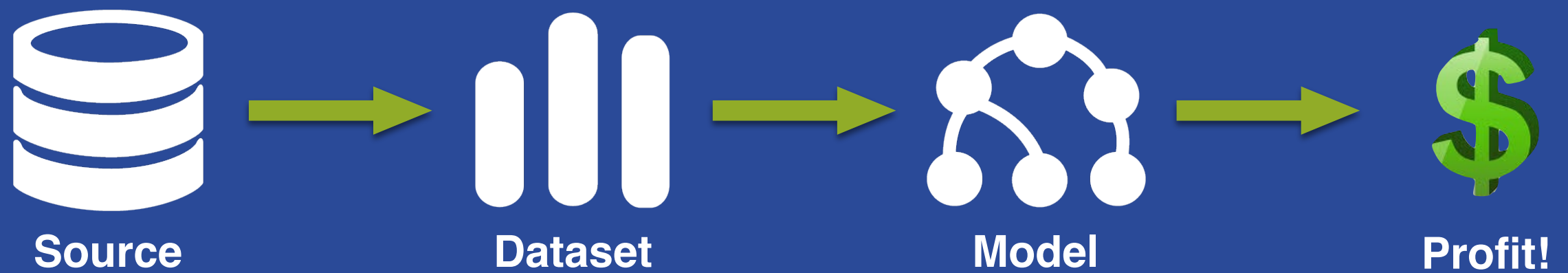
Q: How does a physicist milk a cow?

A: Well, first let us consider a spherical cow...

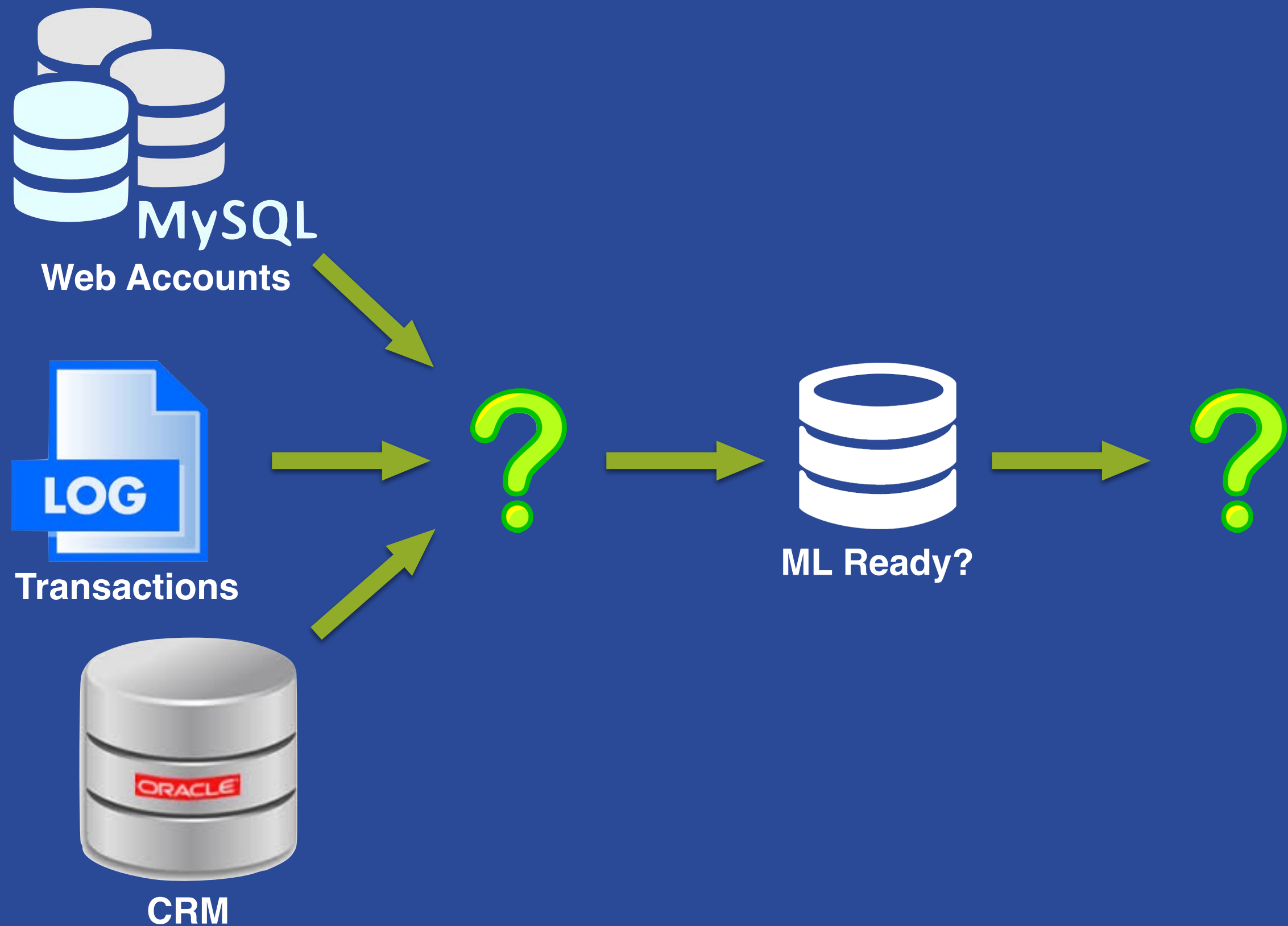
Q: How does a data scientist build a model?

A: Well, first let us consider perfectly formatted data...

The Dream



The Reality



Obstacles



- Data Structure
 - Scattered across systems
 - Wrong "shape"
 - Unlabelled data

Data Transformation

- Data Value
 - Format: spelling, units
 - Missing values
 - Non-optimal correlation
 - Non-existant correlation

Feature Engineering


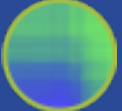




- Data Significance
 - Unwanted: PII, Non-Preferred
 - Expensive to collect
 - Insidious: Leakage, obviously correlated

Feature Selection

Data Structure

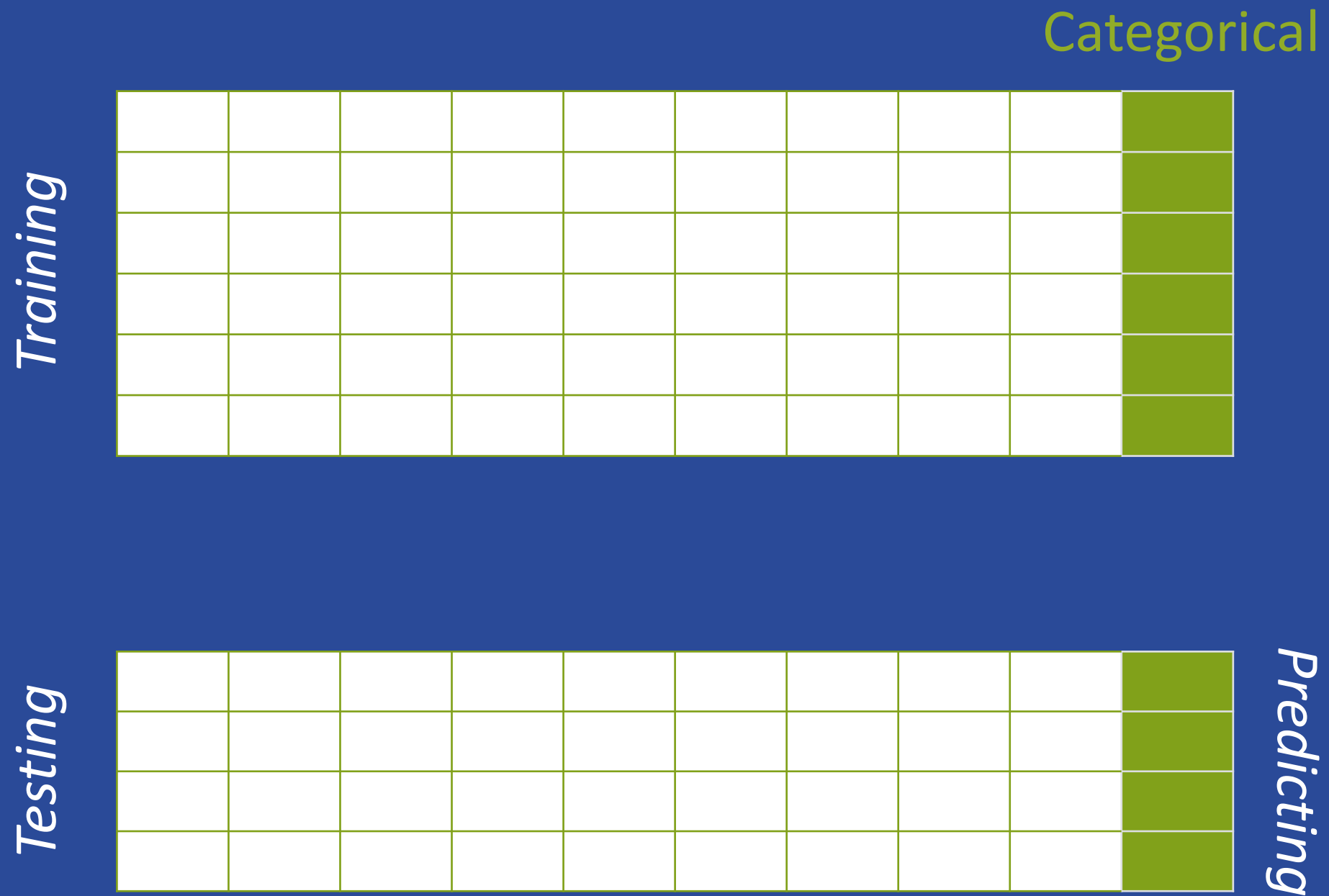
Remember ML Tasks



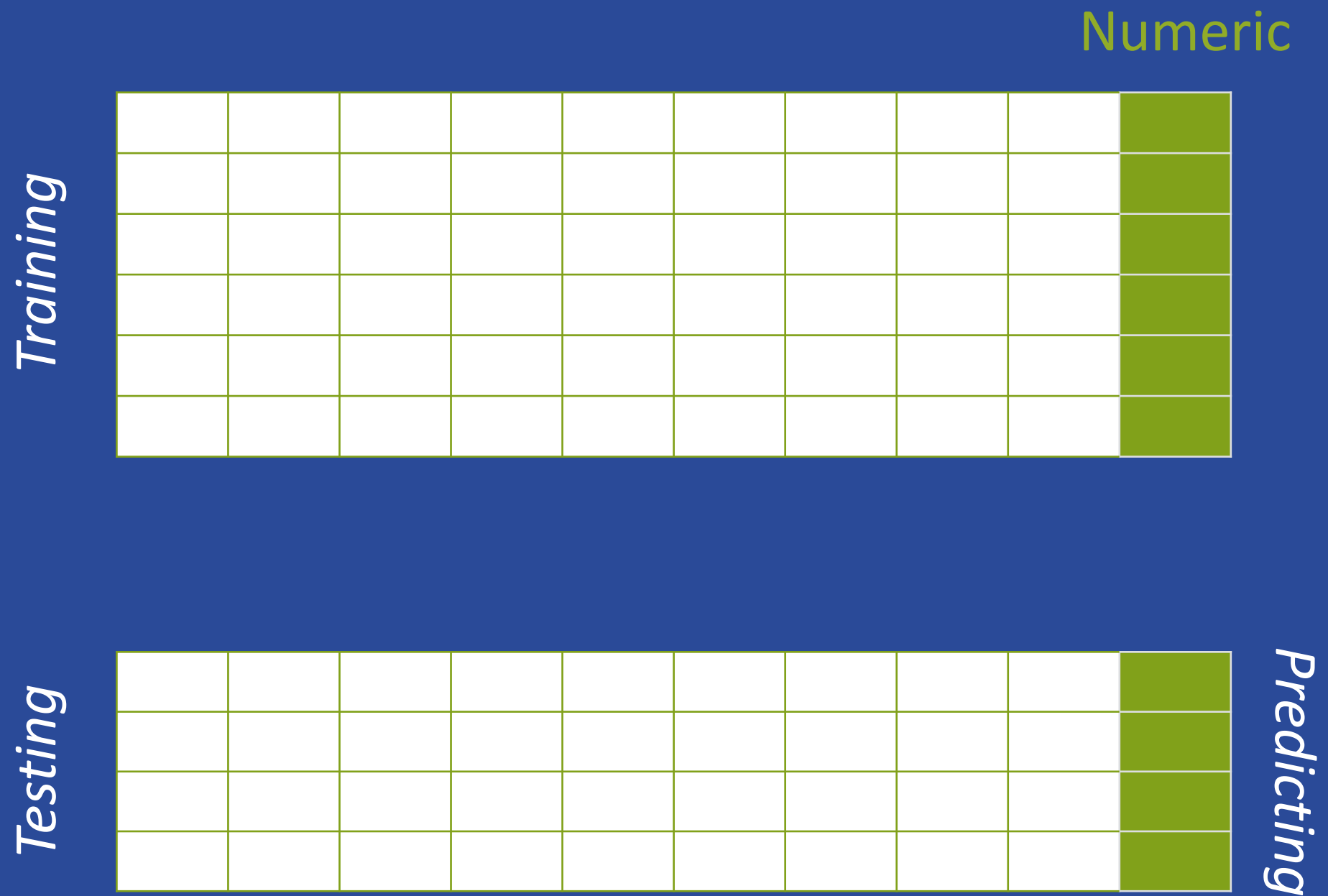
| | |
|---|--|
|  CLASSIFICATION | Will this component fail? |
|  REGRESSION | How many days until this component fails? |
|  TIME SERIES FORECASTING | How many components will fail in a week from now? |
|  CLUSTER ANALYSIS | Which machines behave similarly? |
|  ANOMALY DETECTION | Is this behavior normal? |
|  ASSOCIATION DISCOVERY | What alerts are triggered together before a failure? |

What “shape” is the data for each ML task?

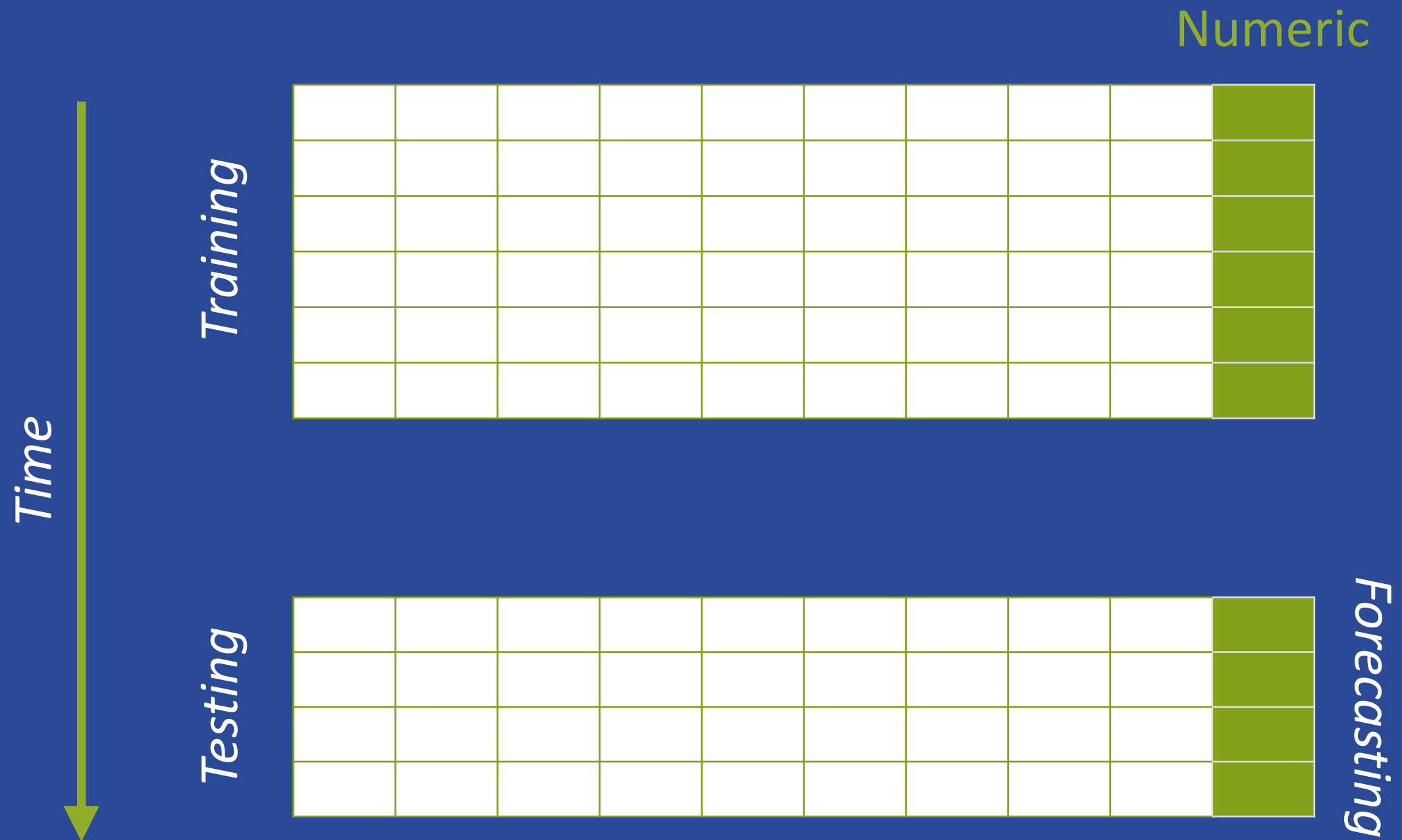
Classification



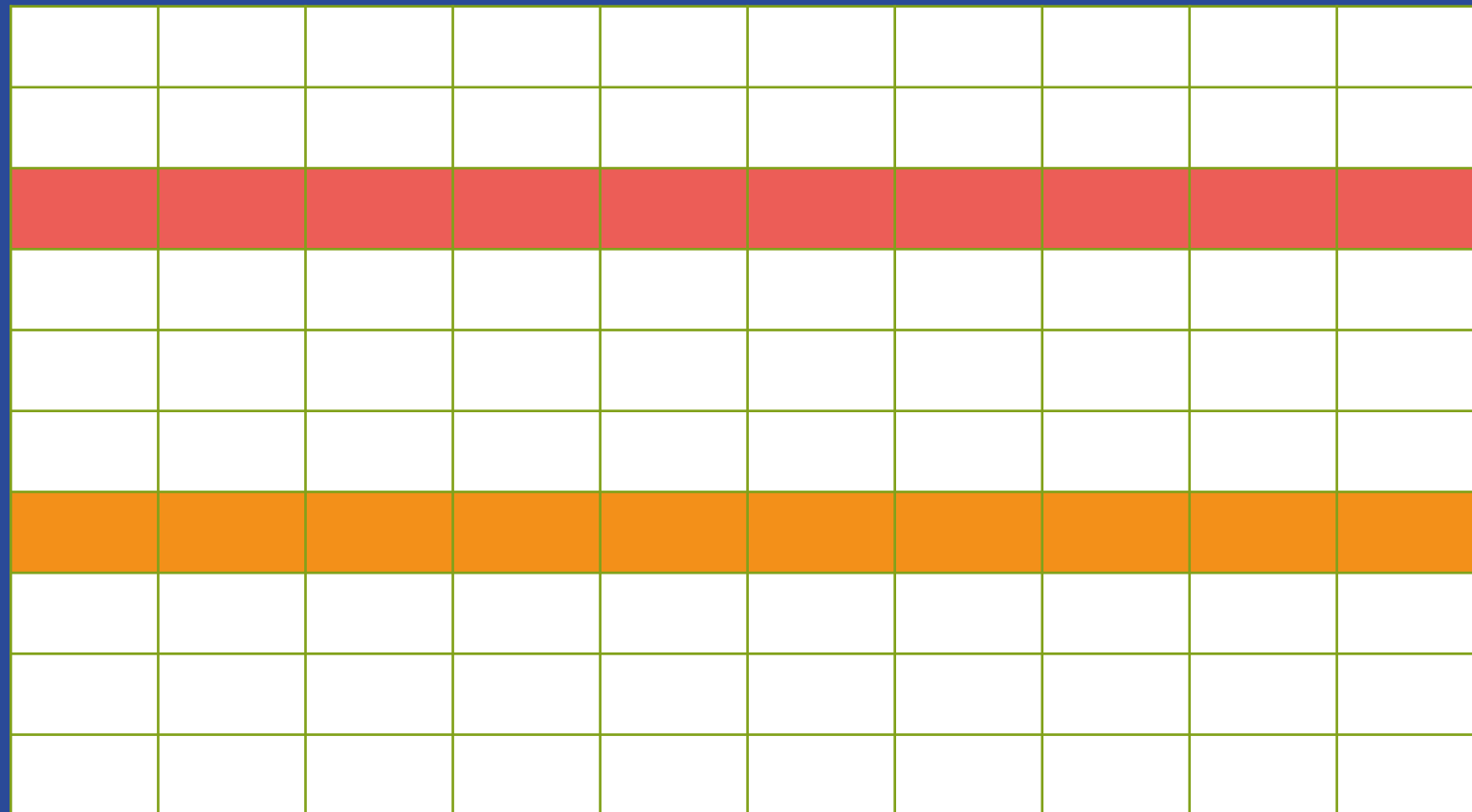
Regression



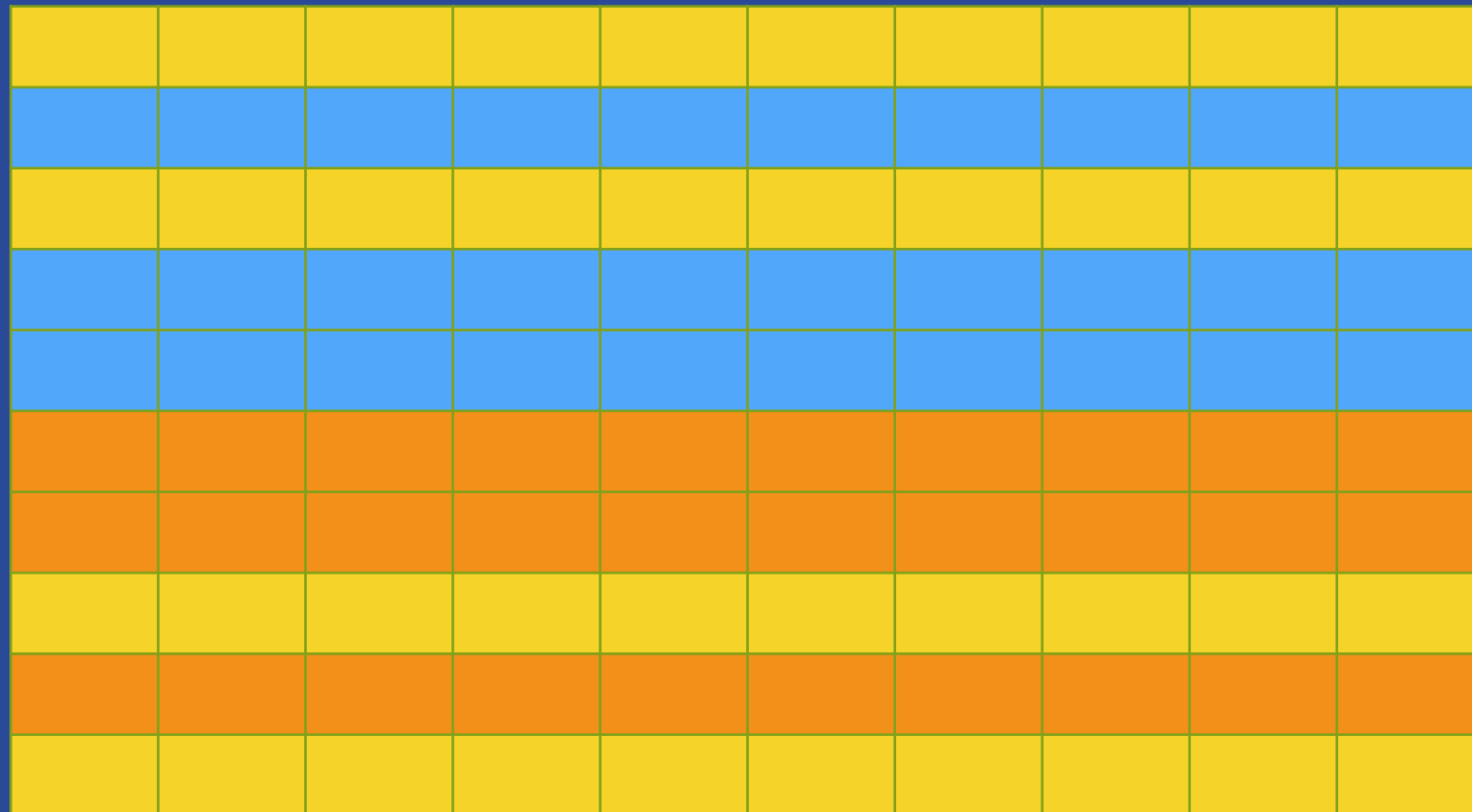
Time Series



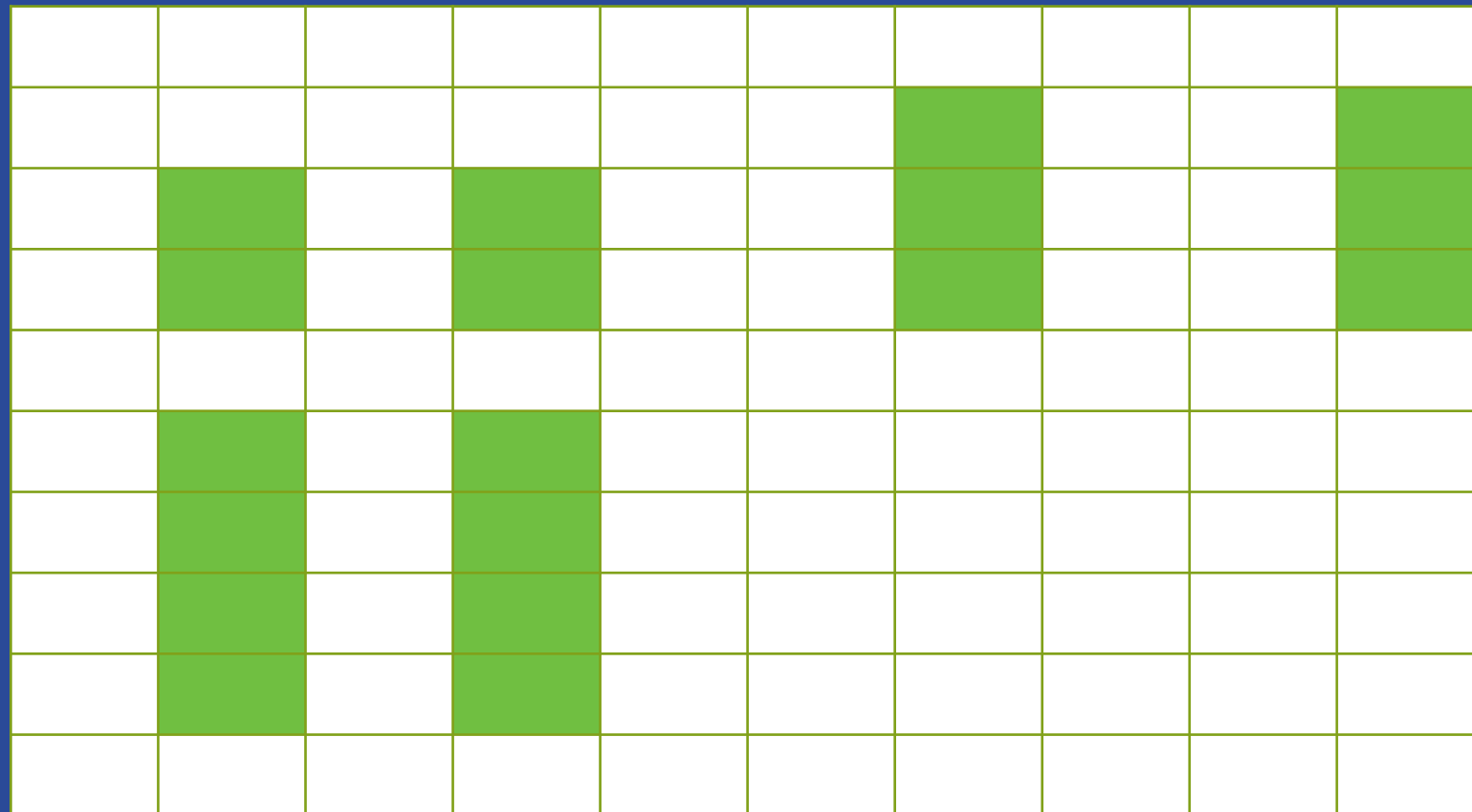
Anomaly Detection



Cluster Analysis



Association Discovery



ML Ready Data



Fields (Features) →

↓ *Instances*

| | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

Tabular Data (rows and columns):

- Each row
 - is one instance
 - contains all the information about that one instance.
 - For Time Series, the rows are **not** independent
- Each column
 - is a field that describes a property of the instance.

Data Transformations

SF Restaurants Data



<https://blog.bigml.com/2013/10/30/data-preparation-for-machine-learning-using-mysql/>

<https://data.sfgov.org/Health-and-Social-Services/Restaurant-Scores/stya-26eb>



```
create database sf_restaurants;  
use sf_restaurants;
```

```
create table businesses (business_id int, name varchar(1000), address varchar(1000), city varchar(1000), state varchar(100),  
postal_code varchar(100), latitude varchar(100), longitude varchar(100), phone_number varchar(100));  
load data local infile './businesses.csv' into table businesses fields terminated by ',' enclosed by '"' lines terminated by  
'\r\n' ignore 1 lines;
```

```
create table inspections (business_id int, score varchar(10), idate varchar(8), itype varchar(100));  
load data local infile './inspections.csv' into table inspections fields terminated by ',' enclosed by '"' lines terminated  
by '\r\n' ignore 1 lines;
```

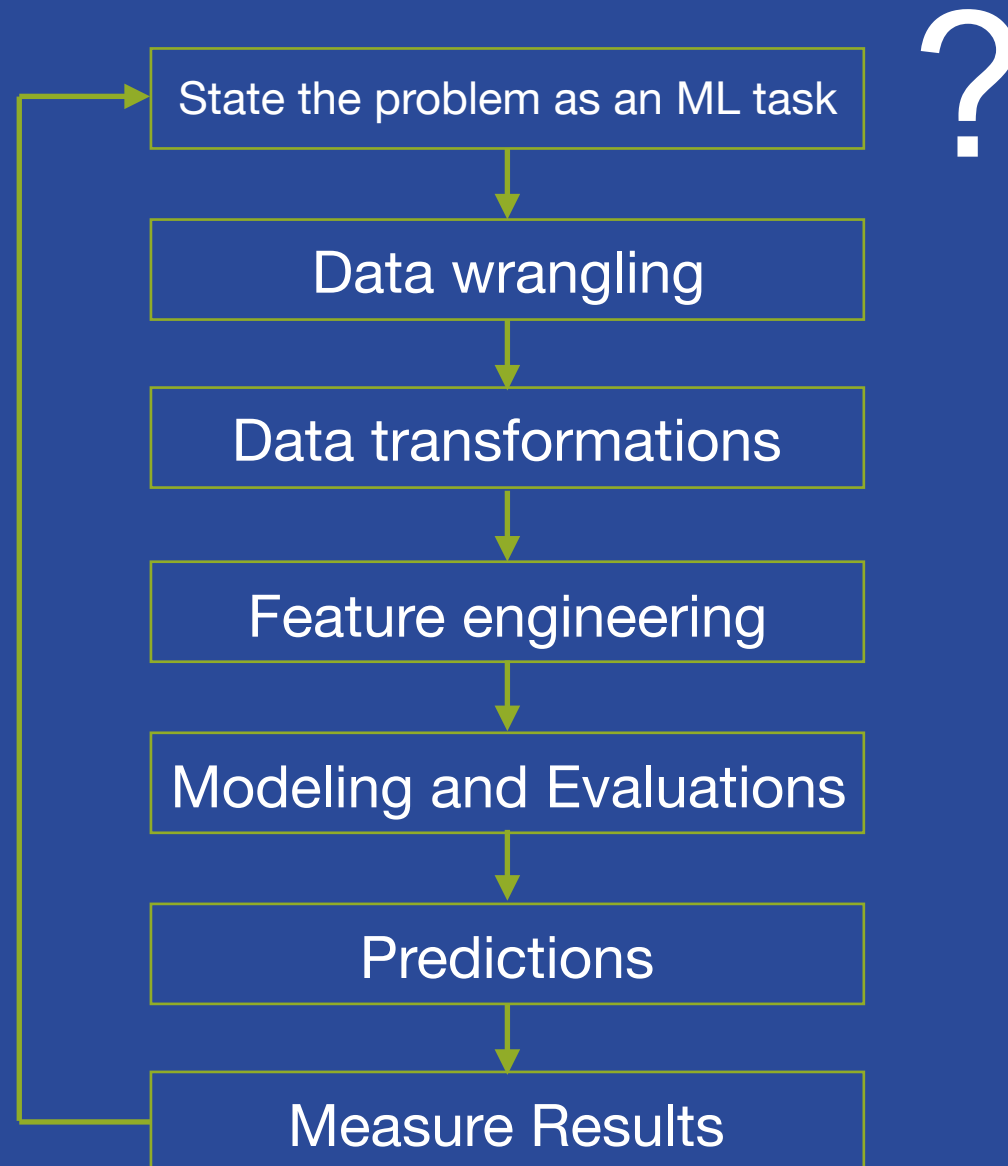
```
create table violations (business_id int, vdate varchar(8), description varchar(1000));  
load data local infile './violations.csv' into table violations fields terminated by ',' enclosed by '"' lines terminated by  
'\r\n' ignore 1 lines;
```

```
create table legend (Minimum_Score int, Maximum_Score int, Description varchar(100));  
load data local infile './legend.csv' into table legend fields terminated by ',' enclosed by '"' lines terminated by '\r\n'  
ignore 1 lines;
```

SF Restaurants Data

Building a ML Application

Task



State the Problem



- Predict rating: Score from 0 to 100
 - This is a **regression** problem
- Based on business profile:
 - Description: kitchen, cafe, etc.
 - Location: zip, latitude, longitude

Problem: Each restaurant may be inspected more than once - which score are we going to use as the label?

Aggregations

Inspections

Instances ↓

| business_id | score | date |
|-------------|-------|----------|
| 10 | 82 | 20160503 |
| 10 | 94 | 20140729 |
| 10 | 92 | 20140114 |
| 19 | 94 | 20160513 |
| 19 | 94 | 20141110 |
| 19 | 94 | 20140214 |
| 24 | 98 | 20161005 |
| 24 | 96 | 20160311 |
| 24 | 96 | 20141124 |
| 24 | 96 | 20140612 |
| 24 | 100 | 20131118 |

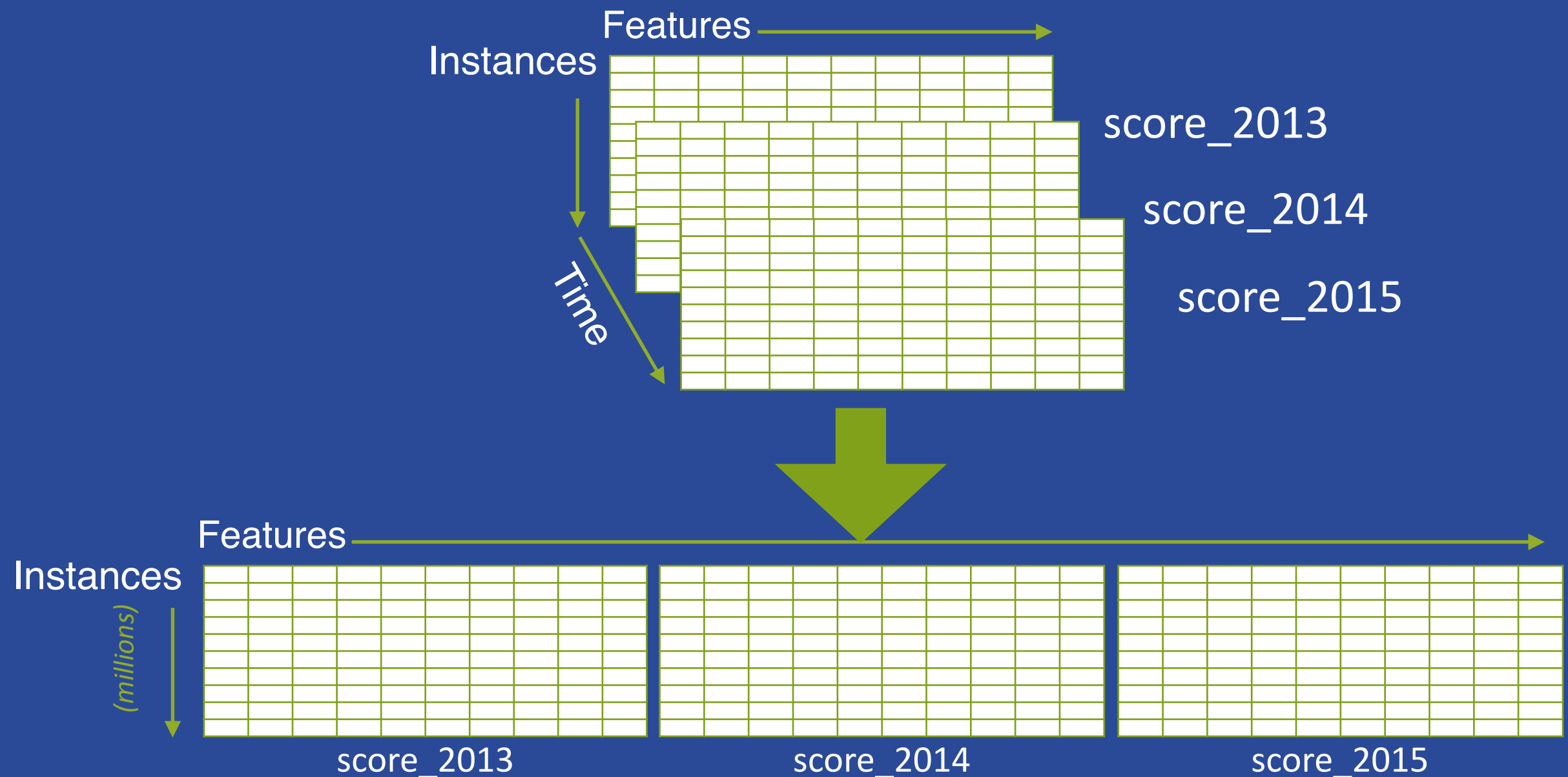
Tabular Data (rows and columns):

- Each row
 - is one instance
 - contains all the information about that one instance.

Aggregations Demo

Time Windows

Rather than aggregate, create new features using values over different periods of time



```
create table scores_2013 select a.business_id, a.score as score_2013, a.idate as idate_2013 from inspections as a JOIN ( select
business_id, max(idate) as idate from inspections where substr(idate,1,4) = "2013" group by business_id) as b where a.business_id =
b.business_id and a.idate = b.idate;
```

```
create table scores_over_time select * from businesses left join scores_2013 USING (business_id) left join scores_2014 USING (business_id);
```

State the Problem



- Predict rating: Score from 0 to 100
 - This is a **regression** problem
- Based on business profile:
 - Description: kitchen, cafe, etc.
 - Location: zip, latitude, longitude

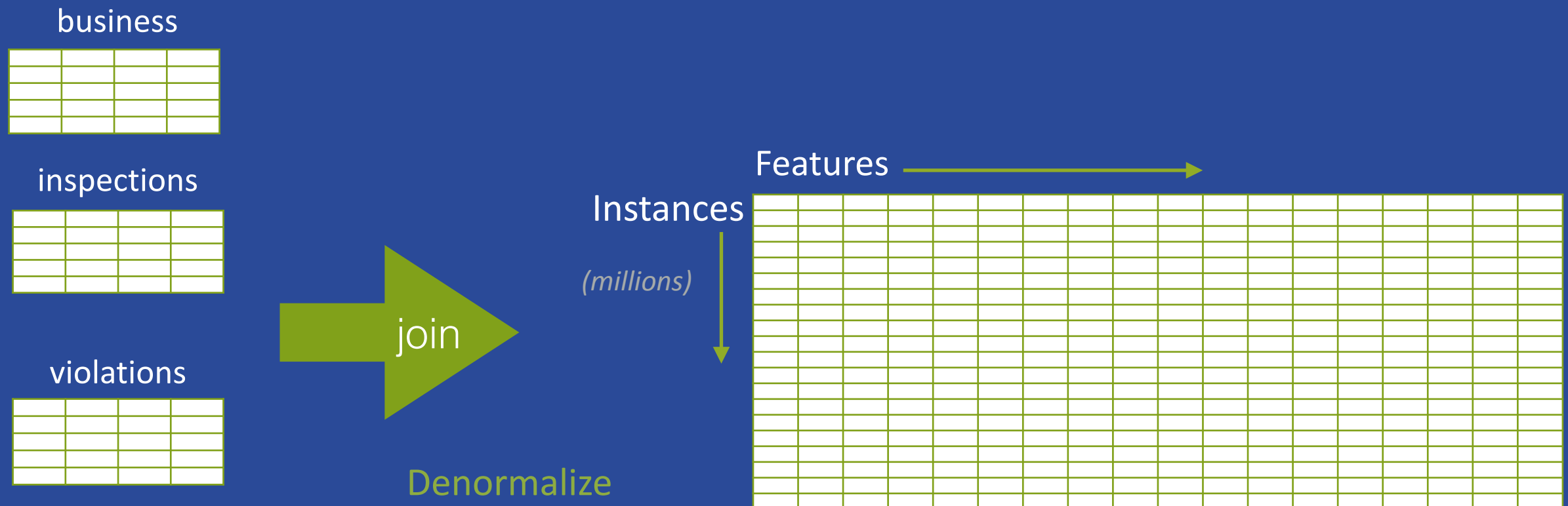
Problem: Each restaurant may be inspected more than once - which score are we going to use as the label?

Solution: Aggregate the score into an avg_score on business_id

Problem: This data is in two datasets: “business” and “avg_score”

Denormalizing with Joins

Data is usually normalized in relational databases, ML-Ready datasets need the information de-normalized in a single dataset.



```
create table scores select * from businesses left join inspections using (business_id);
```

ML-Ready: Each row contains all the information about that one instance.

```
create table scores_last select a.* from scores as a JOIN (select business_id,max(idate)
as idate from scores group by business_id) as b where a.business_id=b.business_id and
a.idate=b.idate;
```

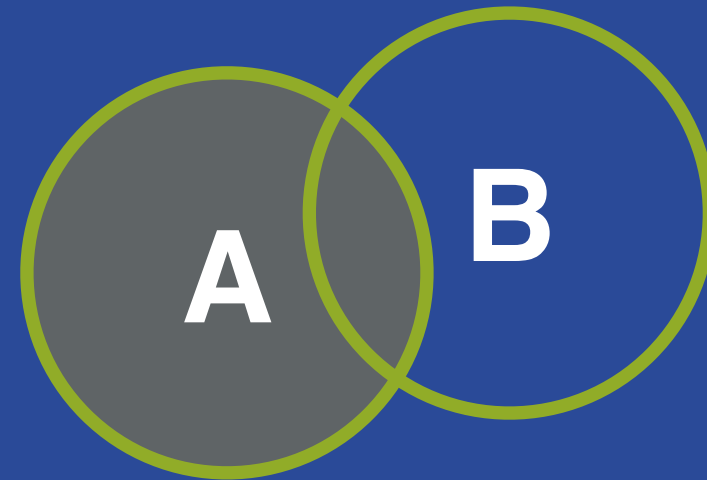
Add Label

```
create table scores_last_label select scores_last.*, Description as score_label from
scores_last join legend on score <= Maximum_Score and score >= Minimum_Score;
```


- **Datasets to join need to have a field in common**
 - joining sales and demographics on **customer_id**
 - joining employee and budget details on **department_id**
- **Datasets to join do not need to have the same dimensions**
- **Joins can be performed in several ways**
 - Left, Right, Inner, Outer...

Left Join

- In a **Left join** of dataset **A** to **B**:
 - Returns **all** records from the left **A**, and the matched records from **B**
 - The result is **NULL** from **B**, if there is no match.



| A | | Left join | B | | = | A left join B | | |
|------------|--------|-----------|------------|--------|---|---------------|--------|--------|
| <u>_id</u> | field1 | | <u>_id</u> | field2 | | <u>_id</u> | field1 | field2 |
| 1 | 34 | | 1 | red | | 1 | 34 | red |
| 2 | 56 | | 2 | green | | 2 | 56 | green |
| 3 | 123 | | 4 | blue | | 3 | 123 | null |
| 4 | 56 | | 6 | black | | 4 | 56 | blue |
| 5 | 79 | | | | | 5 | 79 | null |

No "3" or "5"

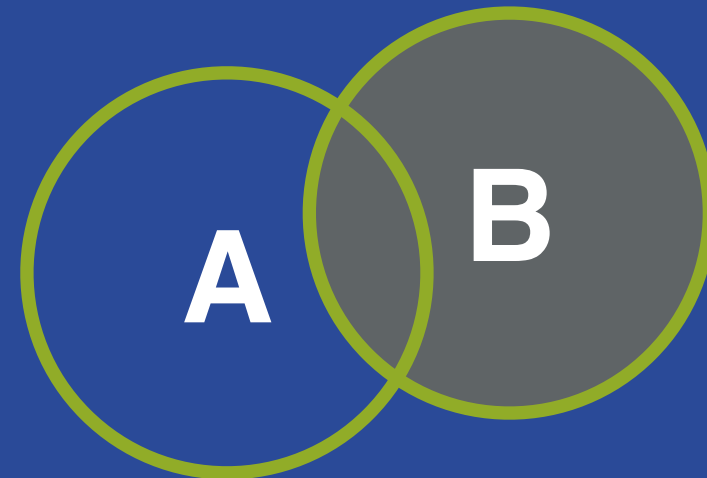
No "3" or "5"

Right Join

- In a Right join of dataset **A** to **B**:

- Returns **all** records from the right **B**, and the matched records from **A**

- The result is NULL from **A**, if there is no match.



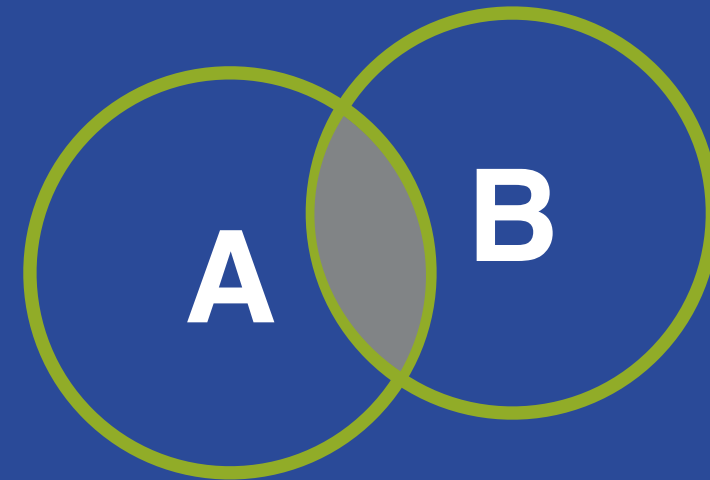
| A | | Right join | B | | = | A right join B | | |
|-----|--------|------------|-----|--------|---|----------------|--------|--------|
| _id | field1 | | _id | field2 | | _id | field2 | field1 |
| 1 | 34 | | 1 | red | | 1 | red | 34 |
| 2 | 56 | | 2 | green | | 2 | green | 56 |
| 3 | 123 | | 4 | blue | | 4 | blue | 56 |
| 4 | 56 | | 6 | black | | 6 | black | null |
| 5 | 79 | | | | | | | |

No "6",
"3" unused

Inner Join

- In an Inner join of dataset **A** to **B**:

- Returns only records from the left **A**, that match records from **B**



- If there is no match between **A** and **B**, the record is ignored

| A | | Inner join | | B | = A inner join B | | |
|-----|--------|------------|--|-----|------------------|--------|--|
| _id | field1 | | | _id | field1 | field2 | |
| 1 | 34 | | | 1 | red | | |
| 2 | 56 | | | 2 | green | | |
| 3 | 123 | | | 4 | blue | | |
| 4 | 56 | | | 6 | black | | |
| 5 | 79 | | | | | | |

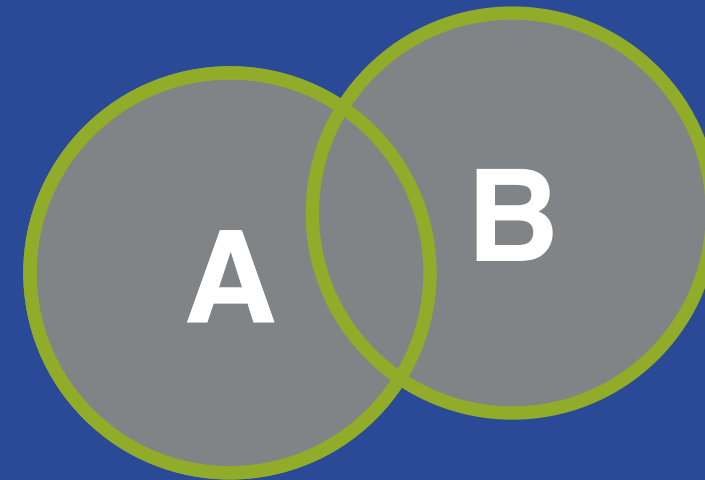
“3” and “5” unused

“6” unused

Full Outer Join

- In a **Full join** of dataset **A** to **B**:

- Returns all records from the left **A**,
and records from **B**



- If there is no match in either **A** and **B**, the field is null

| A | | full join | | B | = A full join B | |
|------------|---------------|------------------|--|------------|------------------------|---------------|
| _id | field1 | | | _id | field1 | field2 |
| 1 | 34 | | | 1 | 34 | red |
| 2 | 56 | | | 2 | 56 | green |
| 3 | 123 | | | 4 | 123 | null |
| 4 | 56 | | | 6 | 56 | blue |
| 5 | 79 | | | | 79 | null |
| | | | | | null | black |

No "6"

No "3" or "5"

Warning: Non-unique IDs

Consider a left join A with B where B has non unique _id entries

| A | | full join | B | | = | A full join B | | |
|-----|--------|-----------|-----|--------|---|---------------|--------|--------|
| _id | field1 | | _id | field2 | | _id | field1 | field2 |
| 1 | 34 | | 1 | red | | 1 | 34 | red |
| 2 | 56 | | 2 | green | | 2 | 56 | green |
| 3 | 123 | | 4 | blue | | 3 | 123 | null |
| 4 | 56 | | 6 | black | | 4 | 56 | blue |
| 5 | 79 | | 4 | green | | 4 | 56 | green |
| | | | | | | 5 | 79 | null |
| | | | | | | 6 | null | black |

Join Demo

State the Problem



- Predict rating: Score from 0 to 100
 - This is a **regression** problem
- Based on business profile:
 - Description: kitchen, cafe, etc.
 - Location: zip, latitude, longitude

Problem: Each restaurant may be inspected more than once - which score are we going to use as the label?

Solution: Aggregate the score into an avg_score on business_id

Problem: This data is in two datasets: “business” and “avg_score”

Solution: Left join “business” with “avg_score” on “business_id”

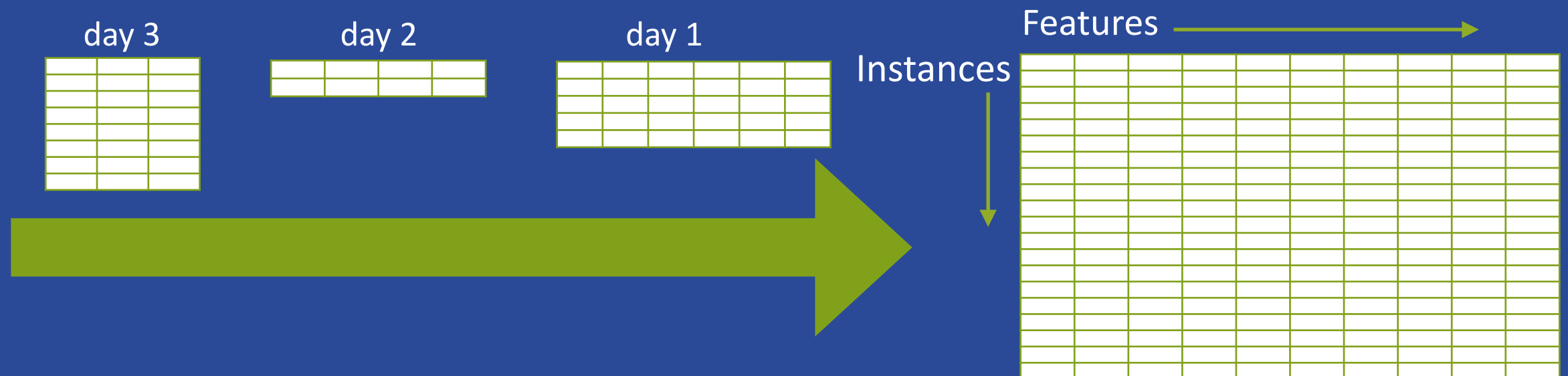
Your Turn!



- Create a Source and Dataset for each of:
 - Businesses
 - Violation
- **Clean** the violation description
 - Hint: `(replace (field "description") "\\[.*" "")`
- **Aggregate** the violations
 - count: violations
 - concat: violation description
- **Join** the business datasets to the violations
- Bonus:
 - Join last_inspections and legend
 - build a model which predicts the score legend

Updates

Need a current view of the data, but new data only comes in batches of changes



Streaming

Data only comes in single changes

