Before we begin, visit https://github.com/radanalyticsio/workshop to download images and code!

Insightful Apps with Apache Spark and OpenShift

William Benton (@willb)

Michael McCune (@FOSSJunkie)

Forecast

Introducing insightful apps

Learning from data

Meet Apache Spark

Hands-on: data engineering and machine learning in Spark and building an insightful application in OpenShift

Preliminaries

Make sure you have OpenShift Origin installed (if you want to build an app) or at least Docker (if you just want to try out Apache Spark)

Pull all of the necessary images for the hands-on portion

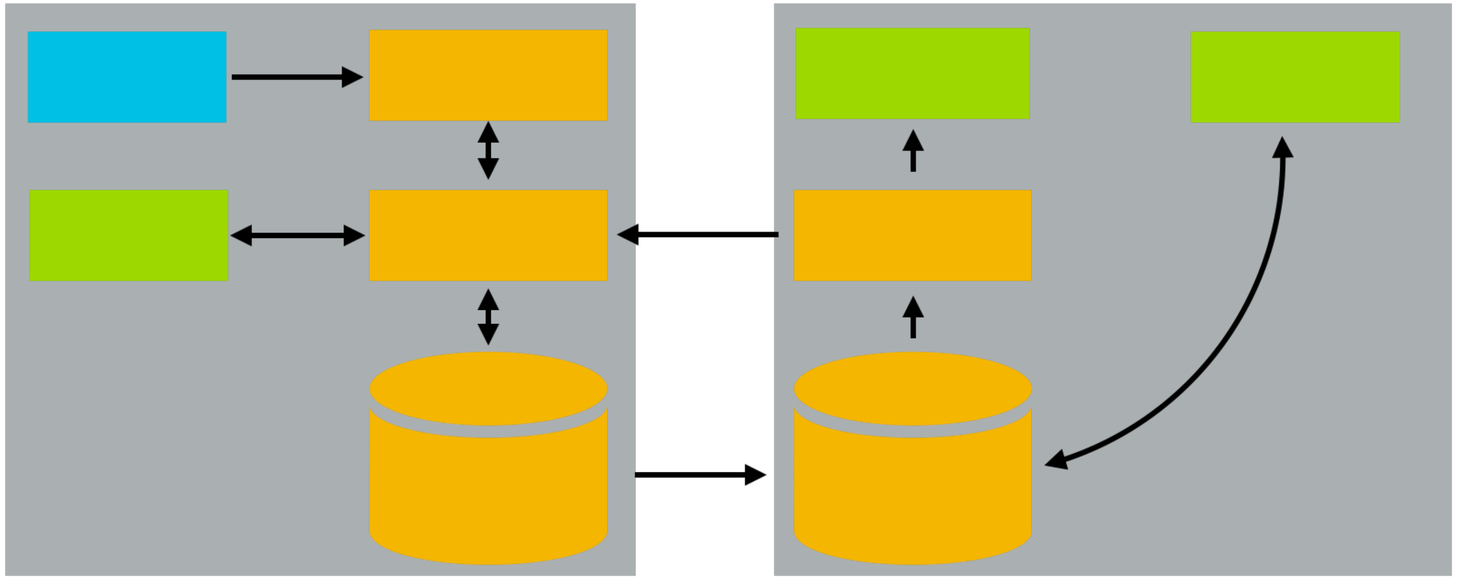
Details here: https://github.com/radanalyticsio/workshop

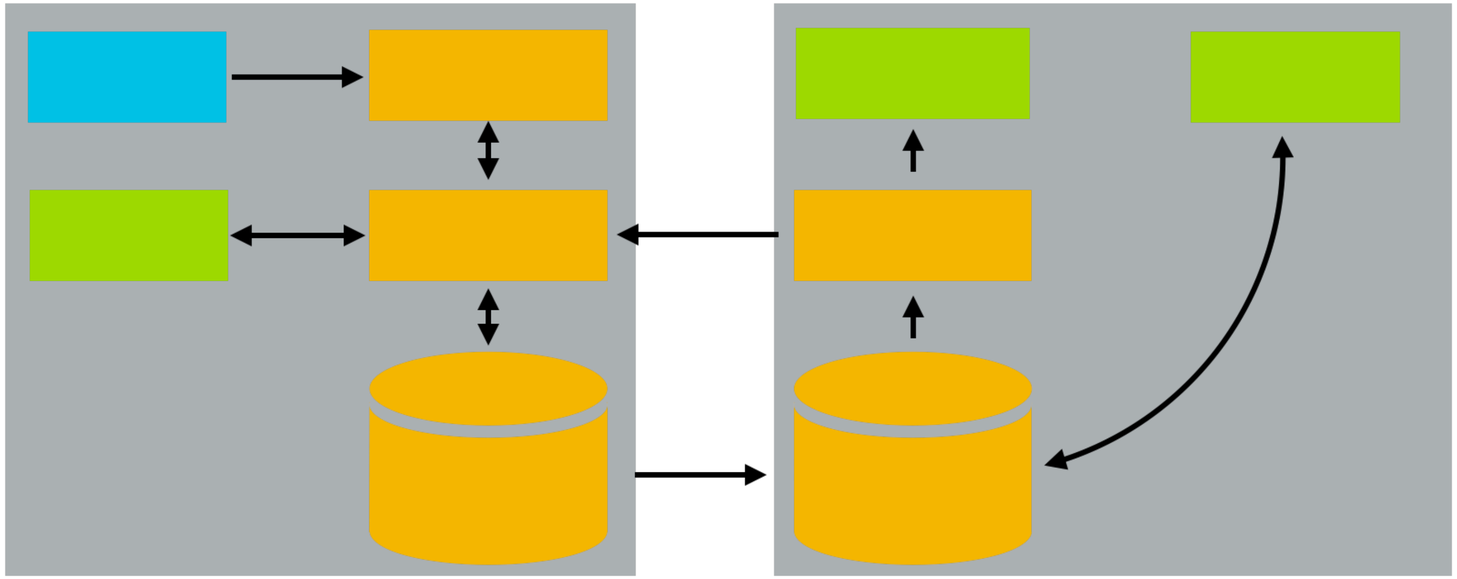
Introducing insightful apps

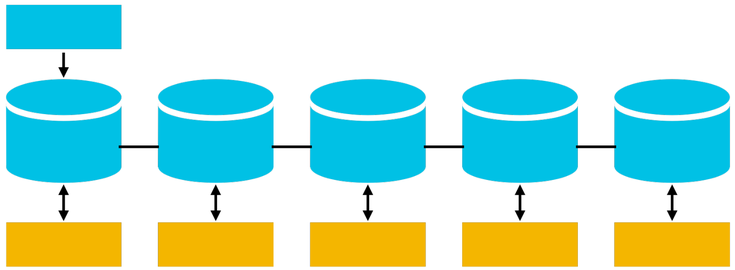
Insightful applications

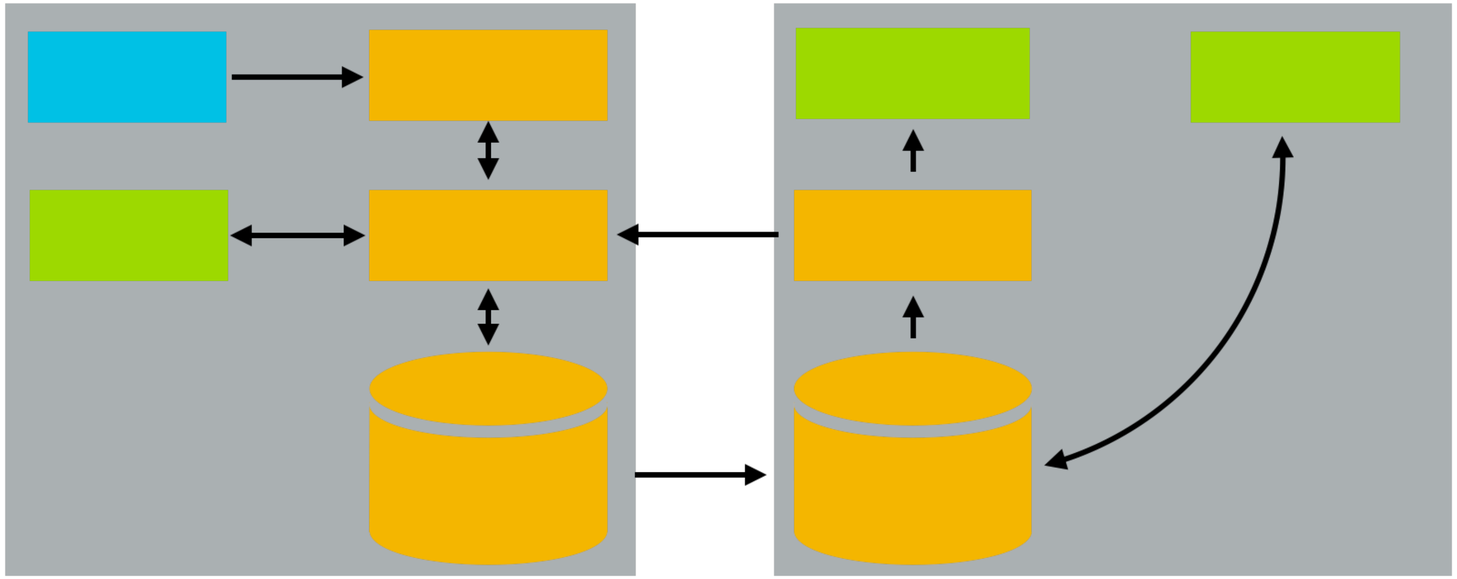
Insightful applications **collect** and **learn from data** that users **generate** and **provide** in order to **work better with longevity and popularity.**

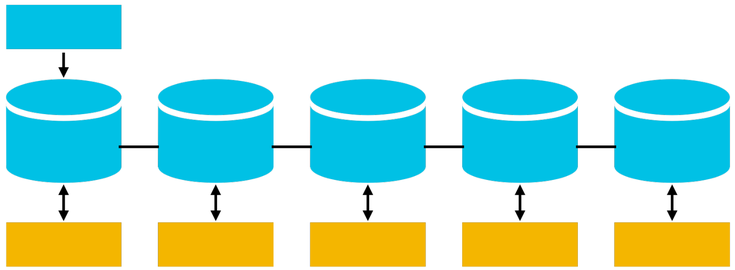
Almost every **exciting** or **important contemporary app** is **insightful**!

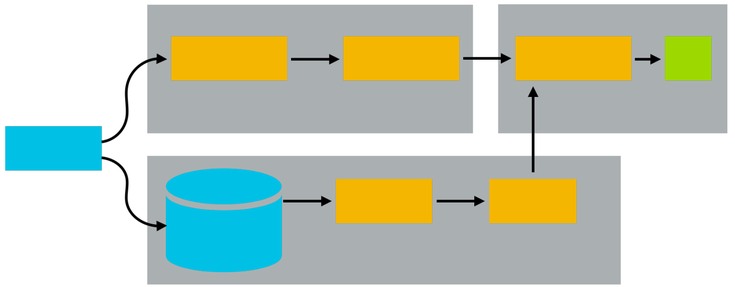


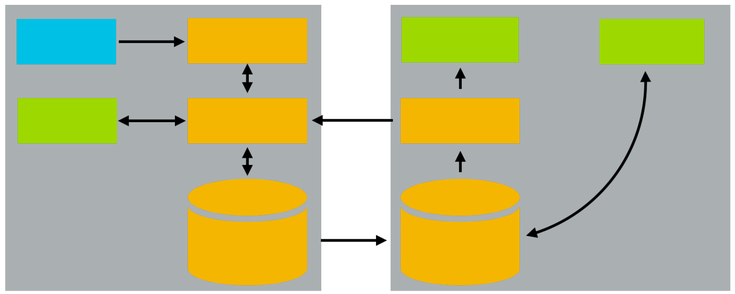


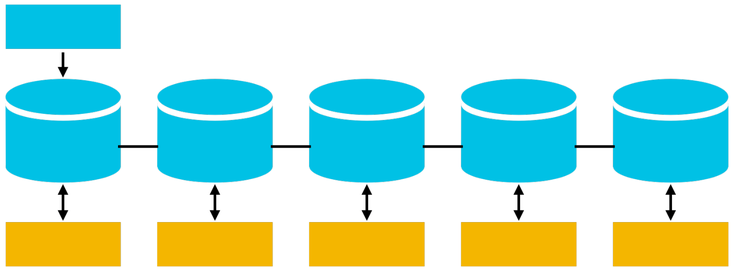


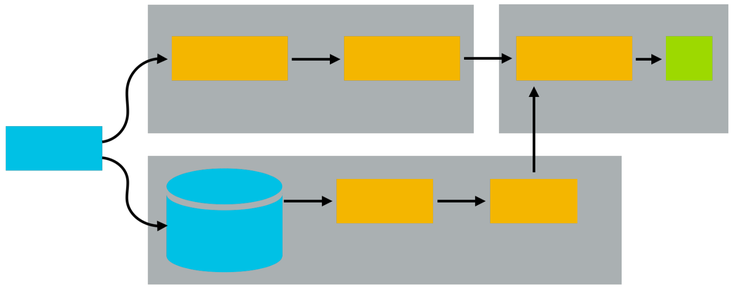


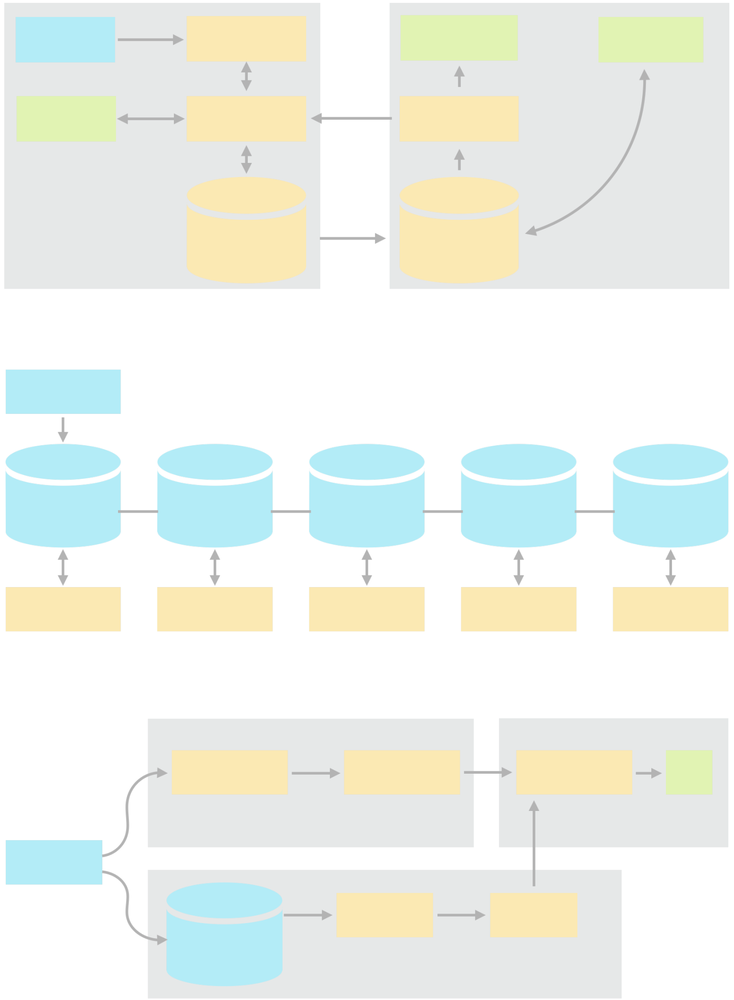


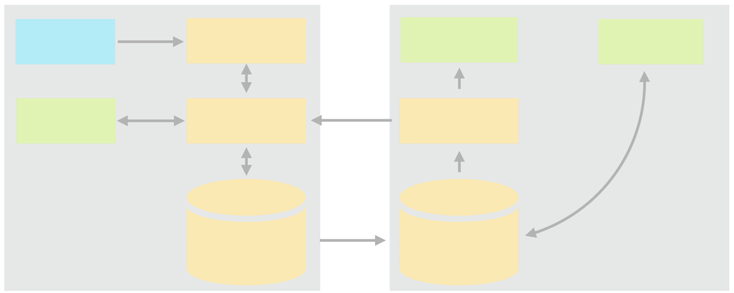


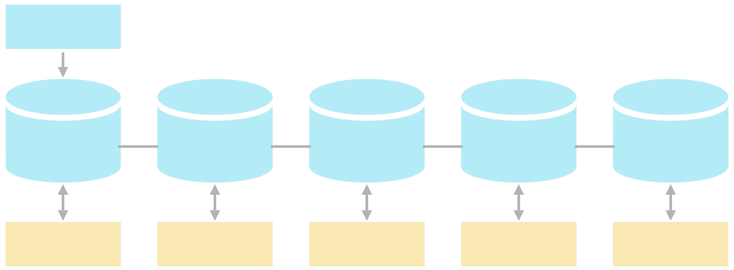


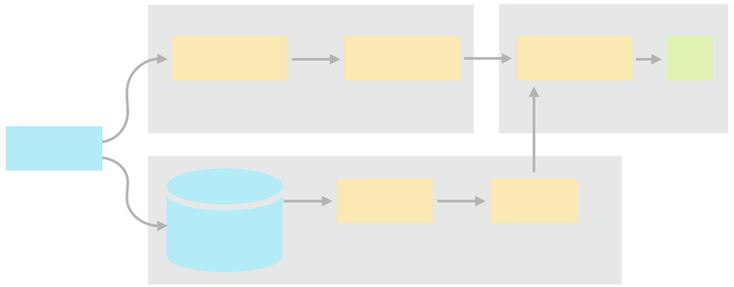


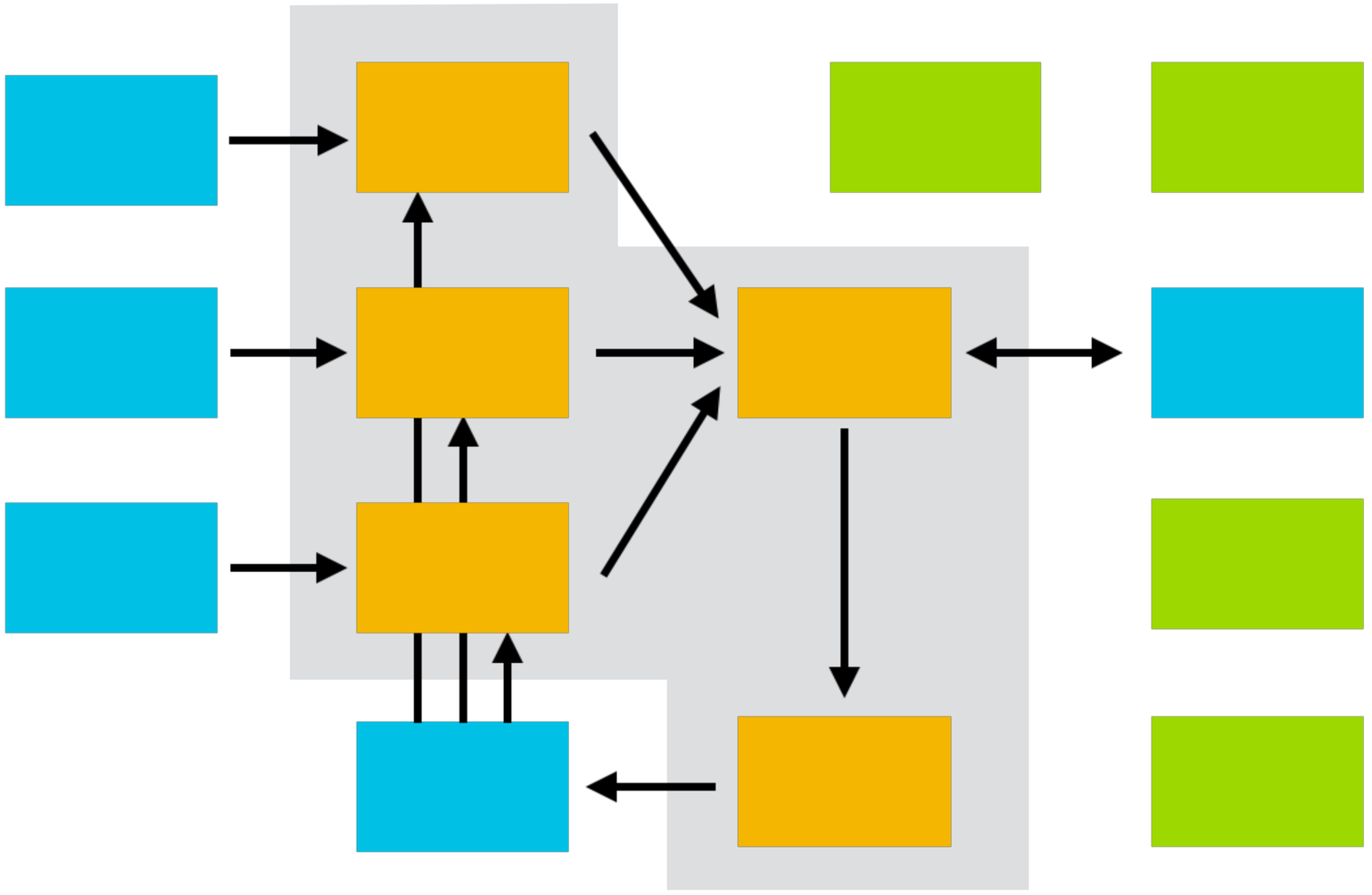










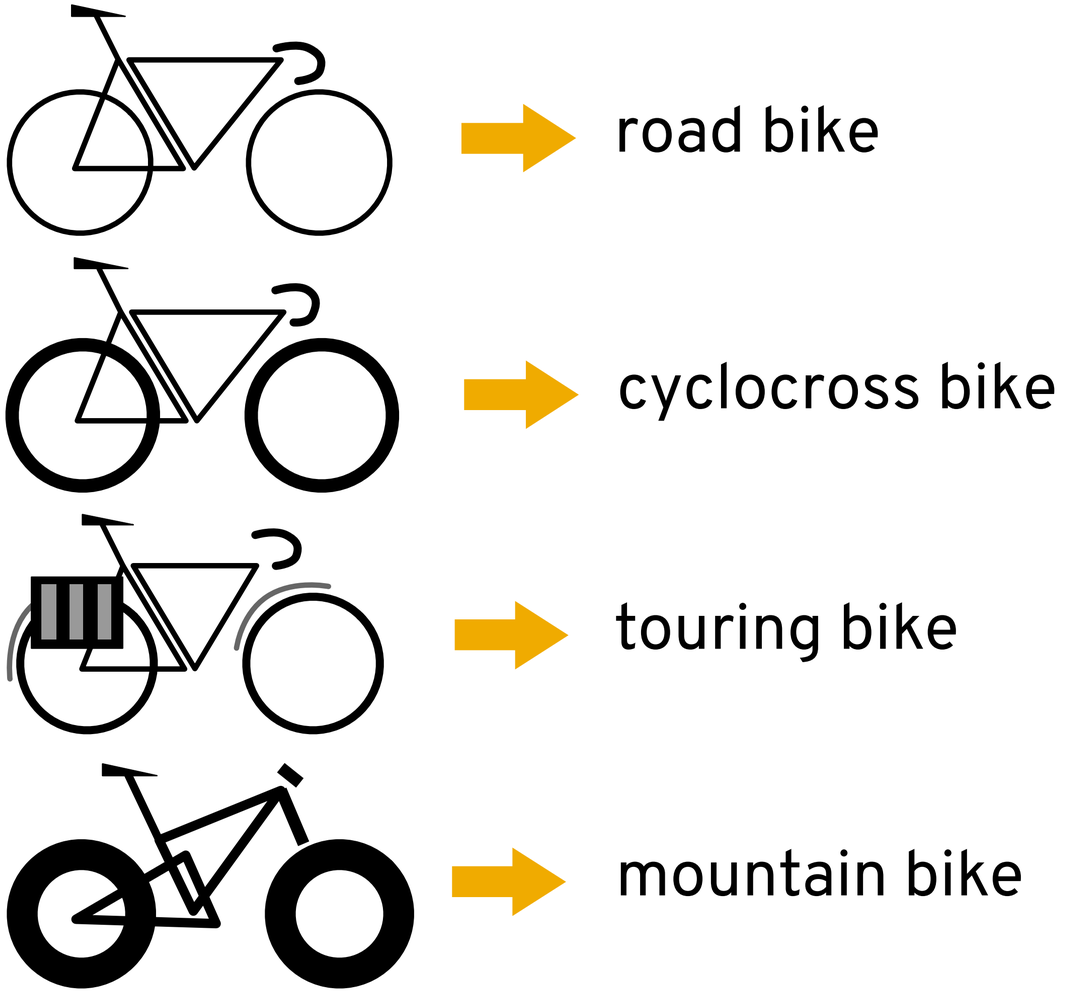


Learning from data

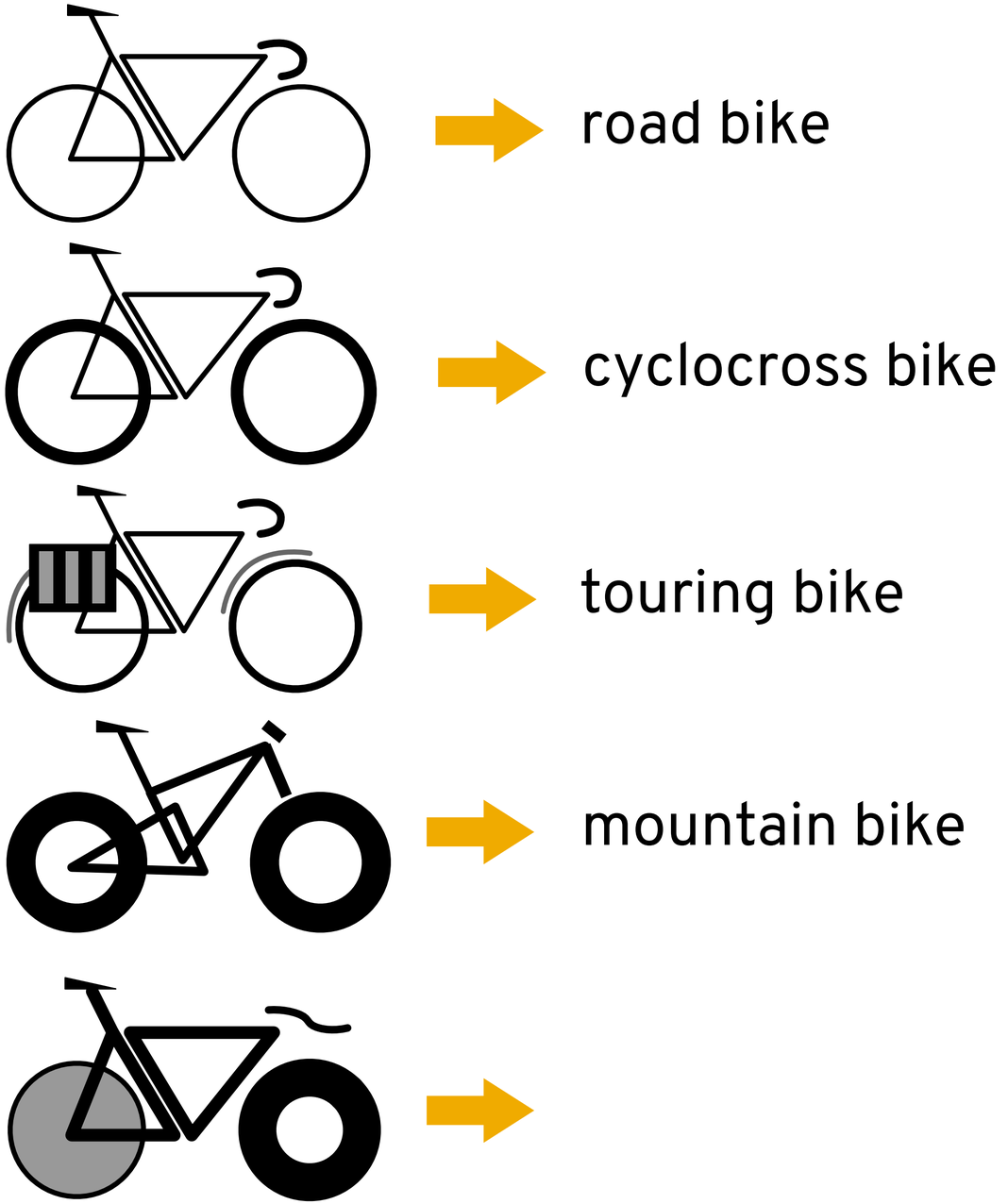
BASIC CONCEPTS

```
def classify(bike):
  if bar_type(bike) == "flat":
    if tire_width(bike) > 80:
      return "winter bike"
    if tire_width(bike) > 50 or has_suspension(bike):
      return "mountain bike"
    if frame_type(bike) == "step-through":
      return "city bike"
  elif bar_type(bike) == "drop":
    if tire_width(bike) <= 27:
      return "road bike"
    if tire_type(bike) == "knobby":
      return "cyclocross bike"
    return "touring bike"
  return "unknown bike"
```

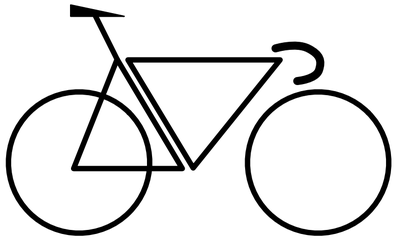
```
def classify(bike):
  if bar_type(bike) == "flat":
    if tire_width(bike) > 80:
      return "winter bike"
    if tire_width(bike) > 50 or has_suspension(bike):
      return "mountain bike"
    if frame_type(bike) == "step-through":
      return "city bike"
  elif bar_type(bike) == "drop":
    if tire_width(bike) <= 27:
      return "road bike"
    if tire_type(bike) == "knobby":
      return "cyclocross bike"
    return "touring bike"
  return "unknown bike"
```



```
def classify(bike):
  if bar_type(bike) == "flat":
    if tire_width(bike) > 80:
      return "winter bike"
    if tire_width(bike) > 50 or has_suspension(bike):
      return "mountain bike"
    if frame_type(bike) == "step-through":
      return "city bike"
  elif bar_type(bike) == "drop":
    if tire_width(bike) <= 27:
      return "road bike"
    if tire_type(bike) == "knobby":
      return "cyclocross bike"
    return "touring bike"
  return "unknown bike"
```

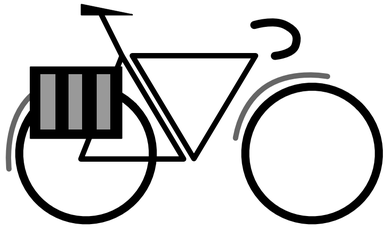


```
def classify(bike):
  if bar_type(bike) == "flat":
    if tire_width(bike) > 80:
      return "winter bike"
    if tire_width(bike) > 50 or has_suspension(bike):
      return "mountain bike"
    if frame_type(bike) == "step-through":
      return "city bike"
  elif bar_type(bike) == "drop":
    if tire_width(bike) <= 27:
      return "road bike"
    if tire_type(bike) == "knobby":
      return "cyclocross bike"
    return "touring bike"
  return "unknown bike"
```

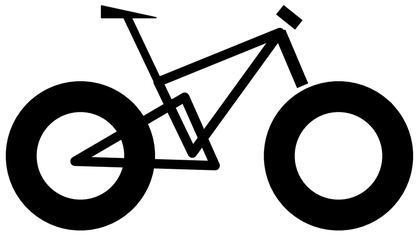


road bike

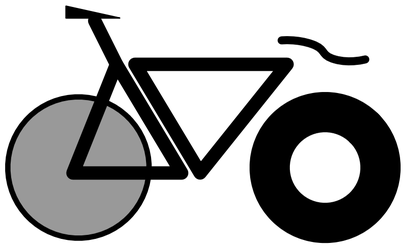
cyclocross bike



touring bike

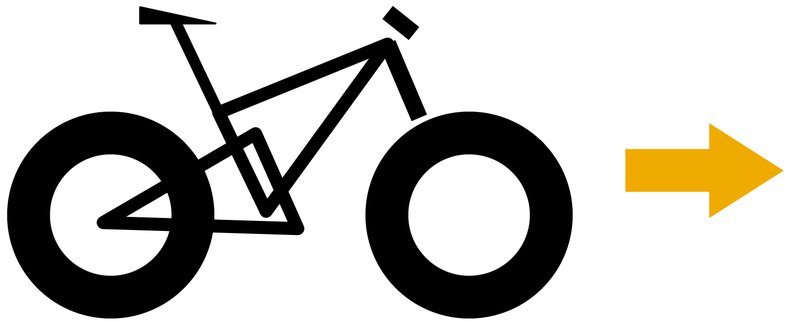


mountain bike

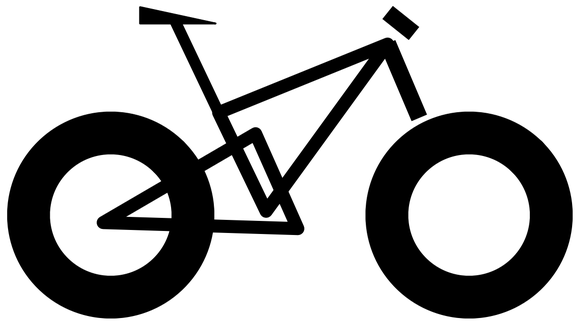


triathlon bike

Feature engineering

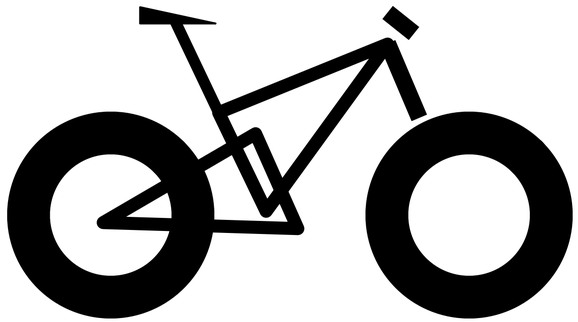


Feature engineering



mountain bike	0	1	60	1	1	1

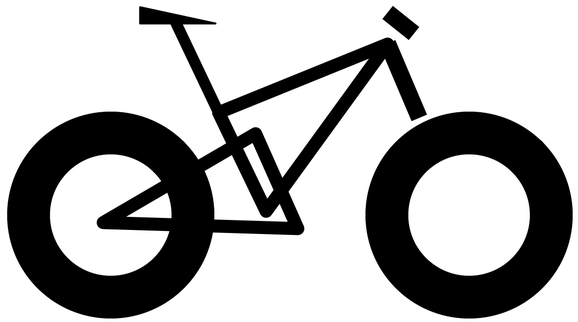
Feature engineering



mountain bike	0	1	60	1	1	1

LABEL

Feature engineering

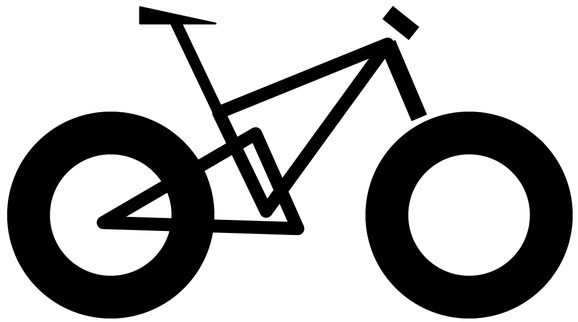


mountain bike	0	1	60	1	1	1

LABEL

HANDLEBAR TYPE

Feature engineering



mountain bike	0	1	60	1	1	1

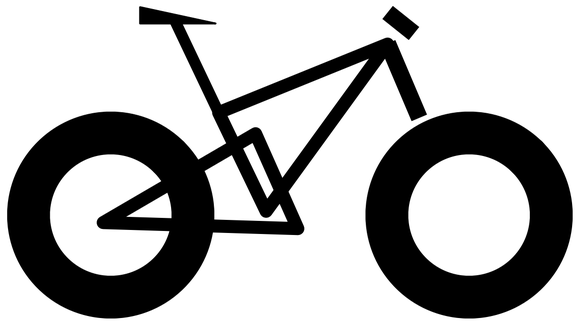
LABEL

DROP

FLAT

HANDLEBAR TYPE

Feature engineering



mountain bike	0	1	60	1	1	1

LABEL

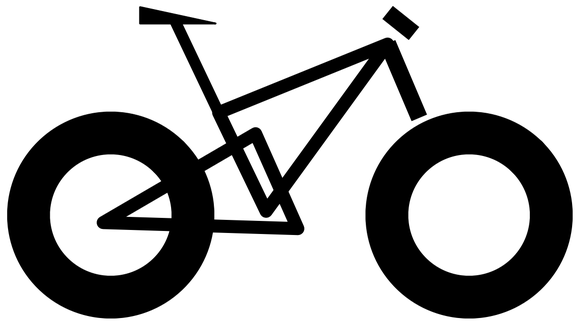
DROP

FLAT

HANDLEBAR TYPE

TIRE SIZE

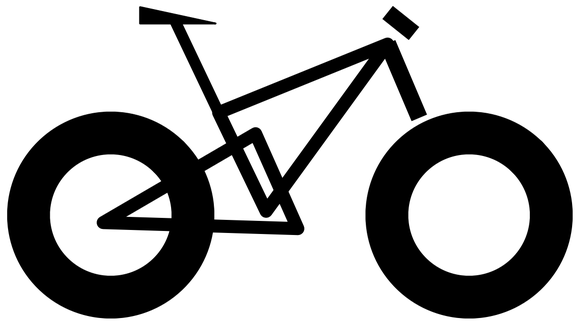
Feature engineering



mountain bike	0	1	60	1	1	1

LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS
	HANDLEBAR TYPE			

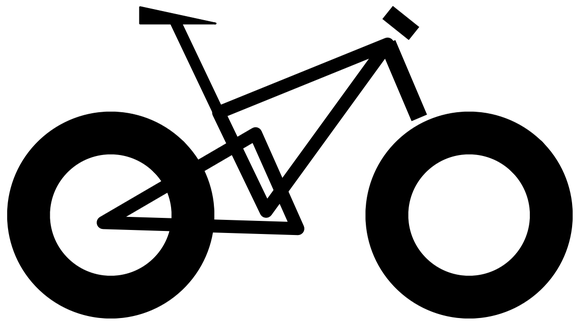
Feature engineering



mountain bike	0	1	60	1	1	1

LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS	SUSPENSION?
	HANDLEBAR TYPE				

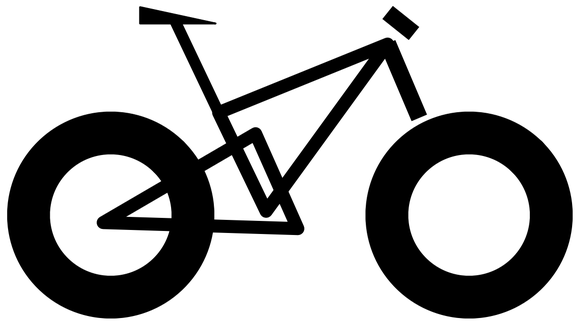
Feature engineering

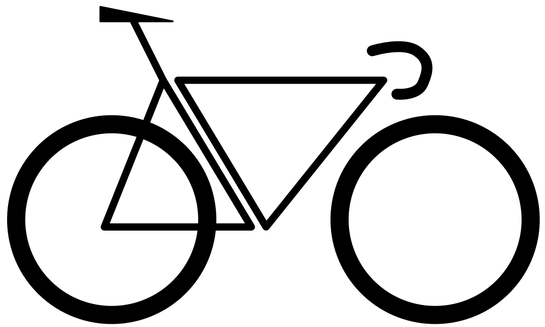


mountain bike	0	1	60	1	1	1

LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS	FRONT	REAR
	HANDLEBAR TYPE				SUSPENSION?	

Feature engineering

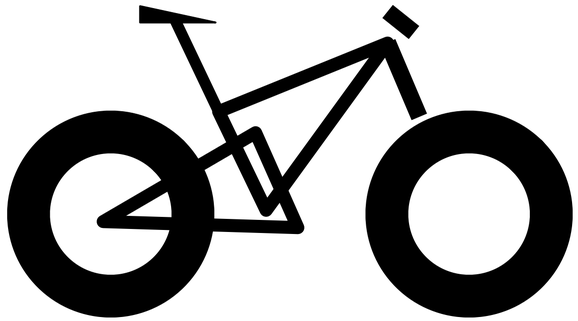


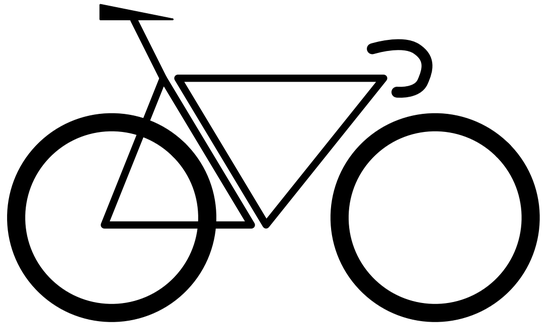


mountain bike	0	1	60	1	1	1

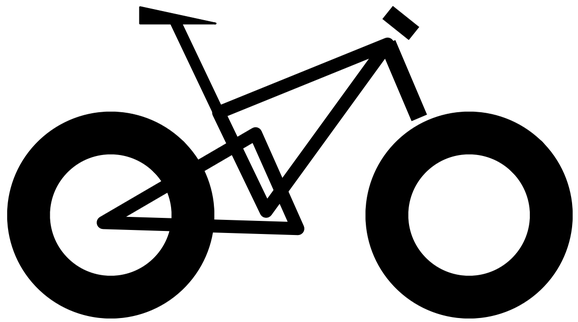
LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS	FRONT	REAR
	HANDLEBAR TYPE				SUSPENSION?	

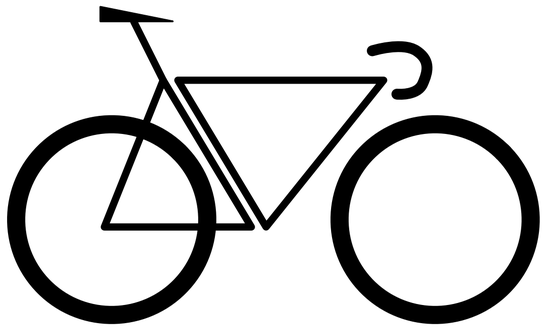
Feature engineering





LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS	FRONT	REAR
	HANDLEBAR TYPE				SUSPENSION?	
mountain bike	0	1	60	1	1	1
cyclocross bike	1	0	33	1	0	0

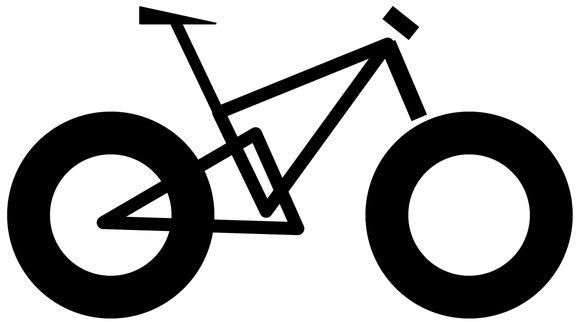


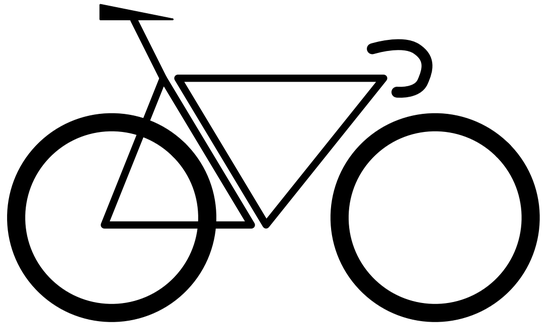


one-hot encoding

LABEL	DROP	FLAT	HANDLEBAR TYPE	TIRE SIZE	TIRE KNOBS	FRONT	REAR	SUSPENSION?
mountain bike	0	1	60	1	1	1		
cyclocross bike	1	0	33	1	0	0		

(convert from a categorical feature with *n* values to an *n*-bit vector)



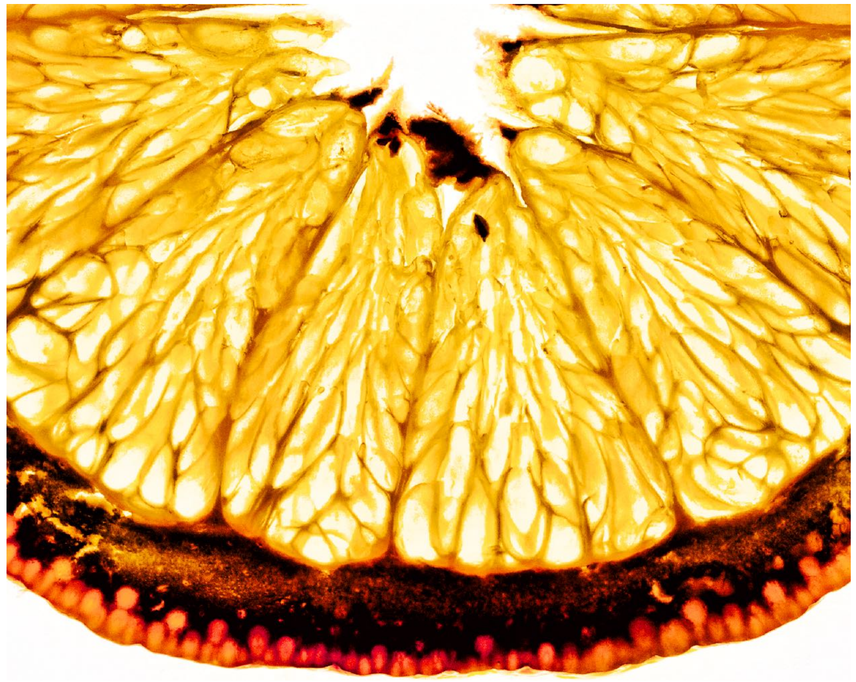


value scaling

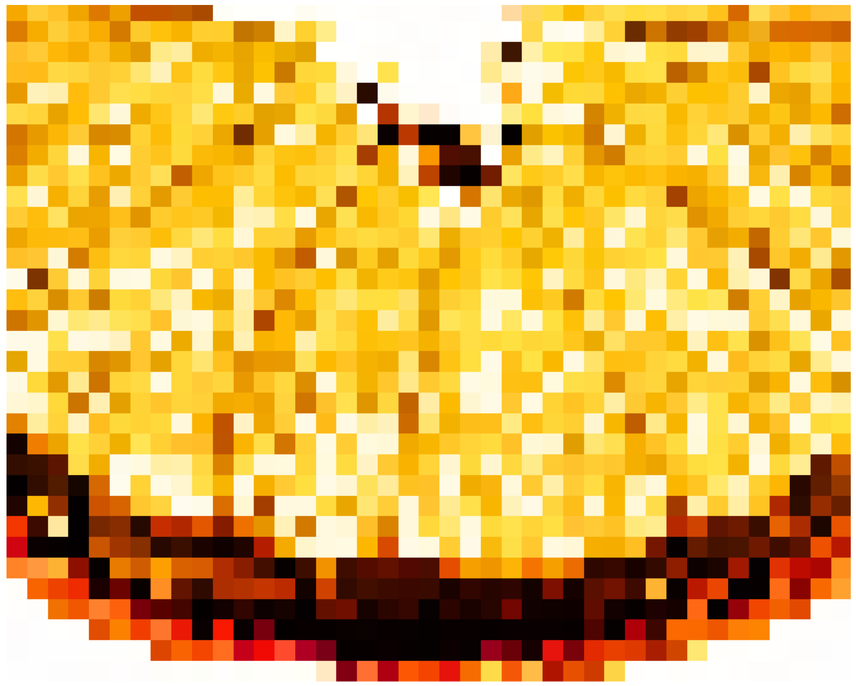
LABEL	DROP / HANDLEBAR TYPE	FLAT / HANDLEBAR TYPE	TIRE SIZE	TIRE KNOBS	FRONT SUSPENSION?	REAR SUSPENSION?
mountain bike	0	1	0.35	1	1	1
cyclocross bike	1	0	0.13	1	0	0

(assuming that all tires are between 19mm and 130mm wide)

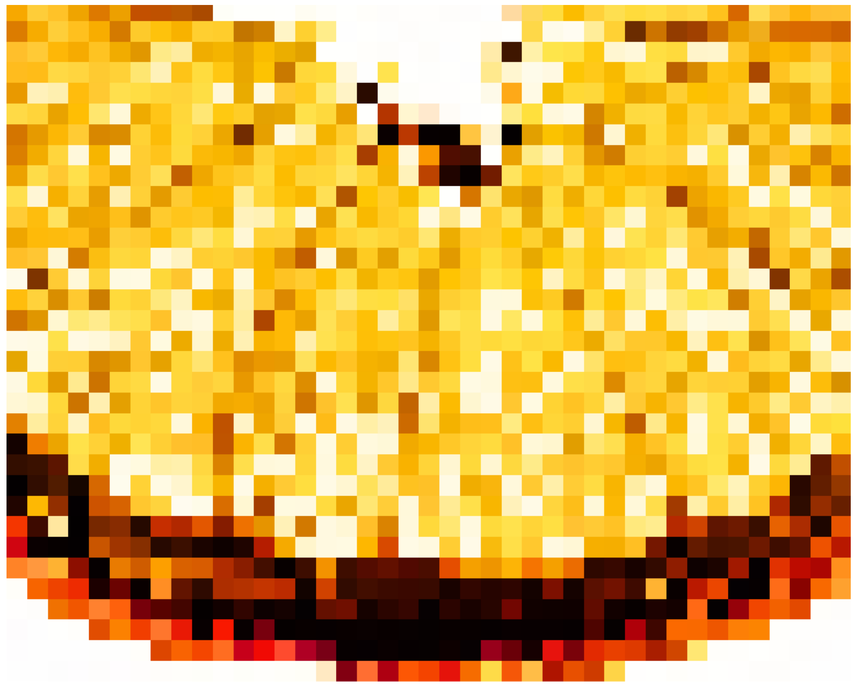
Approximation techniques

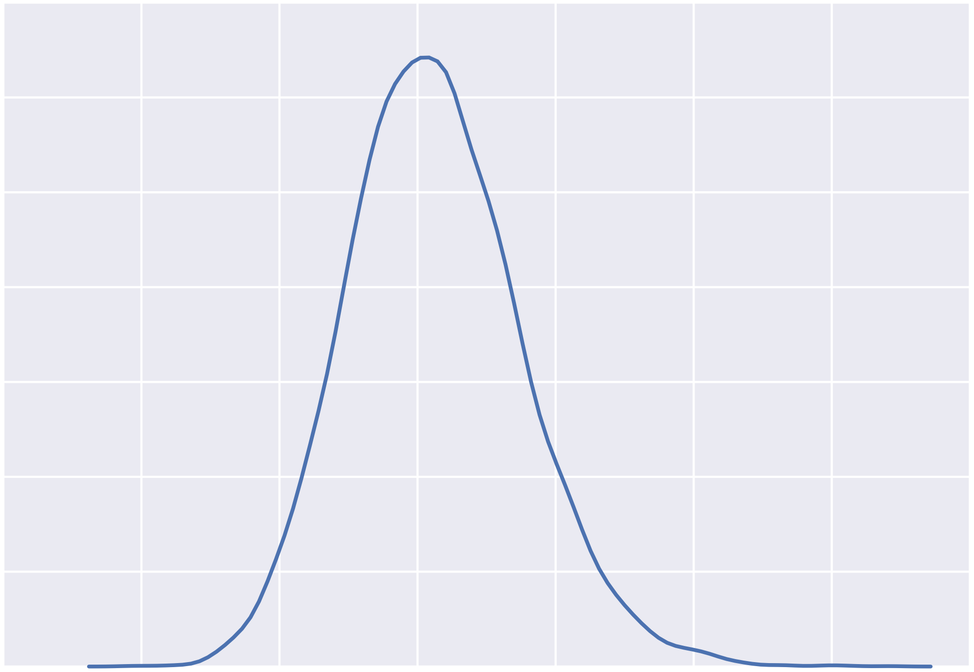


Approximation techniques

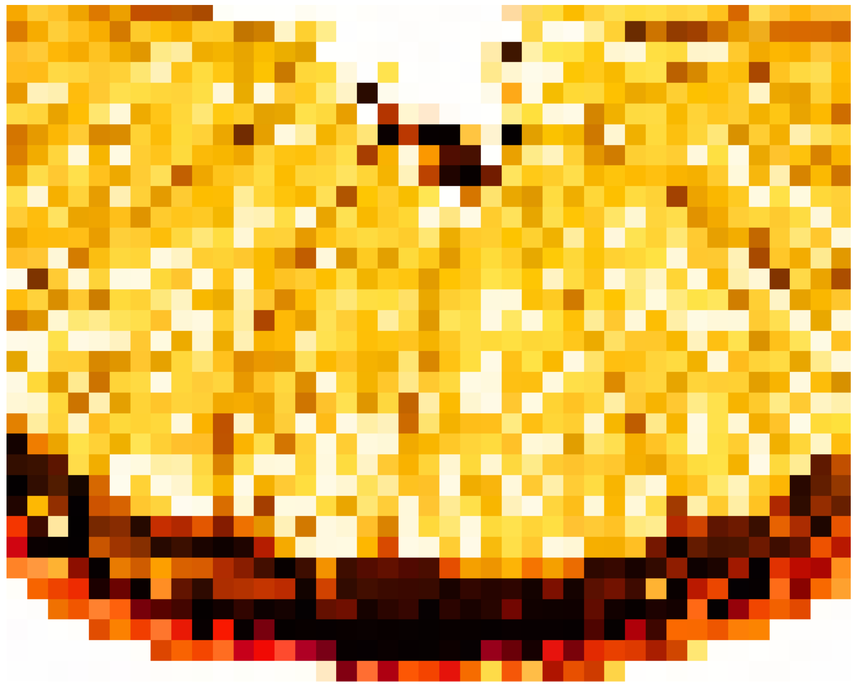


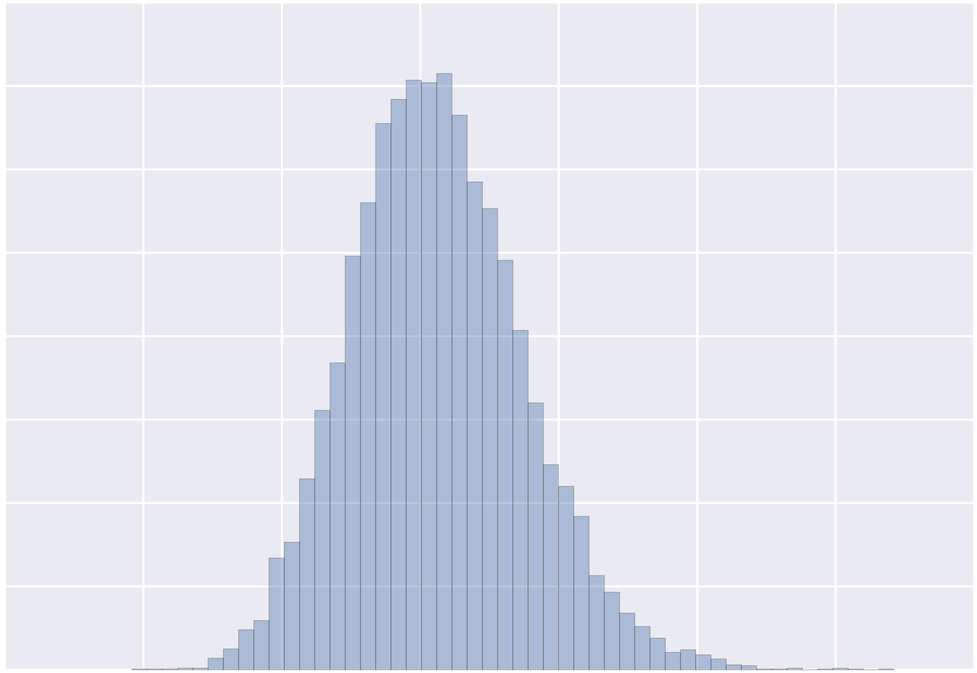
Approximation techniques



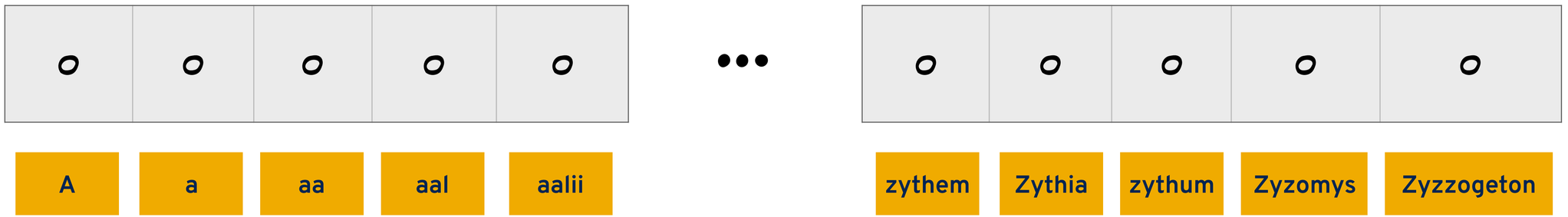


Approximation techniques





Feature hashing

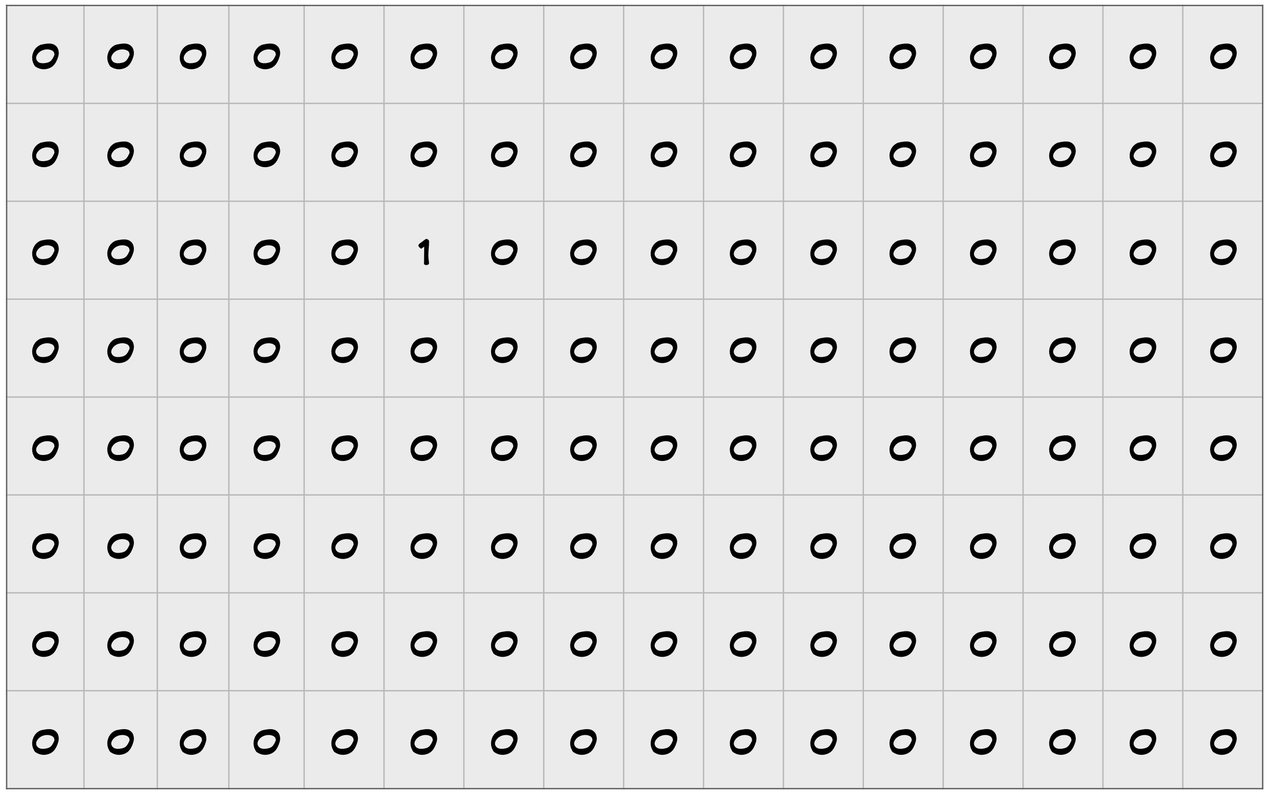


Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```

Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```



"the" → (37, 1)

Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```

"the" → (37, 1)

"quick" → (121, -1)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0

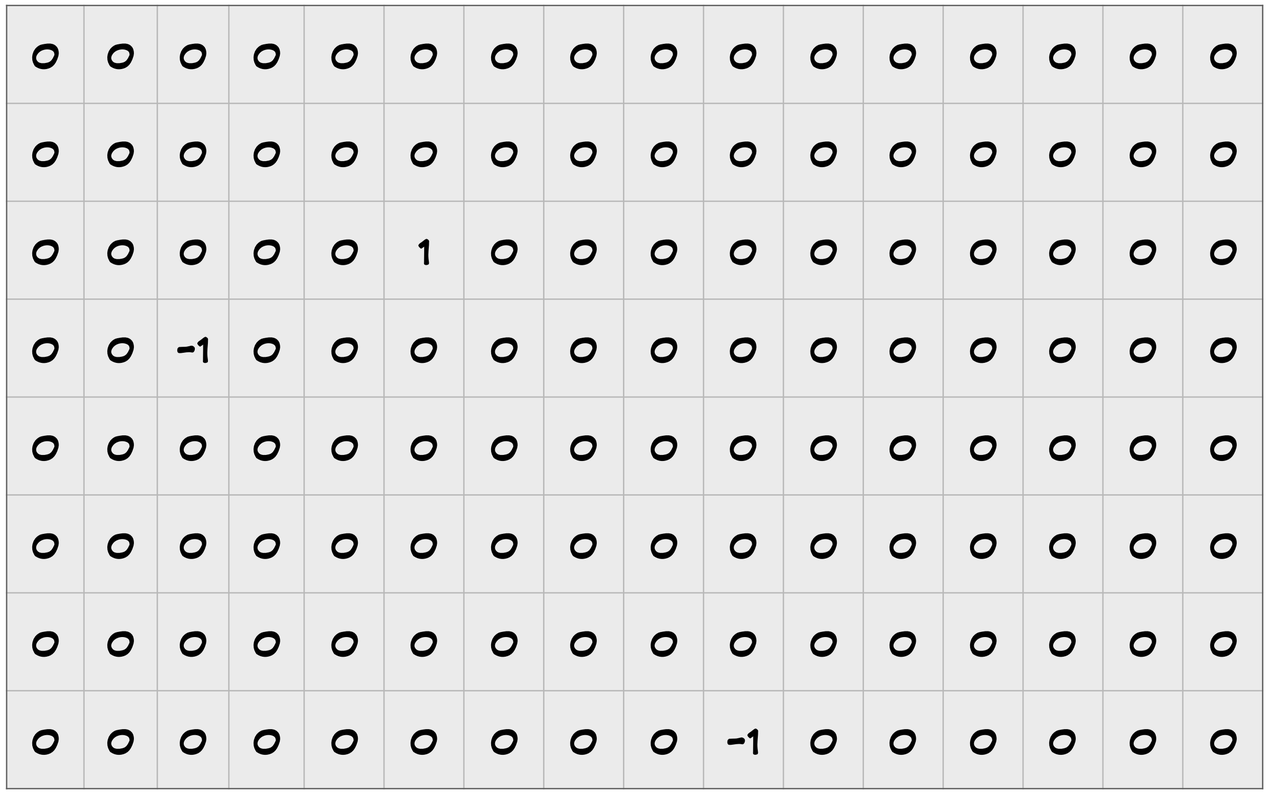
Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```

"the" → (37, 1)

"quick" → (121, -1)

"brown" → (50, -1)



Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```

"brown" → (50, -1)

"fox" → (71, 1)

"jumps" → (39, 1)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0

Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```

"jumps" → (39, 1)

"over" → (100, -1)

"the" → (37, 1)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0

Feature hashing

```
def hash_bucket(s):
  """ Assumes the existence of an external hash function.
      Returns a tuple of
        * a bucket (from 0-127, inclusive) and
        * a sign value (either +1 or -1). """
  raw_hash = my_hash(s) & 0xFF
  sign = (raw_hash & 0x80) != 0 and -1 or 1
  bucket = raw_hash & ~0x80
  return (bucket, sign)
```

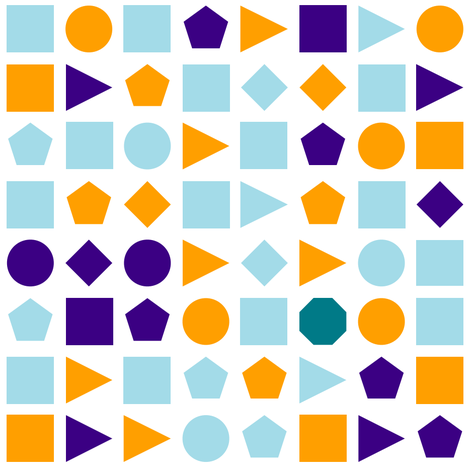
"the" → (37, 1)

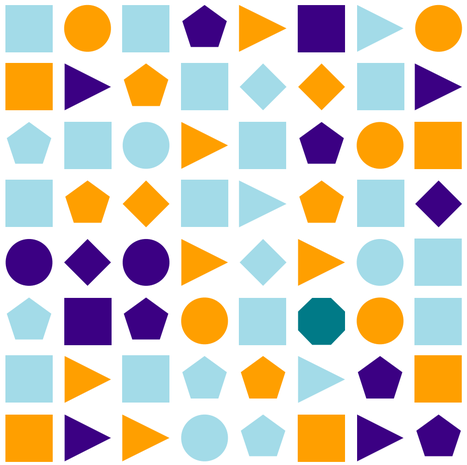
"lazy" → (120, -1)

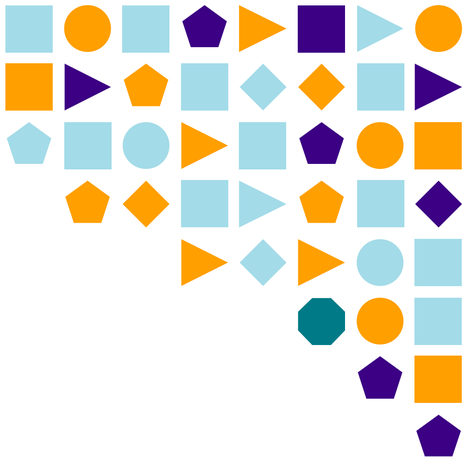
"dog" → (54, 1)

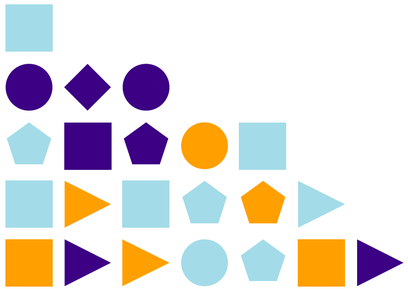
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-1	-1	0	0	0	0	0	0

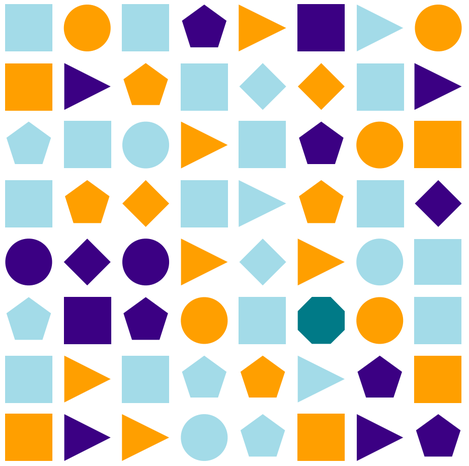
CLASSIFICATION

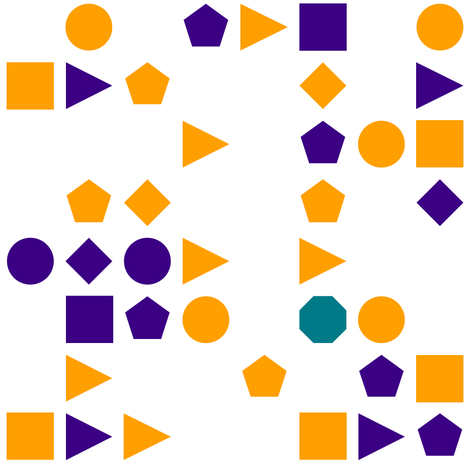


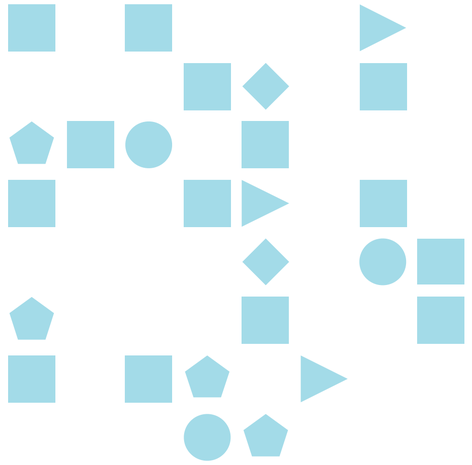


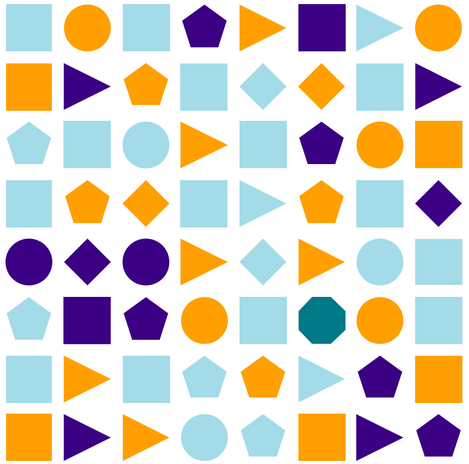


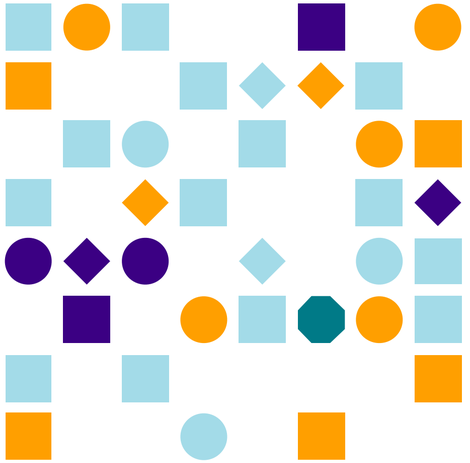


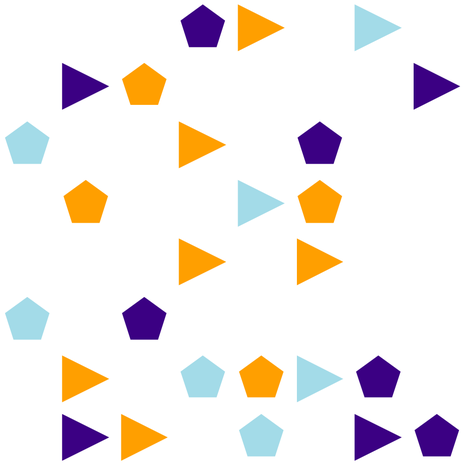




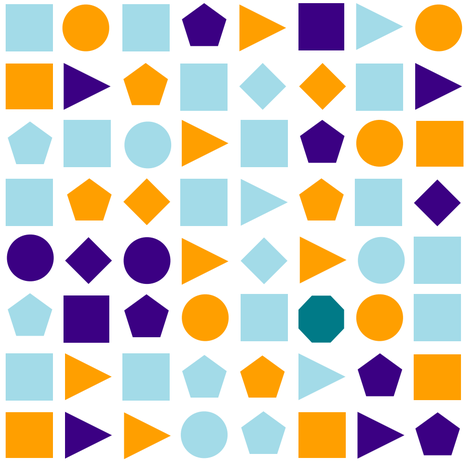


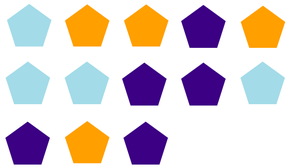




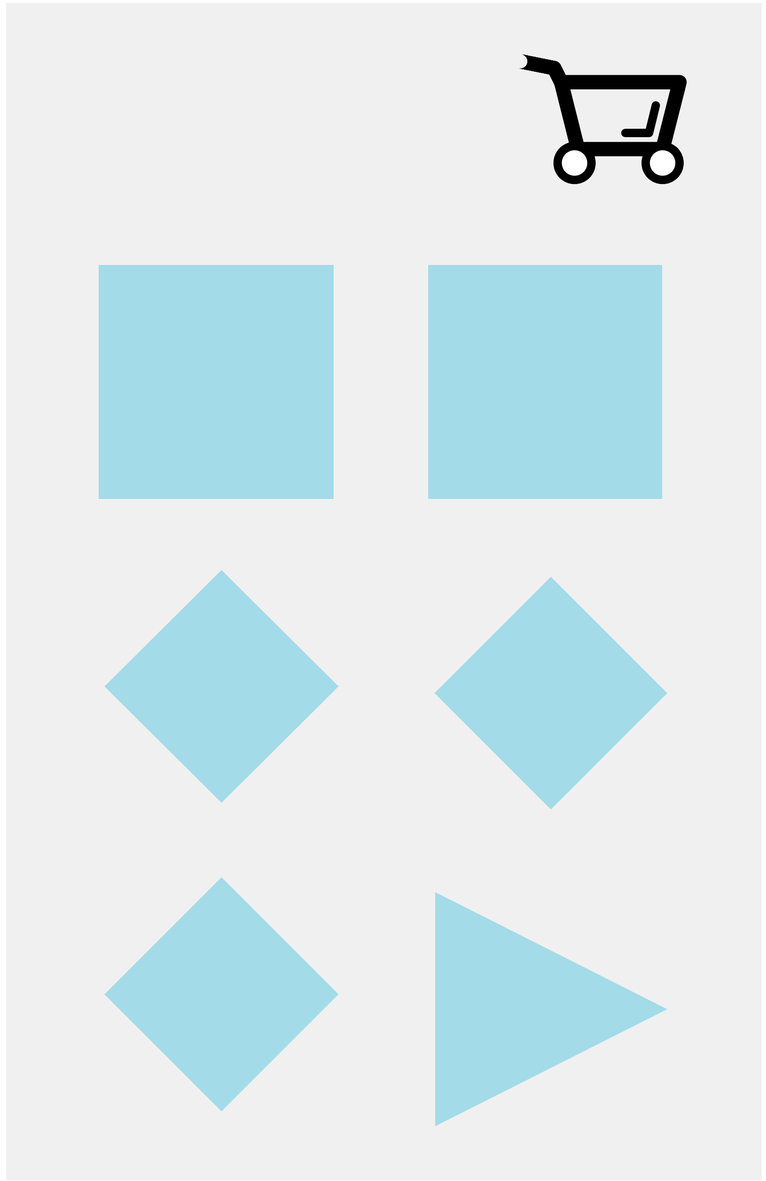


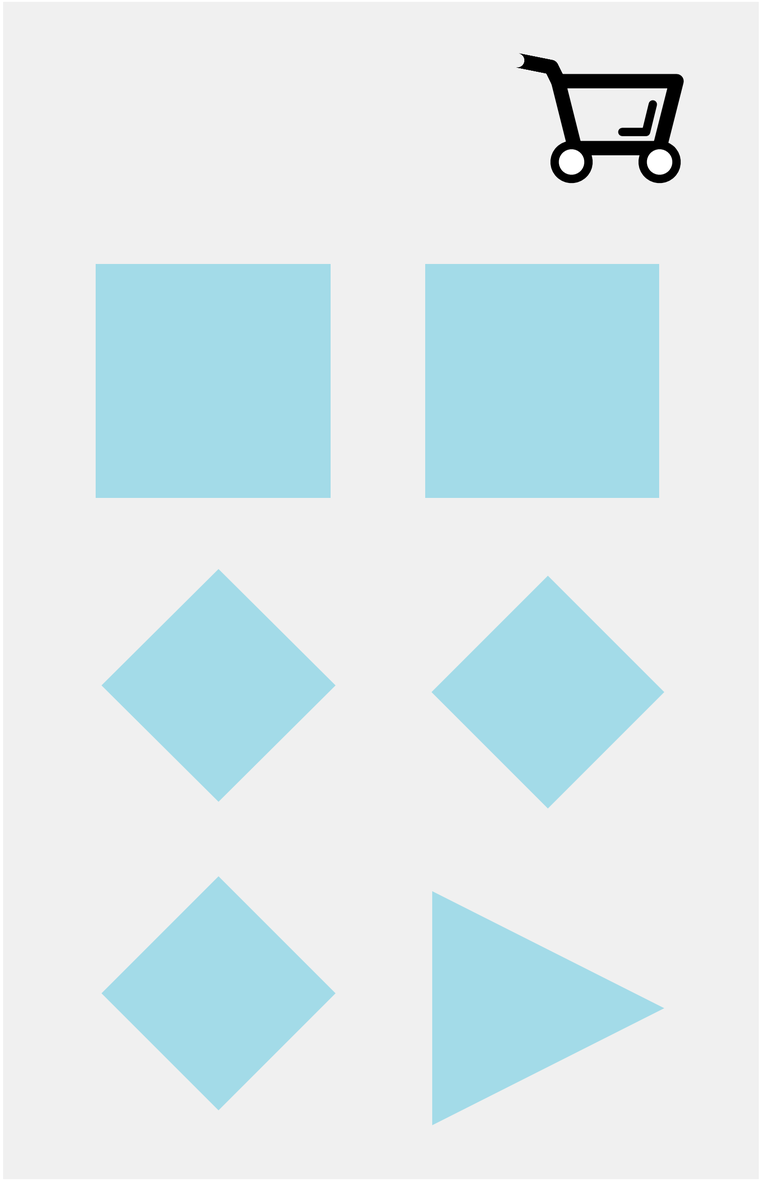
CLUSTERING

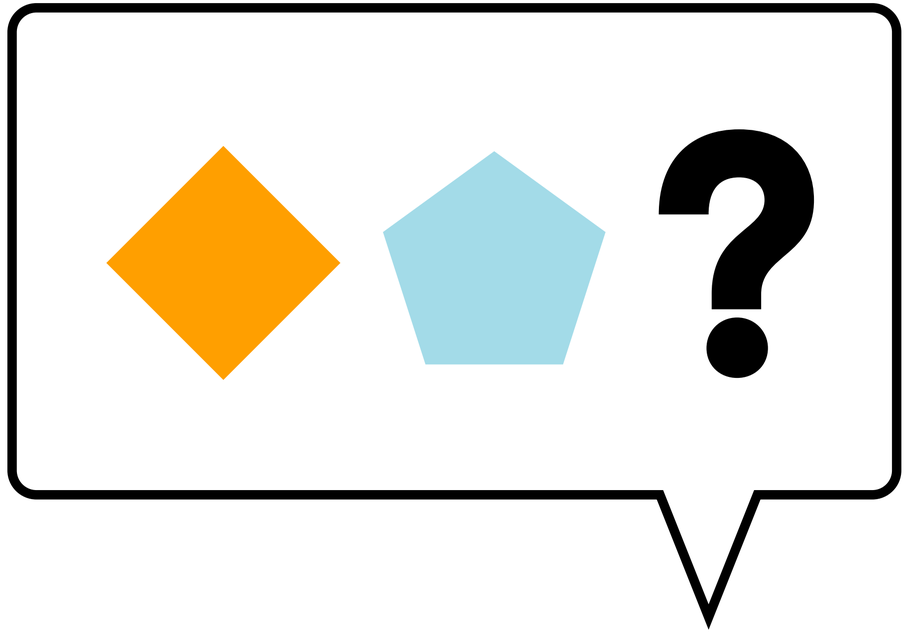




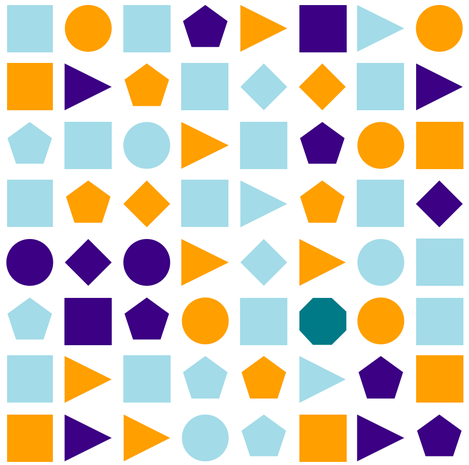
RECOMMENDATION

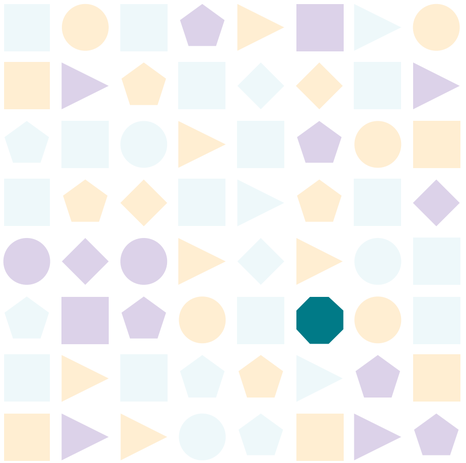




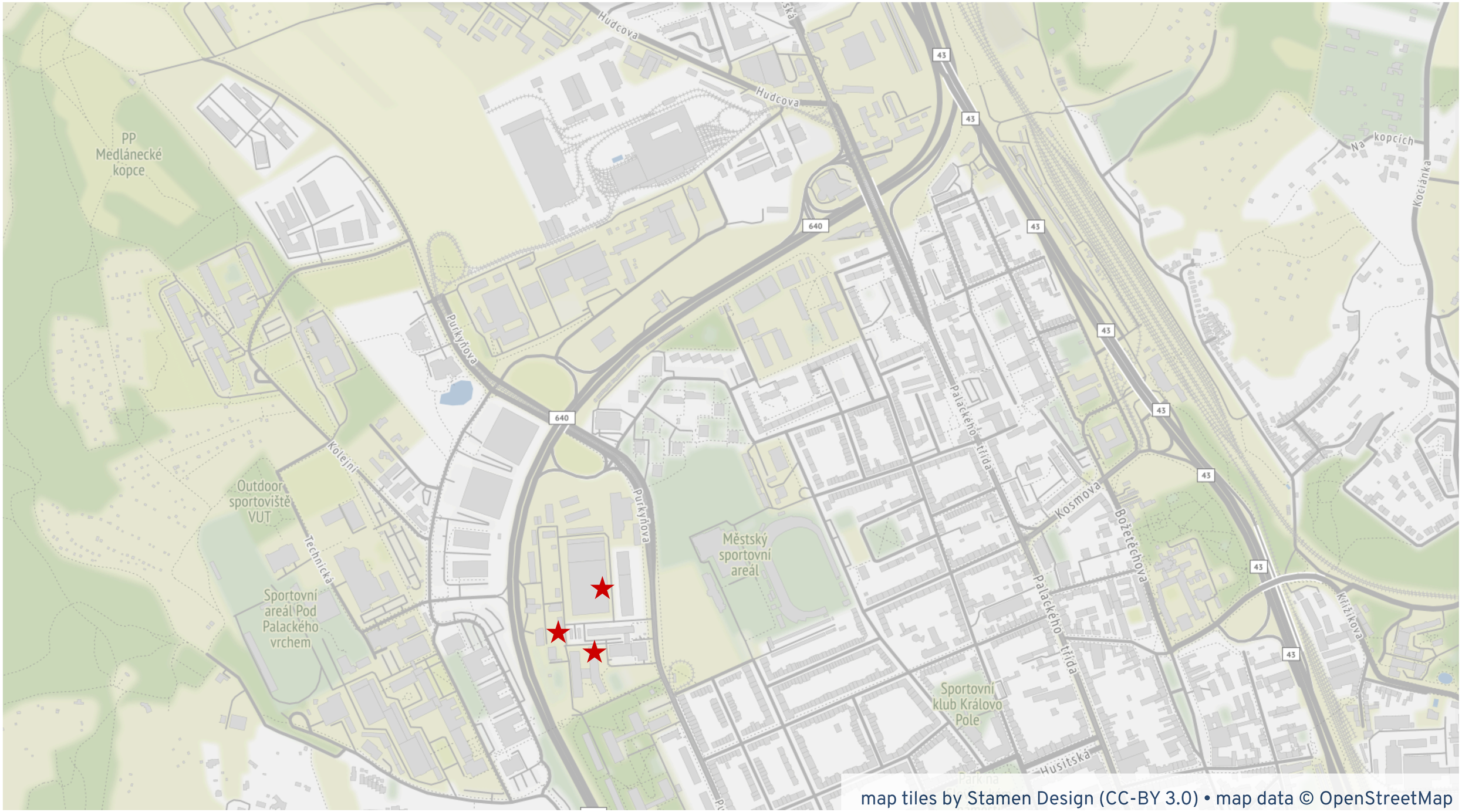


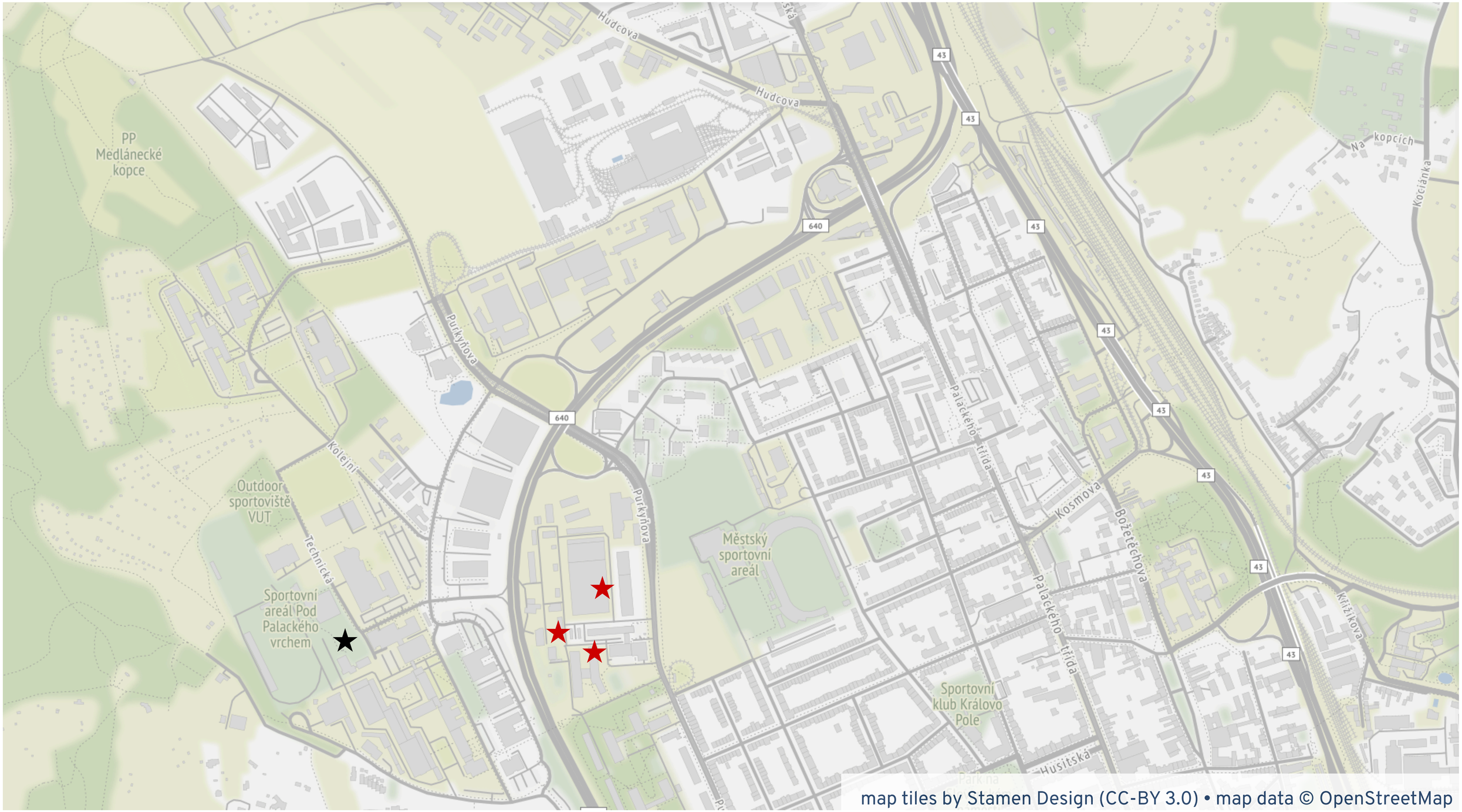
OUTLIER DETECTION

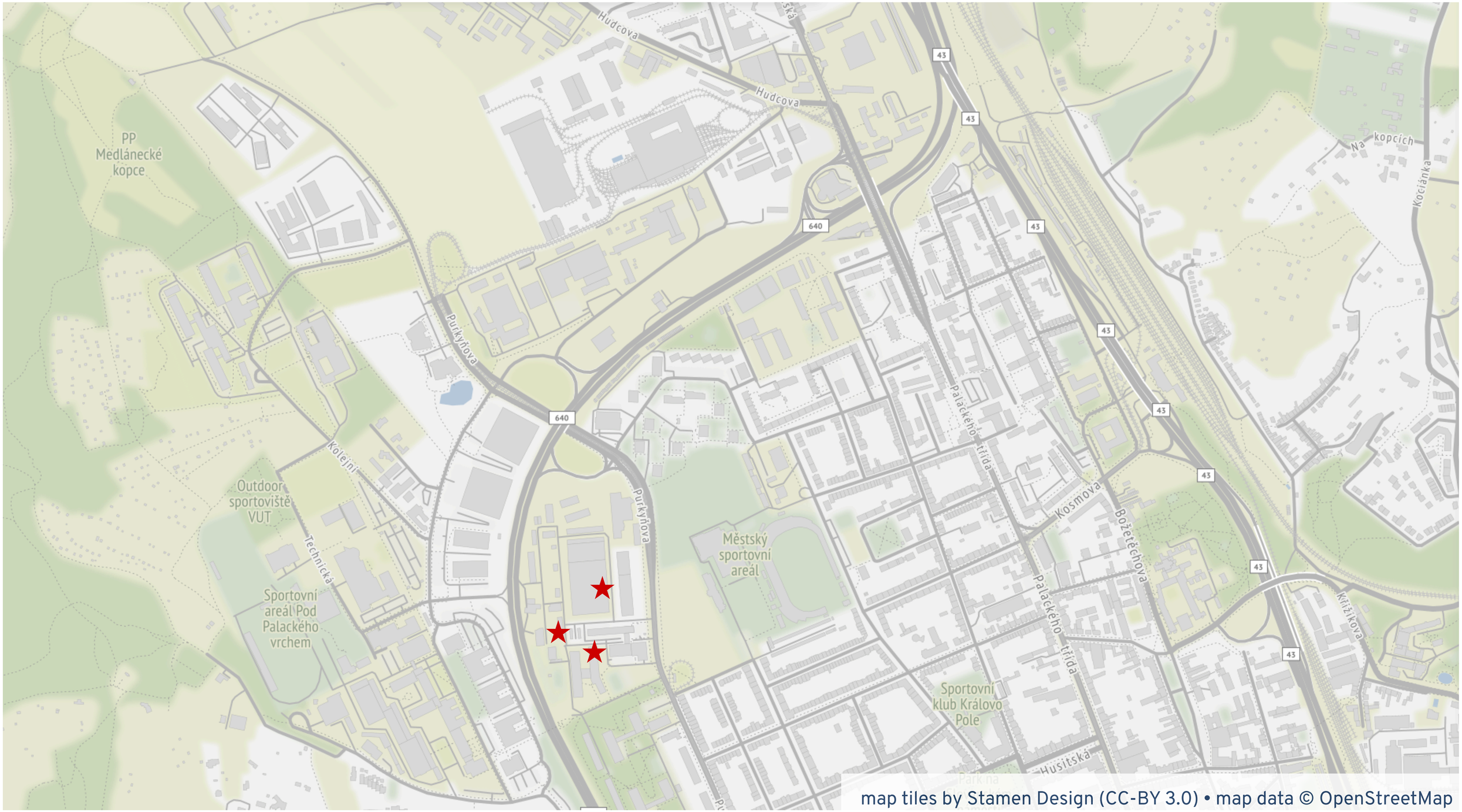


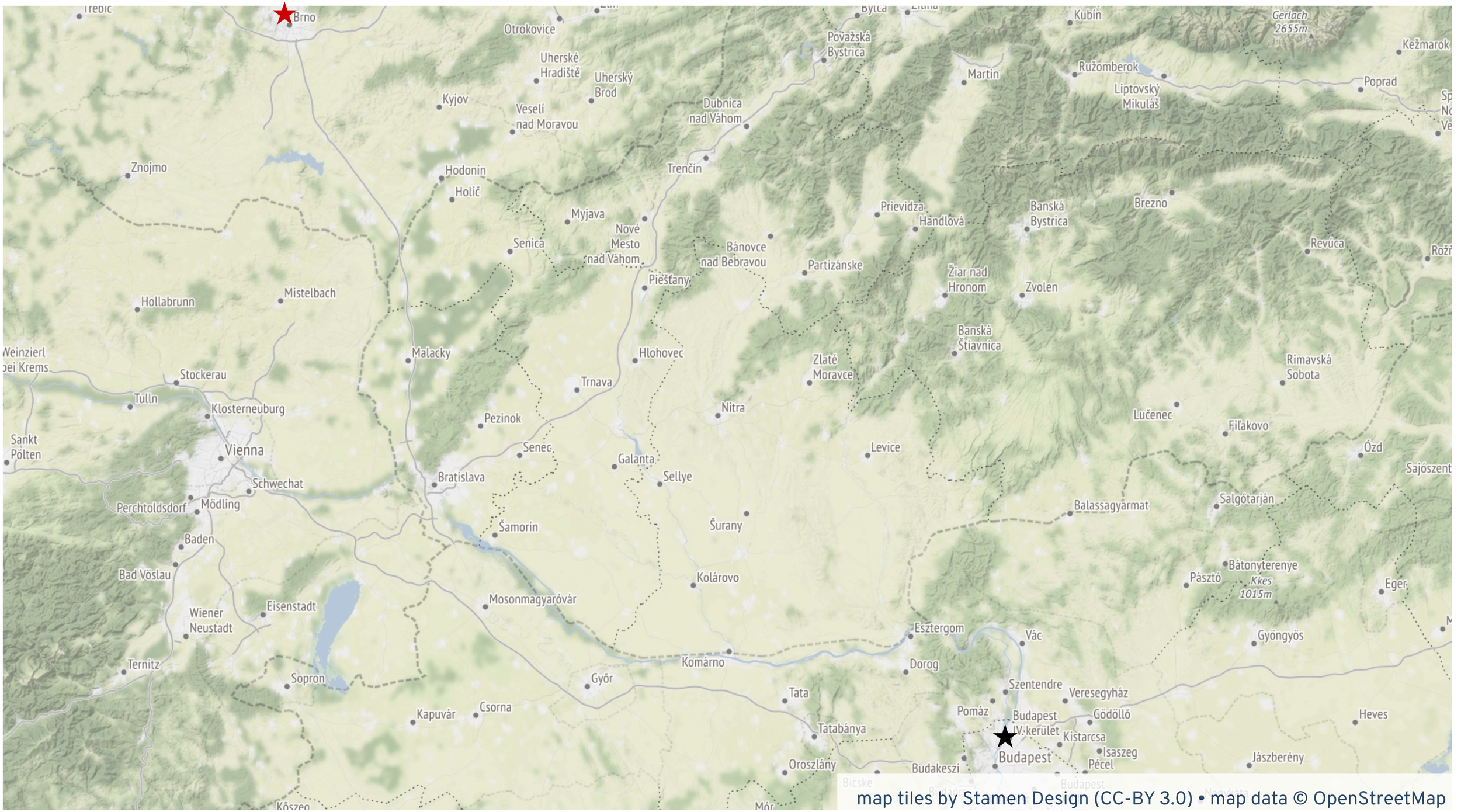



Hudcova
Hudcova
PP
Medlánecké
kopce
43
640
Purkyňova
640
Kolejní
Outdoor
sportoviště
VUT
Technická
Sportovní
areál Pod
Palackého
vrchem
Purkyňova
Městský
sportovní
areál
Palackého třída
Kosmova
Božetěchova
Palackého třída
Sportovní
klub Královo
Pole
Husitská
Na kopcích
Kociánka
Křižíkova
map tiles by Stamen Design (CC-BY 3.0) • map data © OpenStreetMap

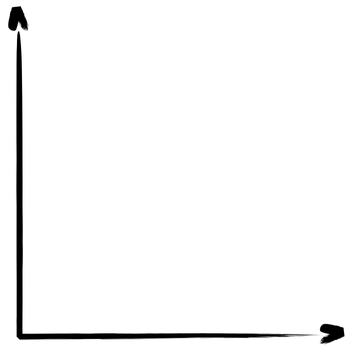

Hudcova
Hudcova
PP
Medlánecké
kopce
43
640
Purkyňova
640
Kolejní
Outdoor
sportoviště
VUT
Technická
Sportovní
areál Pod
Palackého
vrchem
Purkyňova
Městský
sportovní
areál
Palackého třída
Kosmova
Božetěchova
Palackého třída
Sportovní
klub Královo
Pole
Husitská
Na kopcích
Kociánka
Křižíkova
map tiles by Stamen Design (CC-BY 3.0) • map data © OpenStreetMap


Hudcova
Hudcova
PP
Medlánecké
kopce
Purkyňova
640
43
Palackého třída
Kosmova
Božetěchova
Na kopcích
Kociánka
Kolejní
Outdoor
sportoviště
VUT
Technická
Sportovní
areál Pod
Palackého
vrchem
Purkyňova
Městský
sportovní
areál
Sportovní
klub Královo
Pole
Husitská
Křižíkova
map tiles by Stamen Design (CC-BY 3.0) • map data © OpenStreetMap

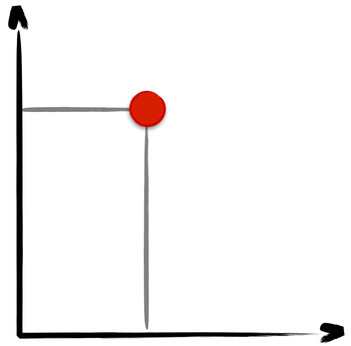

Hudcova
Hudcova
PP
Medlánecké
kopce
43
640
Purkyňova
640
Kolejní
Outdoor
sportoviště
VUT
Technická
Sportovní
areál Pod
Palackého
vrchem
Purkyňova
Městský
sportovní
areál
Palackého třída
Kosmova
Božetěchova
Palackého třída
Sportovní
klub Královo
Pole
Husitská
Na kopcích
Kociánka
Křižíkova
map tiles by Stamen Design (CC-BY 3.0) • map data © OpenStreetMap



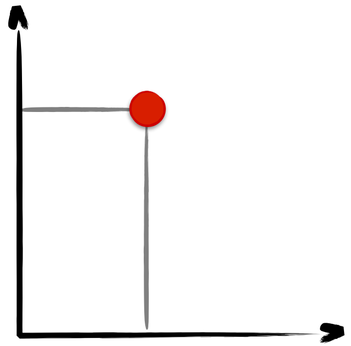
UNDERSTANDING DATA WITH MANY DIMENSIONS



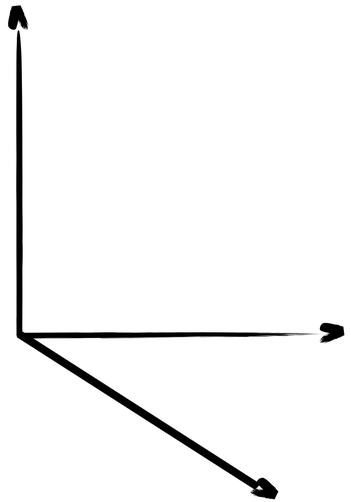
[4,7]



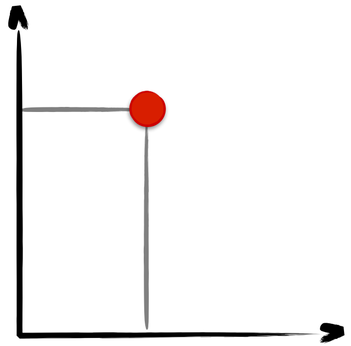
[4,7]



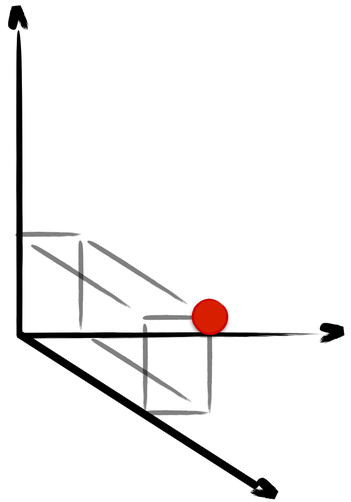
[4,7]



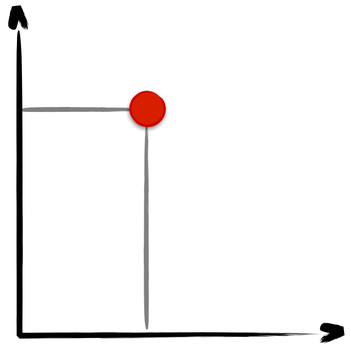
[2,3,5]



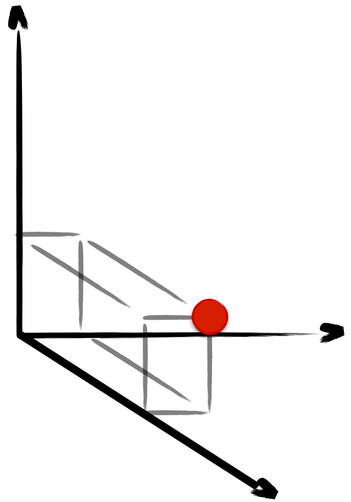
[4,7]



[2,3,5]

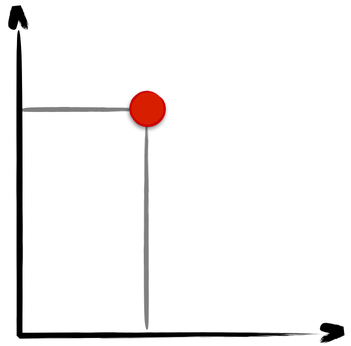


[4,7]

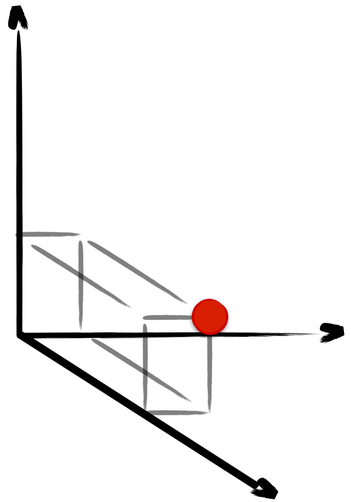


[2,3,5]

[7,1,6,5,12,
8,9,2,2,4,
7,11,6,1,5]



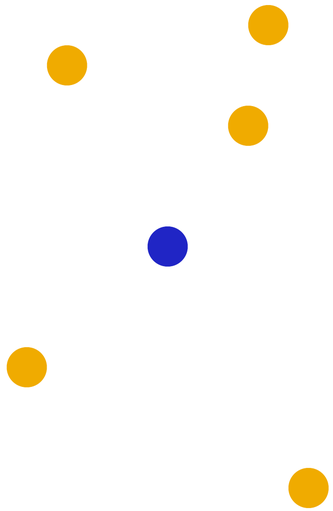
[4,7]



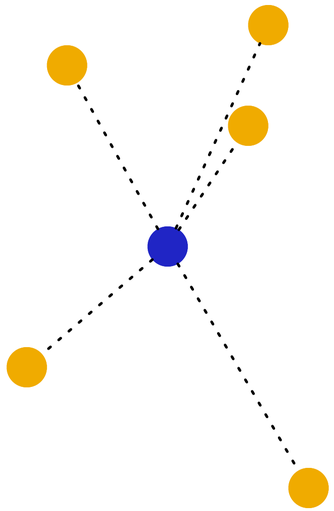
[2,3,5]

[7,1,6,5,12,
8,9,2,2,4,
7,11,6,1,5]

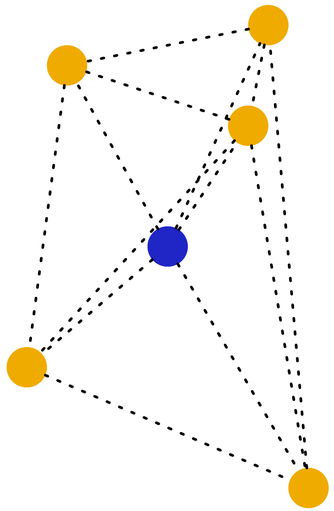
Similarity and distance



Similarity and distance

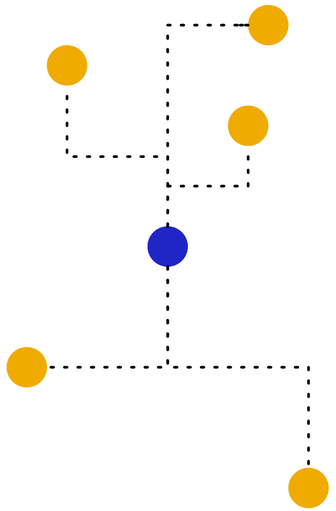


Similarity and distance



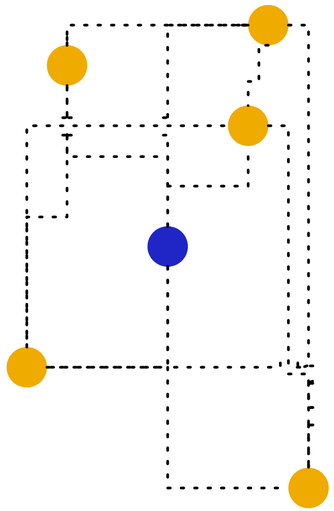
$$\sqrt{(q - p) \cdot (q - p)}$$

Similarity and distance



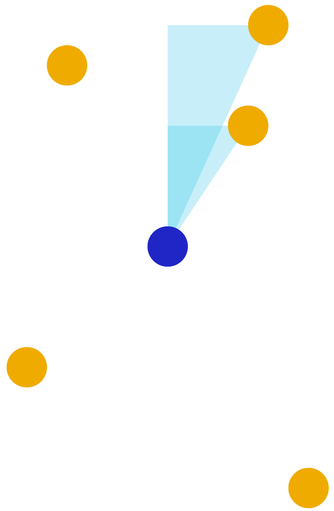
$$\sum_{i=1}^{n} |p_i - q_i|$$

Similarity and distance



$$\sum_{i=1}^{n} |p_i - q_i|$$

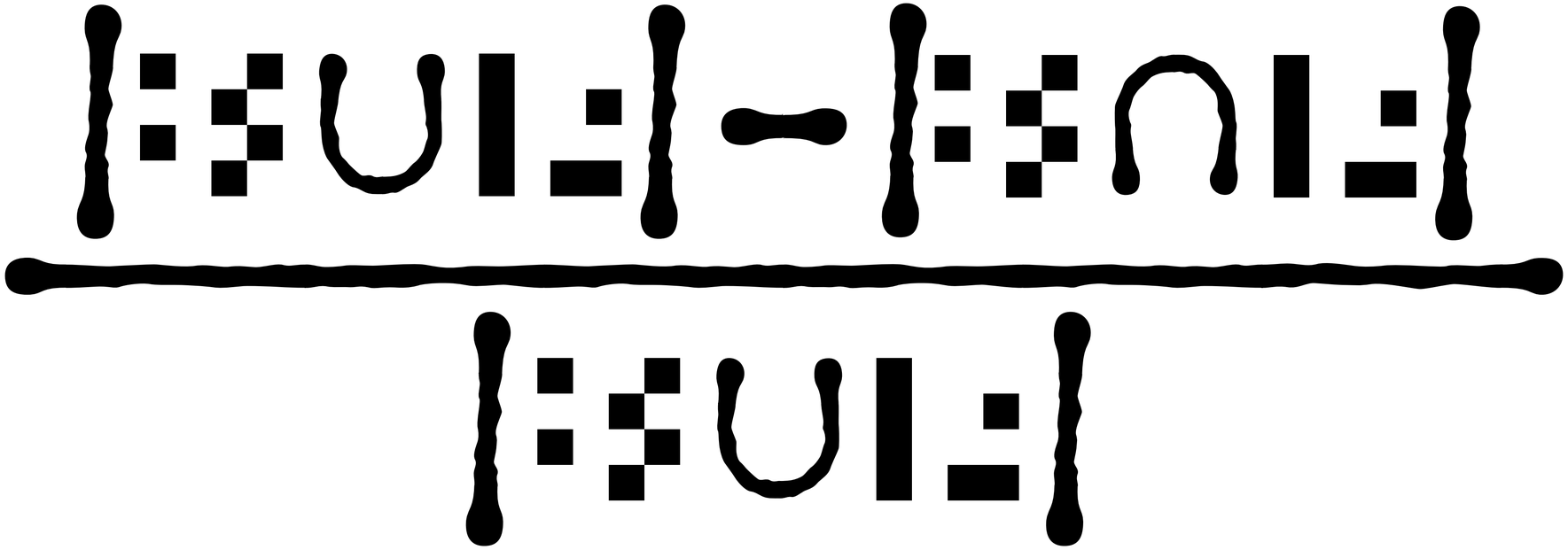
Similarity and distance



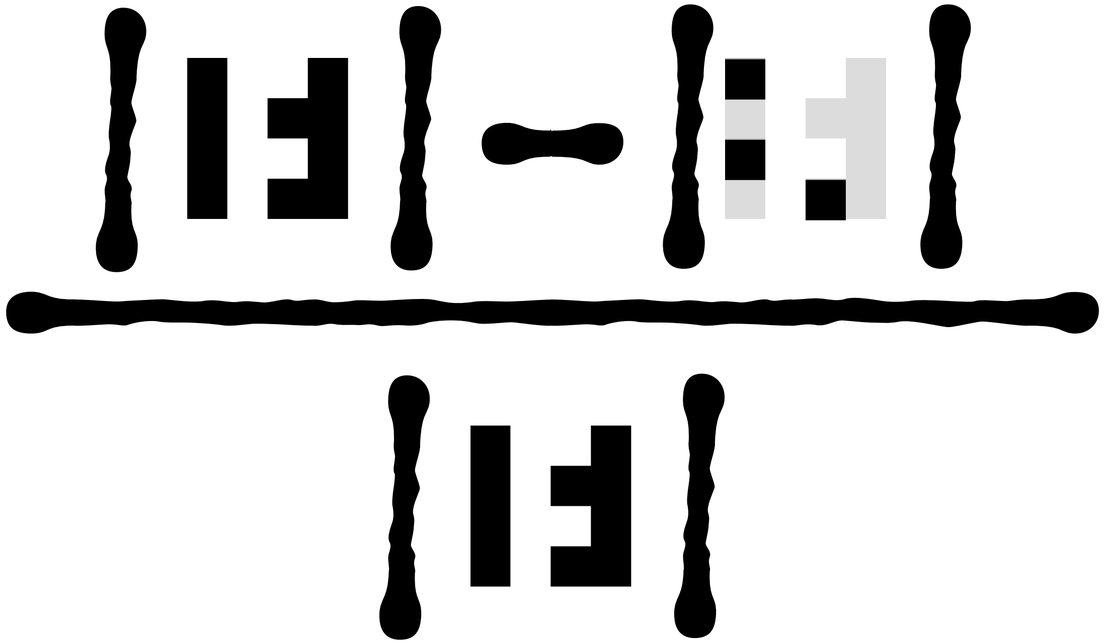
$$\frac{p \cdot q}{\|p\| \|q\|}$$

Similarity and distance

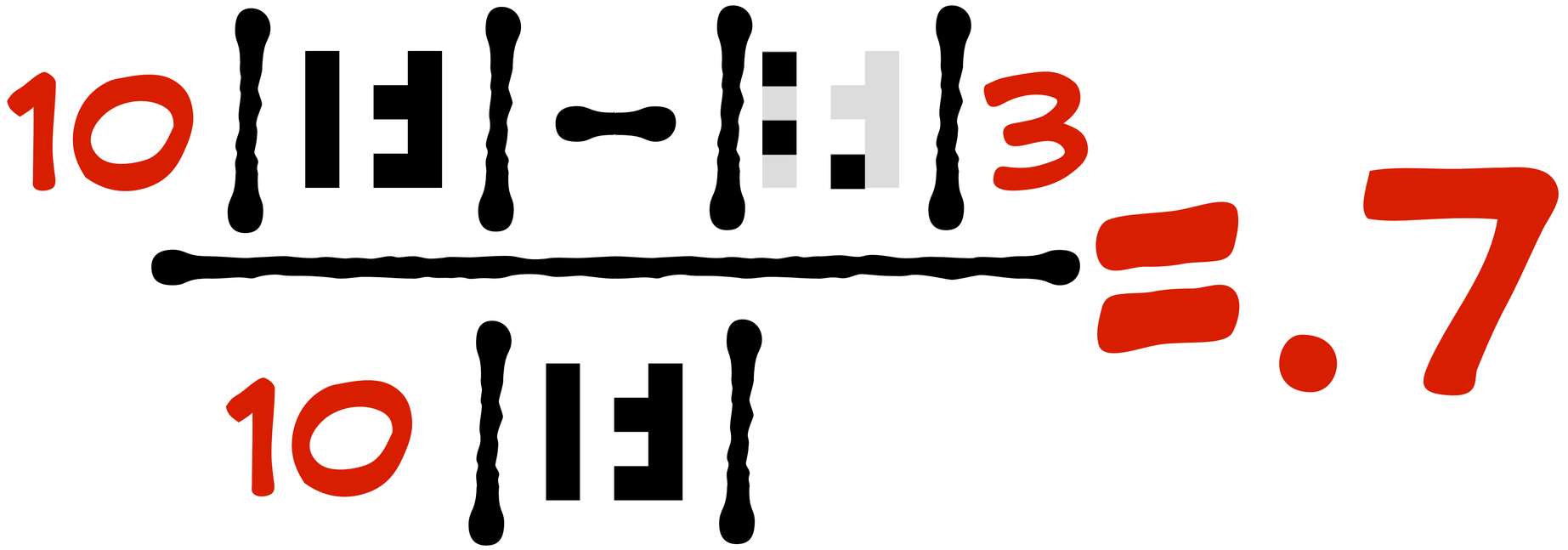
Similarity and distance



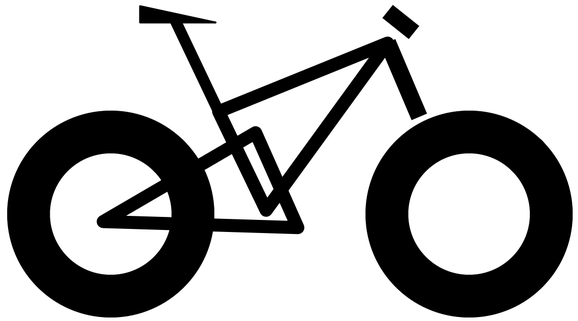
Similarity and distance

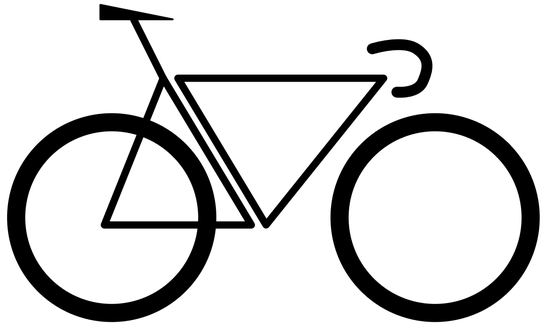


Similarity and distance



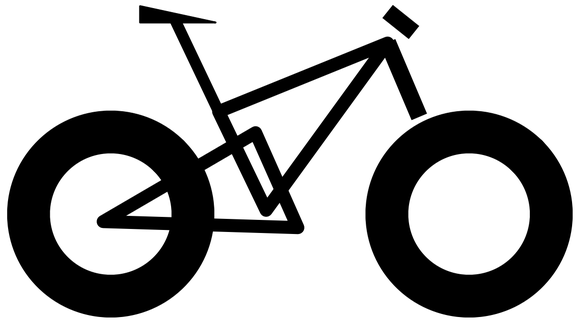
Eliminating inessential features

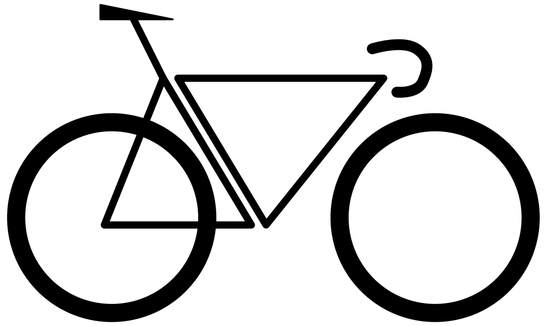




LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS	FRONT	REAR
	HANDLEBAR TYPE				SUSPENSION?	
mountain bike	0	1	0.35	1	1	1
cyclocross bike	1	0	0.13	1	0	0

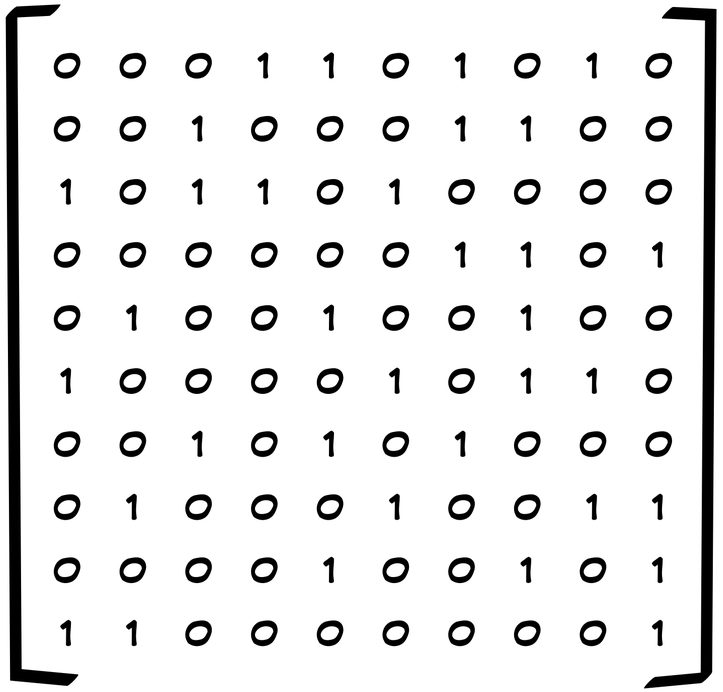
Eliminating inessential features





LABEL	DROP	FLAT	TIRE SIZE	TIRE KNOBS	FRONT	REAR
	HANDLEBAR TYPE				SUSPENSION?	
mountain bike	0	1	0.35	1	1	1
cyclocross bike	1	0	0.13	1	0	0

Very simple: random projection

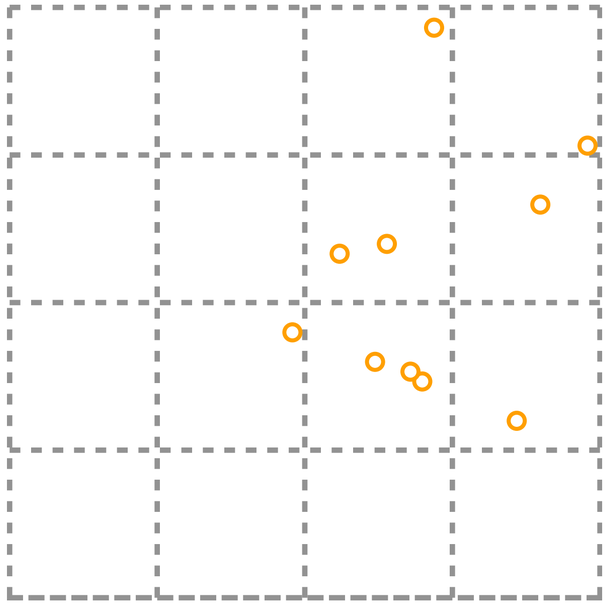


Very simple: random projection

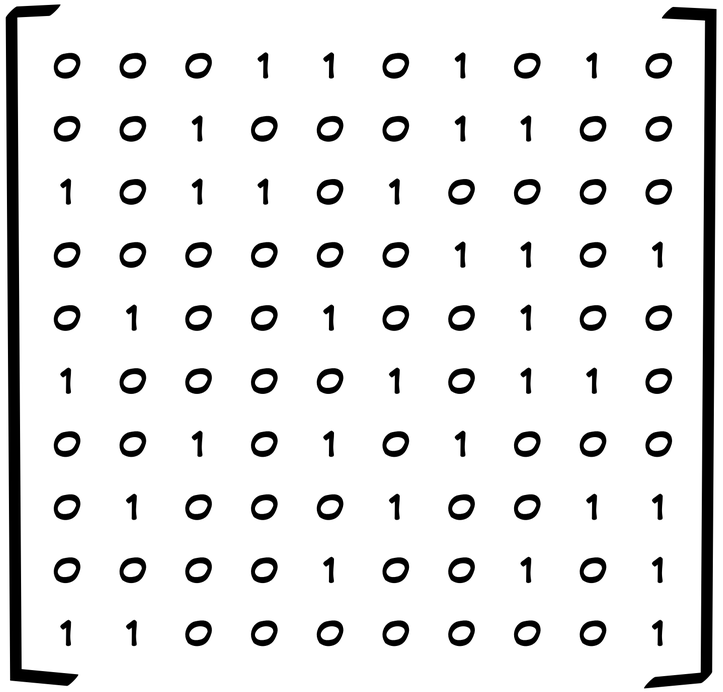
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.13 & 0.13 \\ 0.06 & 0.07 \\ 0.07 & 0.06 \\ 0.02 & 0.08 \\ 0.17 & 0.11 \\ 0.11 & 0.09 \\ 0.04 & 0.18 \\ 0.13 & 0.04 \\ 0.13 & 0.21 \\ 0.14 & 0.03 \end{bmatrix}$$

Very simple: random projection

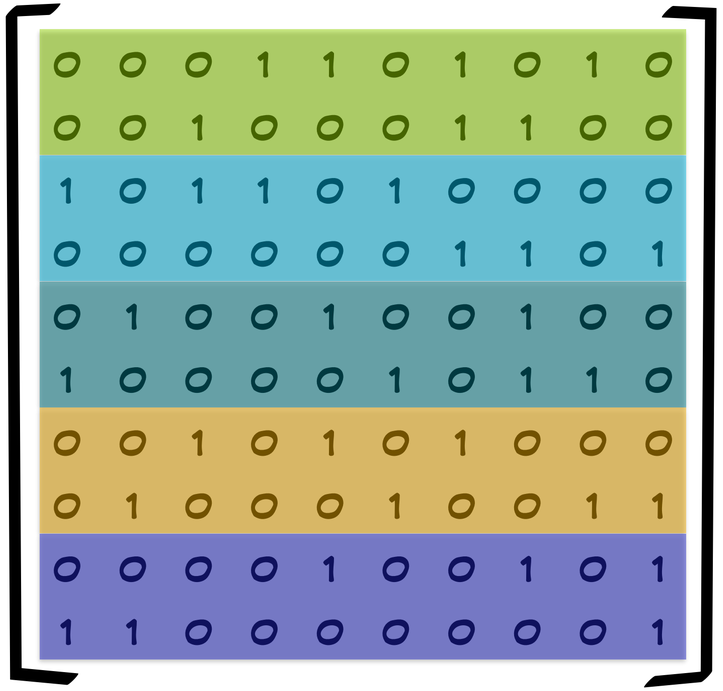
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.13 & 0.13 \\ 0.06 & 0.07 \\ 0.07 & 0.06 \\ 0.02 & 0.08 \\ 0.17 & 0.11 \\ 0.11 & 0.09 \\ 0.04 & 0.18 \\ 0.13 & 0.04 \\ 0.13 & 0.21 \\ 0.14 & 0.03 \end{bmatrix} =$$

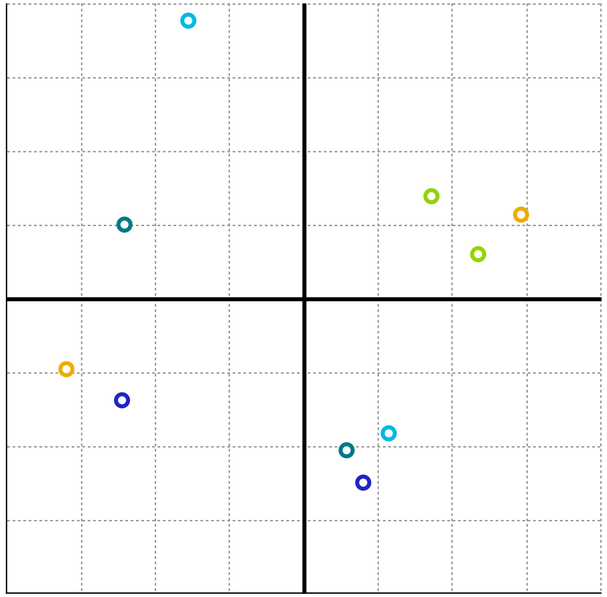


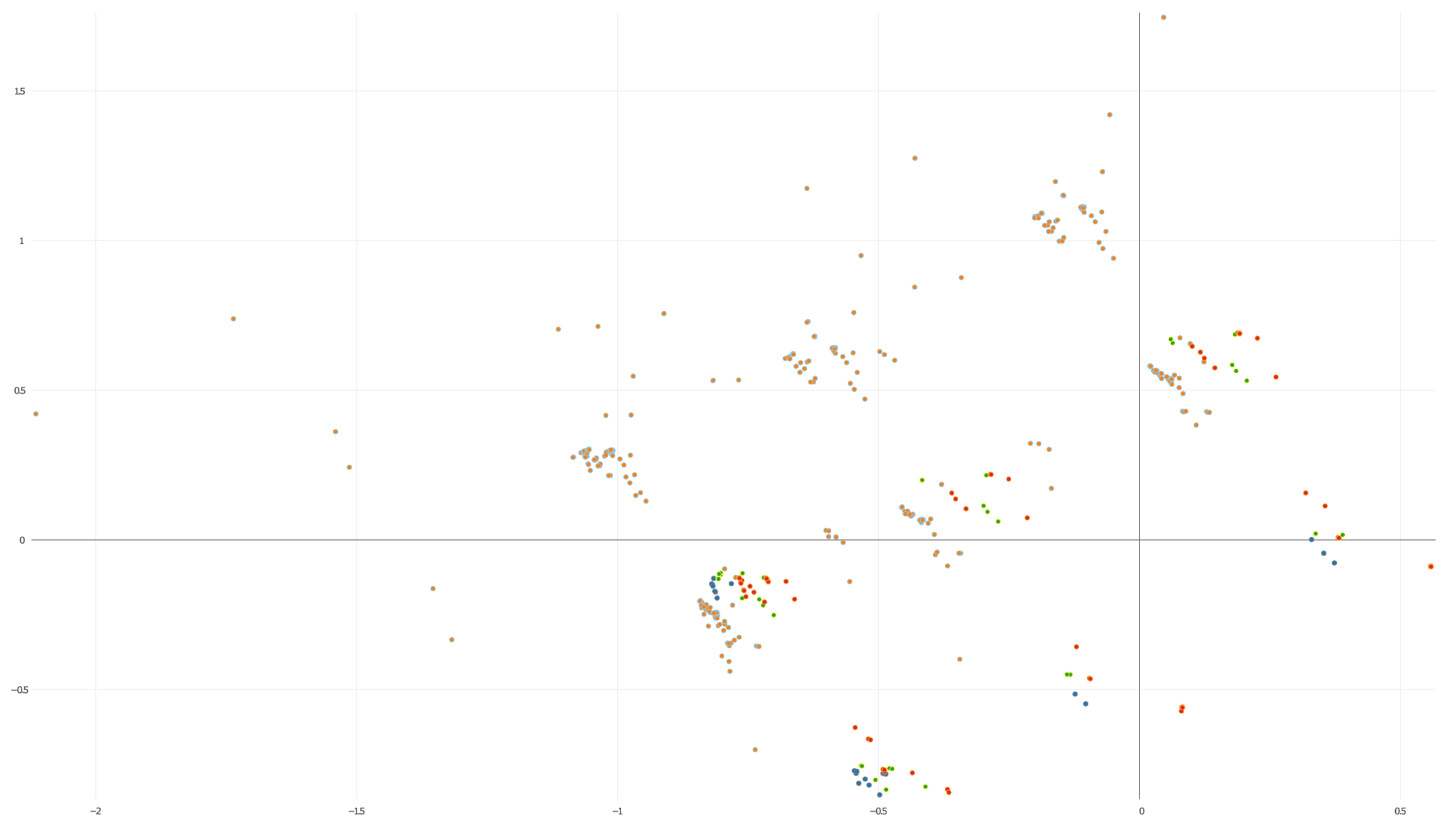
A linear approach: PCA



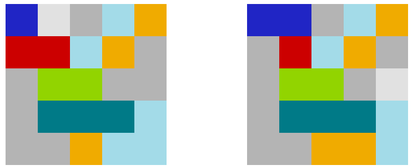
A linear approach: PCA



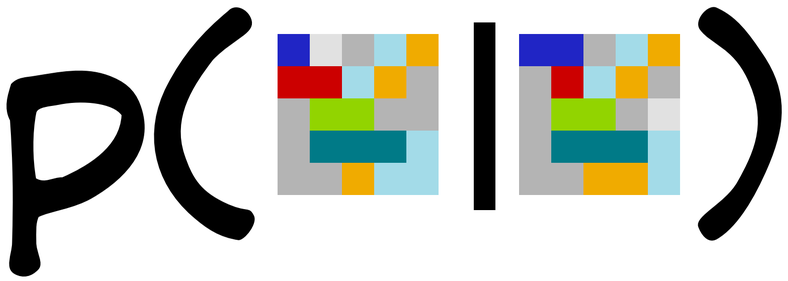



1.5
1
0.5
0
−0.5
−2
−1.5
−1
−0.5
0
0.5

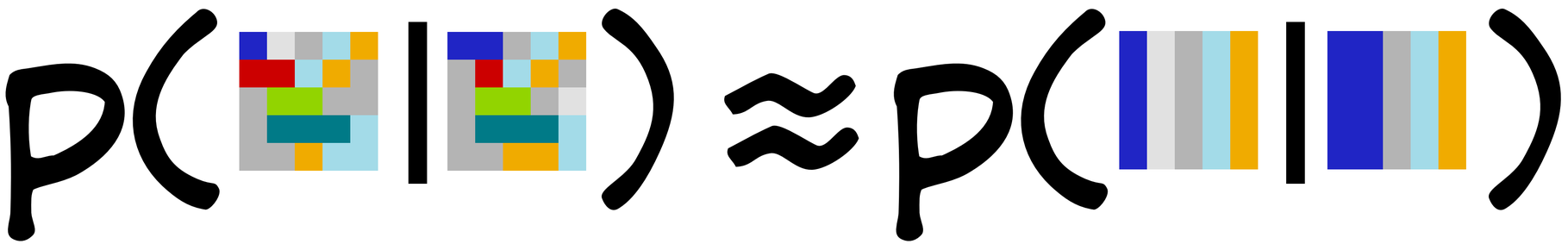
A nonlinear approach: t-SNE

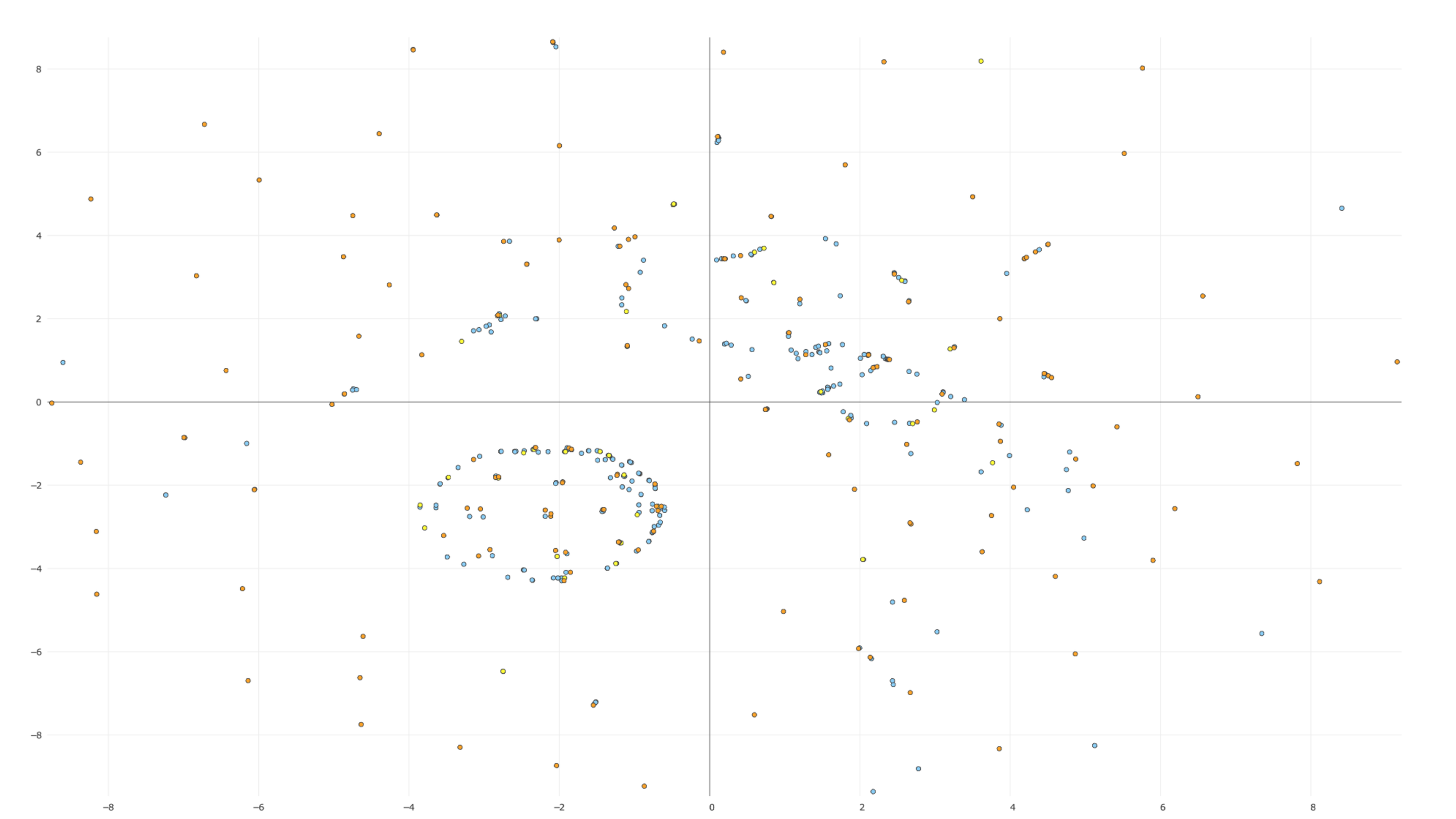


A nonlinear approach: t-SNE



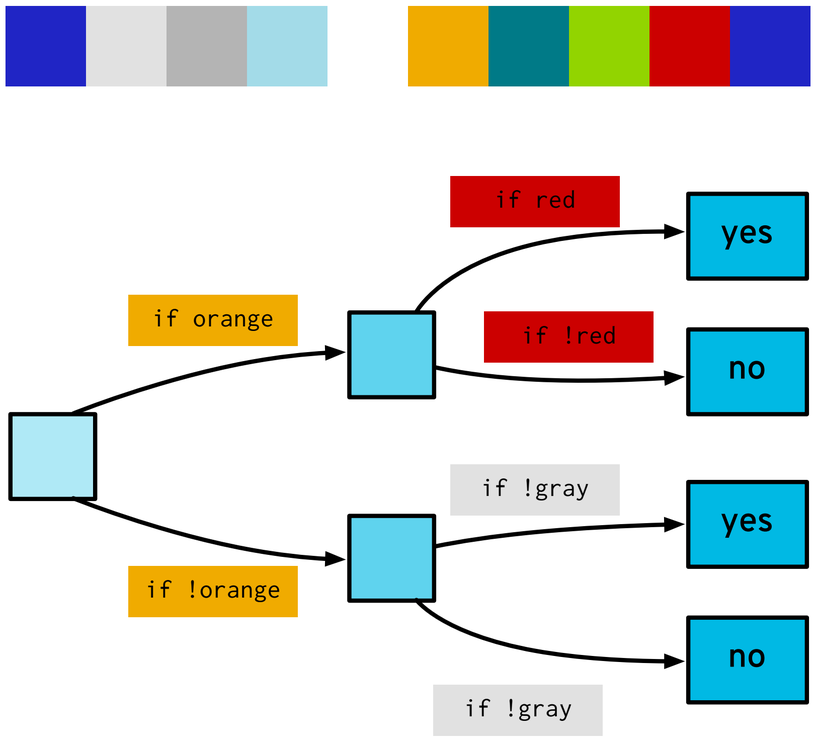
A nonlinear approach: t-SNE



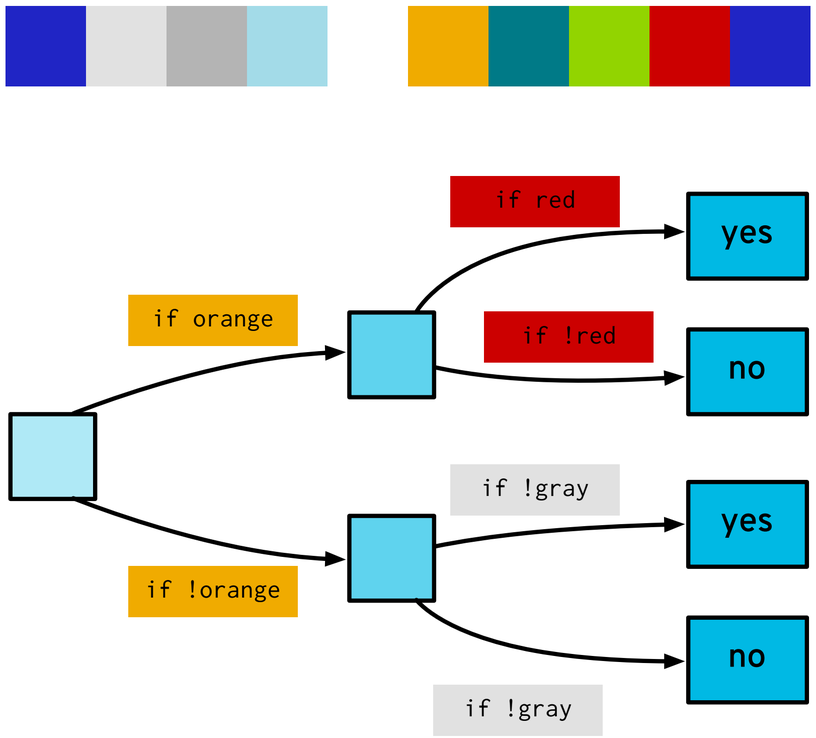

8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8

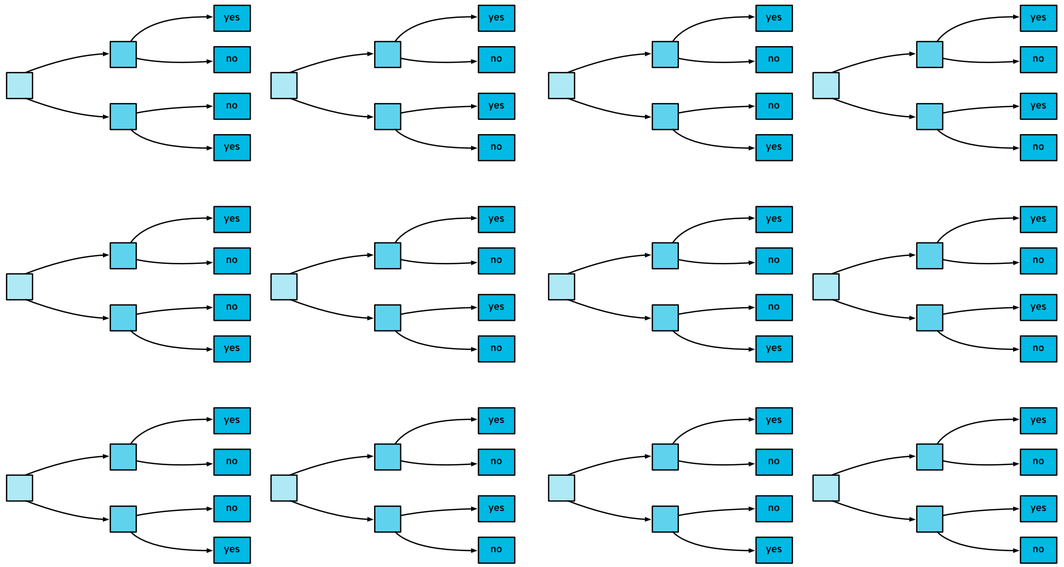
Tree-based approaches

Tree-based approaches

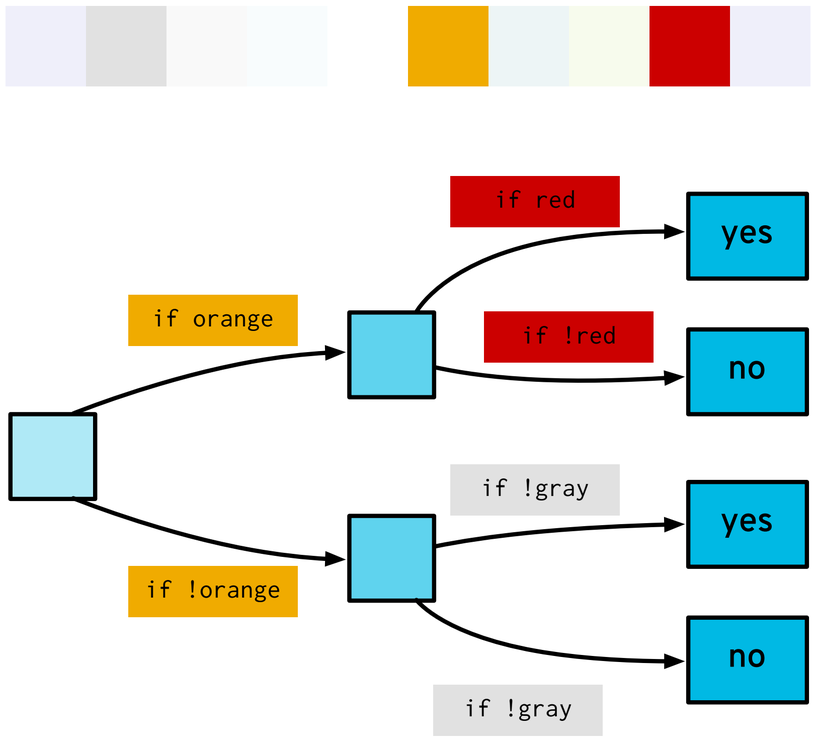


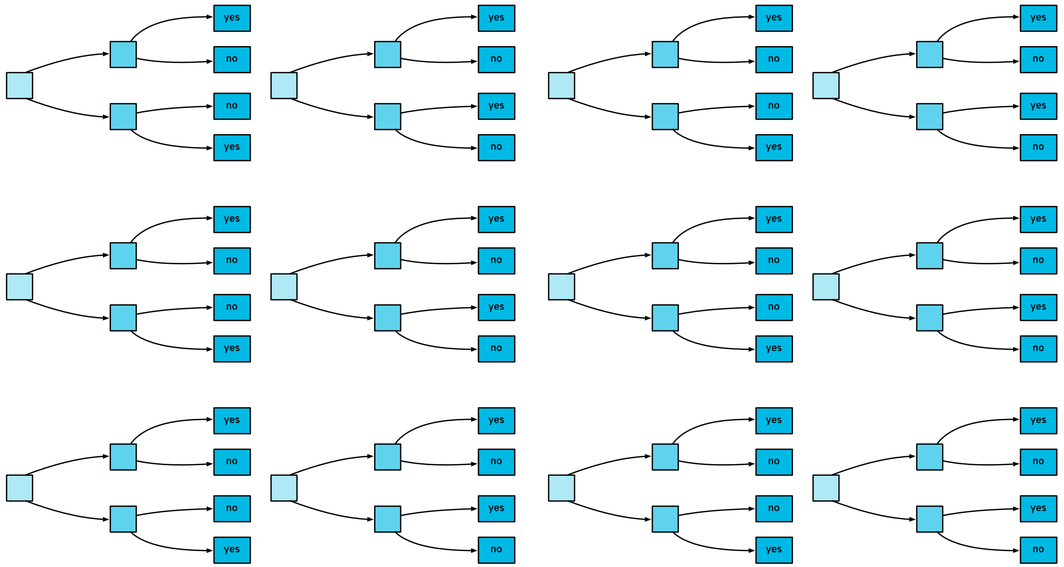
Tree-based approaches



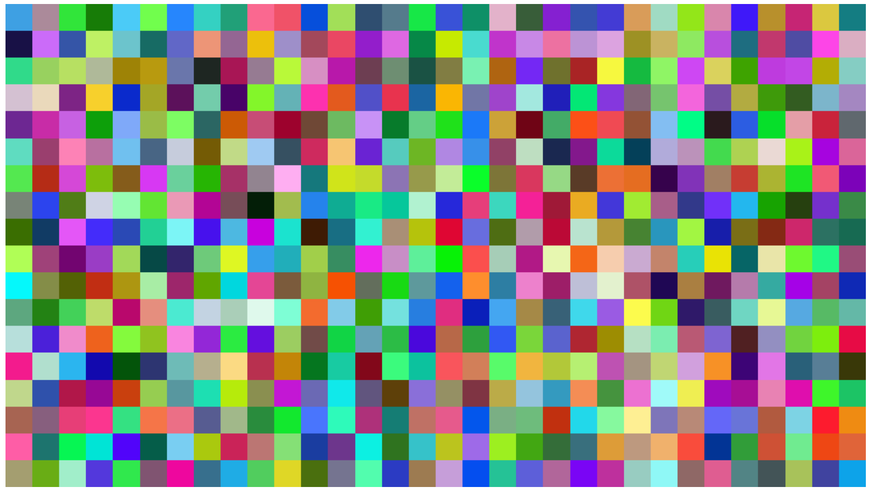


Tree-based approaches

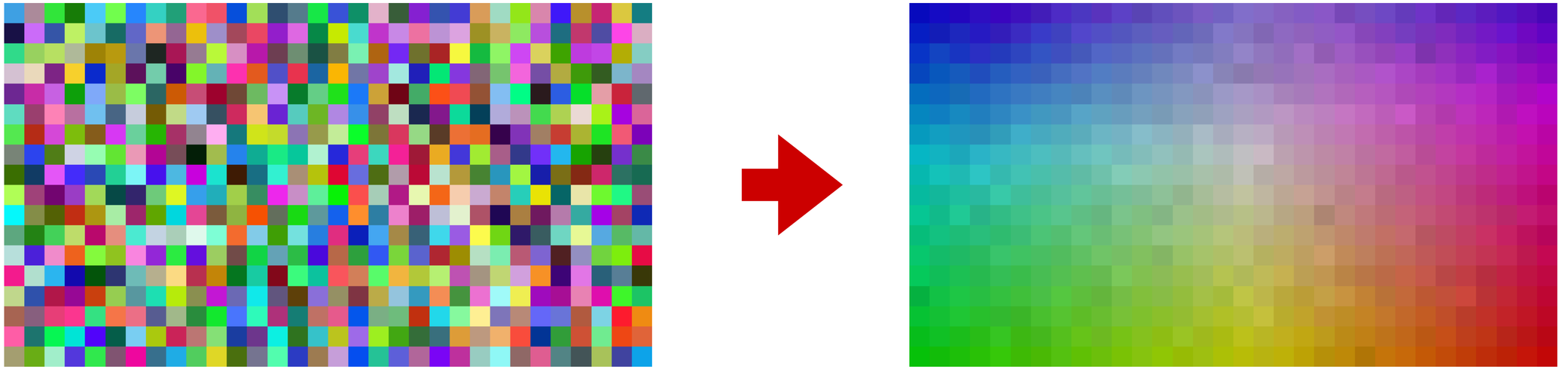


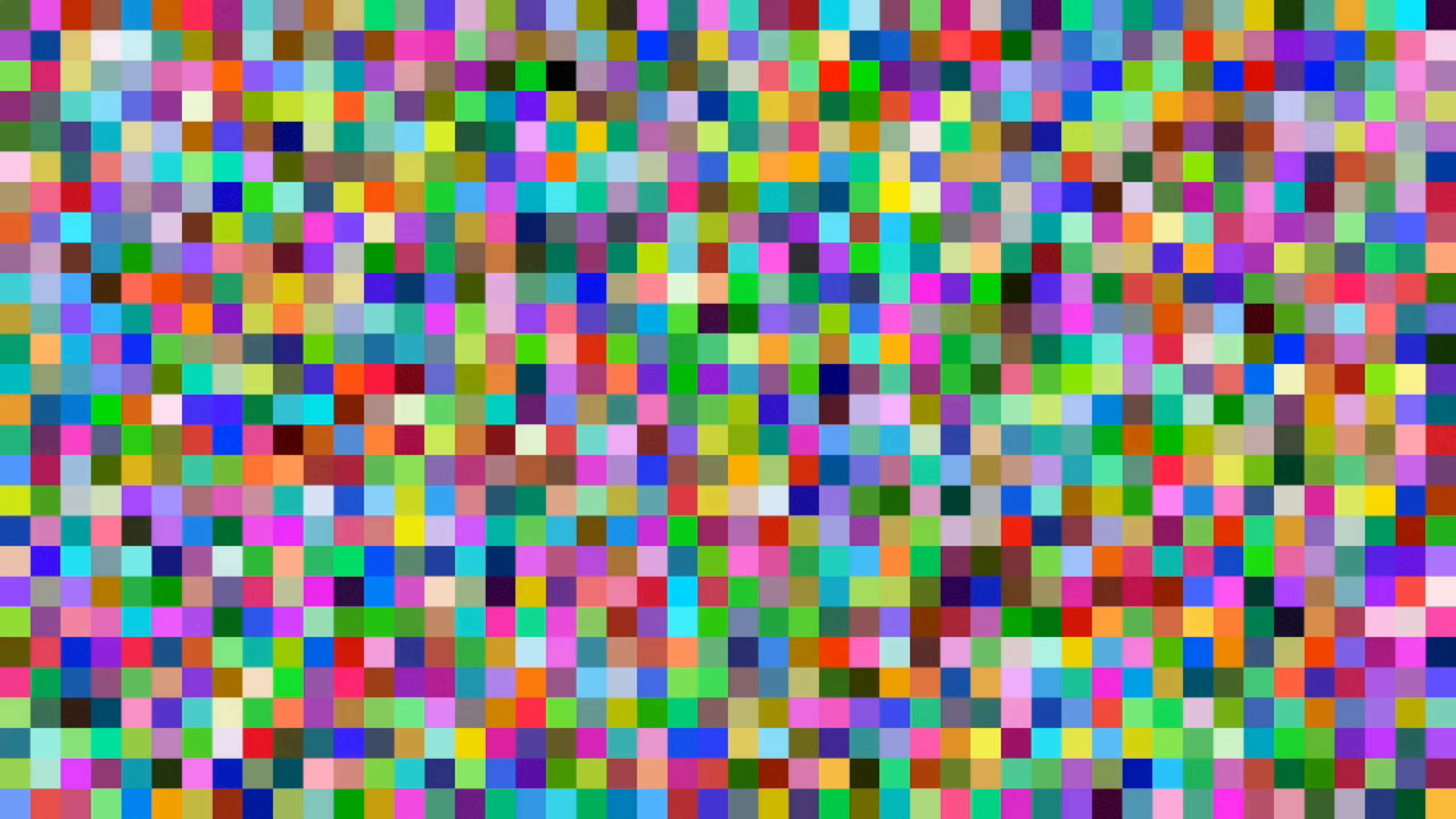


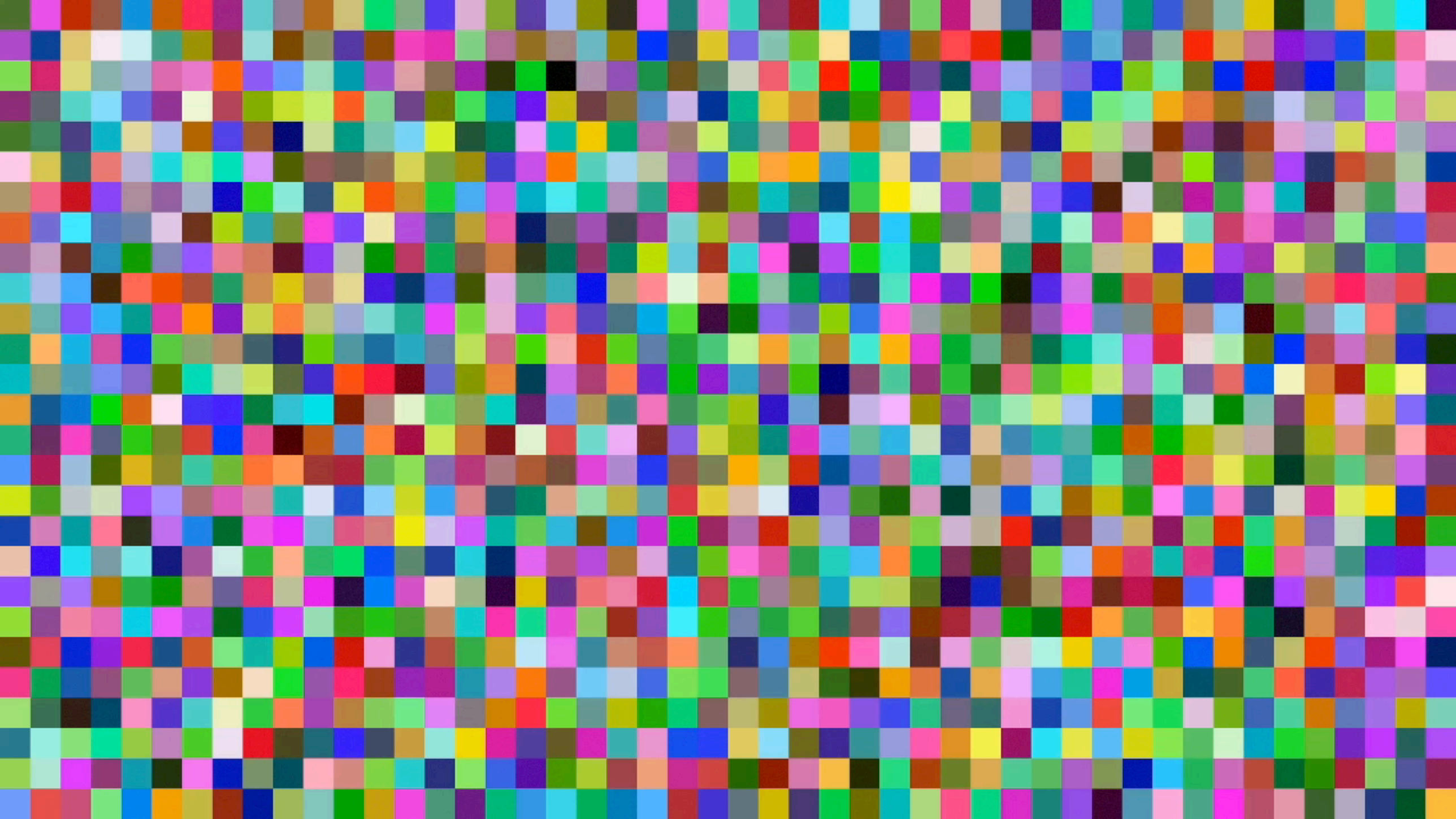
Self-organizing maps

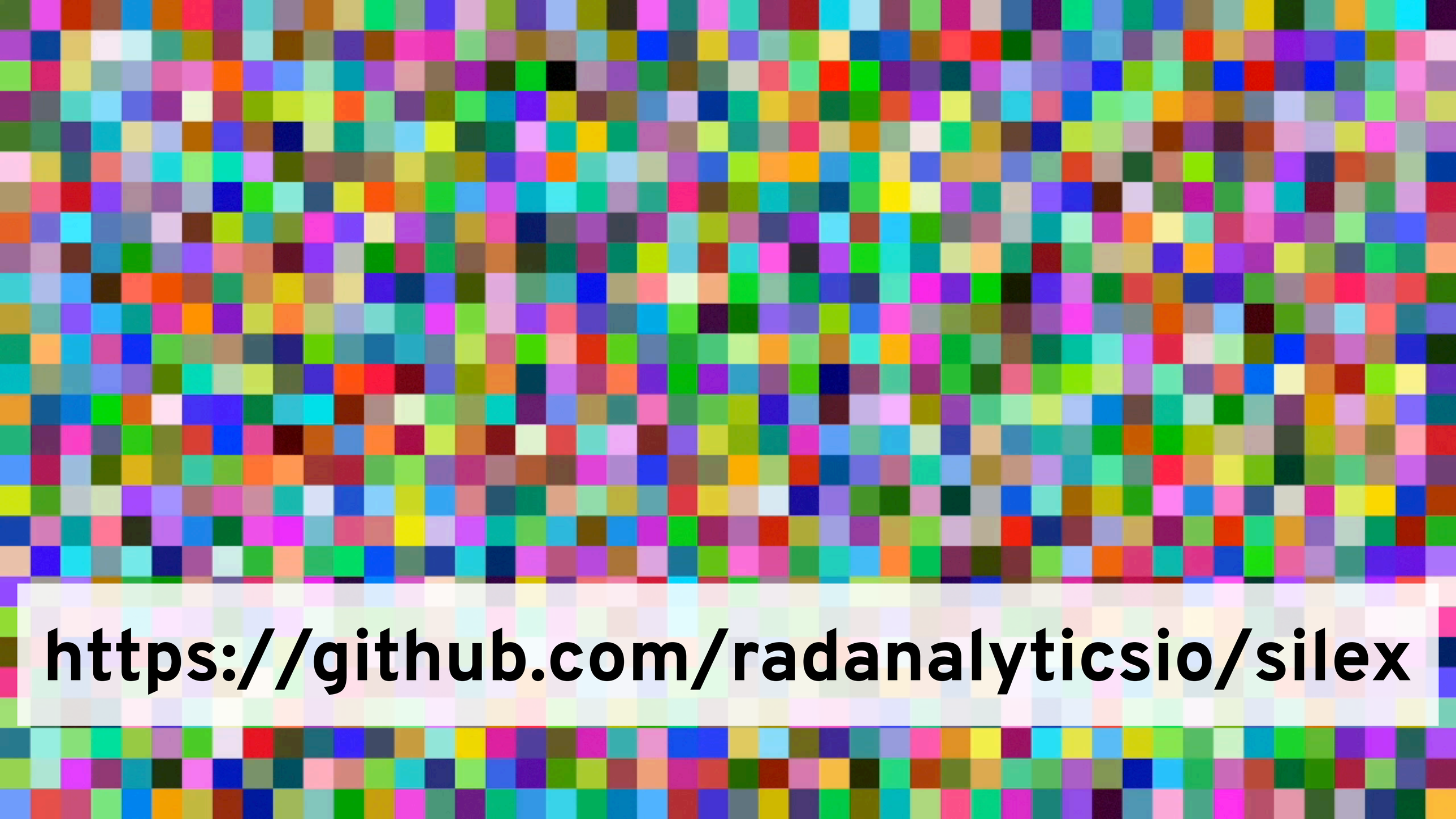


Self-organizing maps





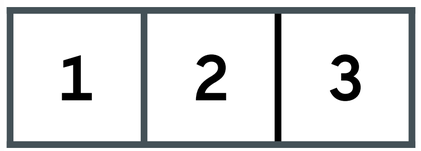


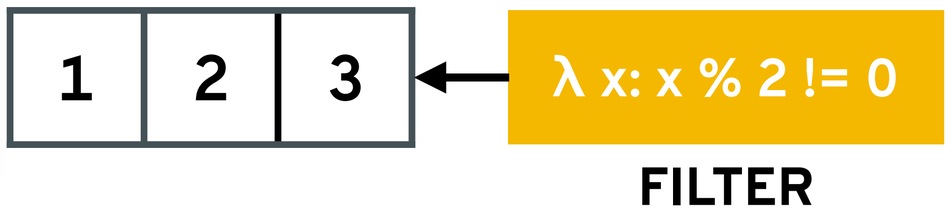

https://github.com/radanalyticsio/silex

Meet Apache Spark

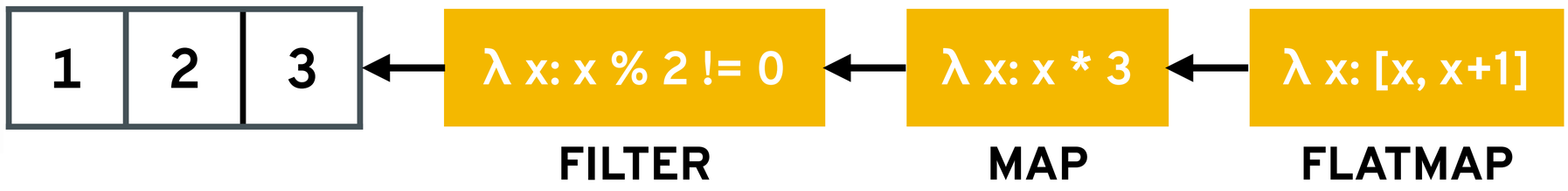
A FUNDAMENTAL ABSTRACTION, NOT AN EXECUTION MODEL

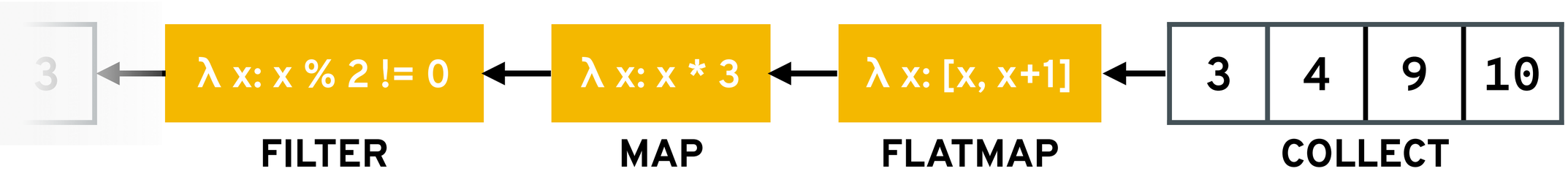
Resilient Distributed Datasets are **partitioned**, **lazy**, and **immutable** **homogeneous collections.**

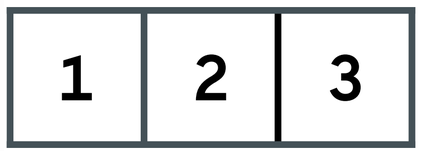

1
2
3

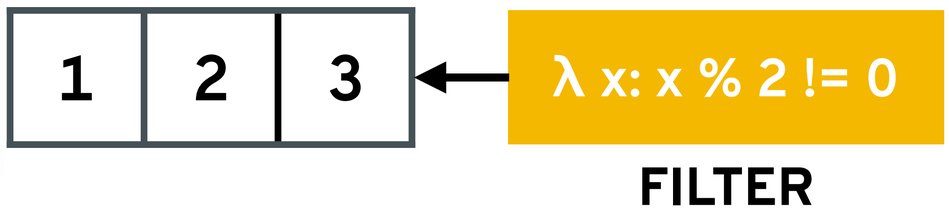

1
2
3
λ x: x % 2 != 0
FILTER

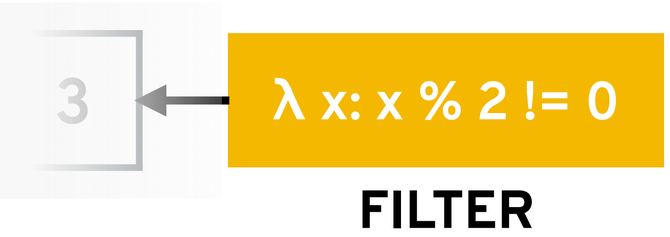

1
2
3
λ x: x % 2 != 0
λ x: x * 3
FILTER
MAP


1
2
3
λ x: x % 2 != 0
λ x: x * 3
λ x: [x, x+1]
FILTER
MAP
FLATMAP

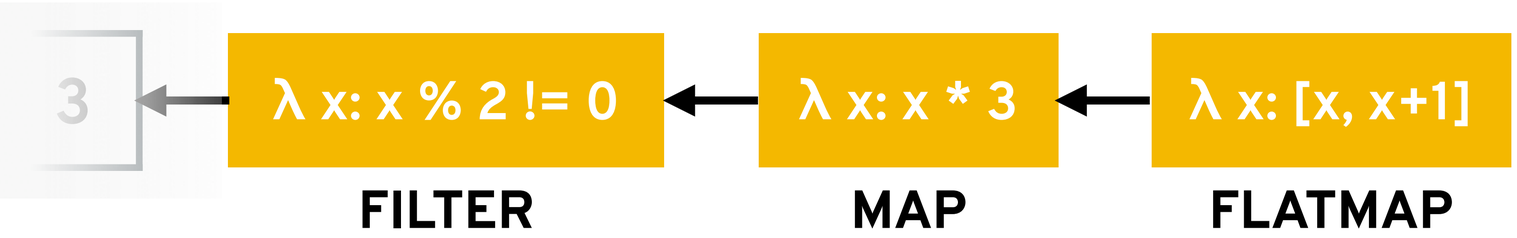

3
λ x: x % 2 != 0
λ x: x * 3
λ x: [x, x+1]
3
4
9
10
FILTER
MAP
FLATMAP
COLLECT

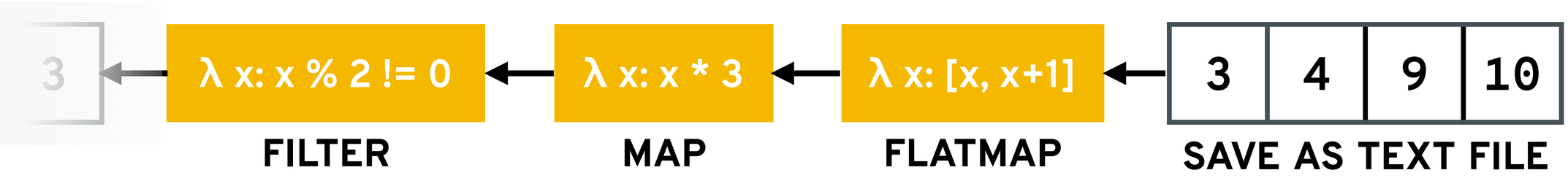

1
2
3

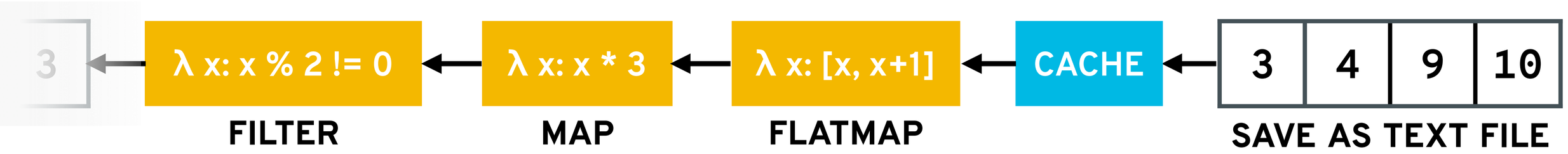

1
2
3
λ x: x % 2 != 0
FILTER

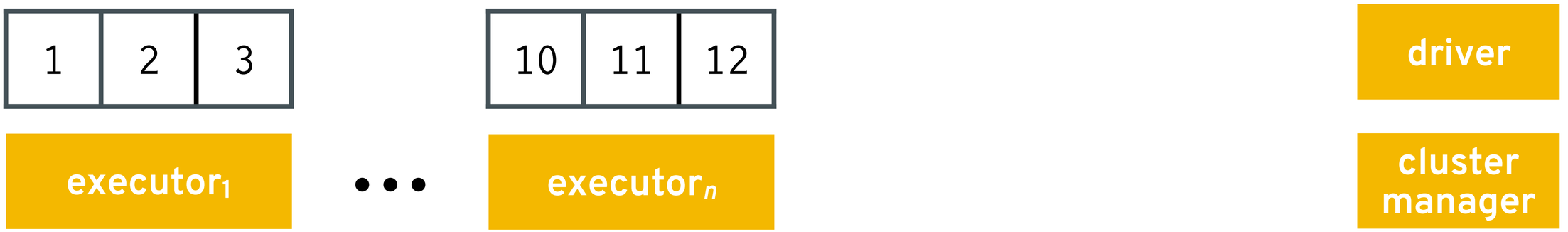

3
λ x: x % 2 != 0
FILTER

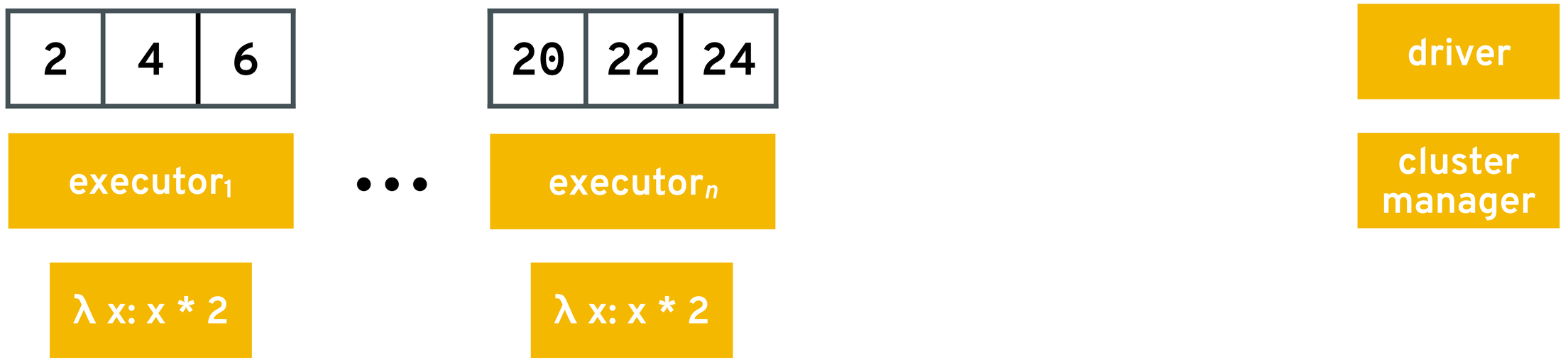

3
λ x: x % 2 != 0
λ x: x * 3
FILTER
MAP

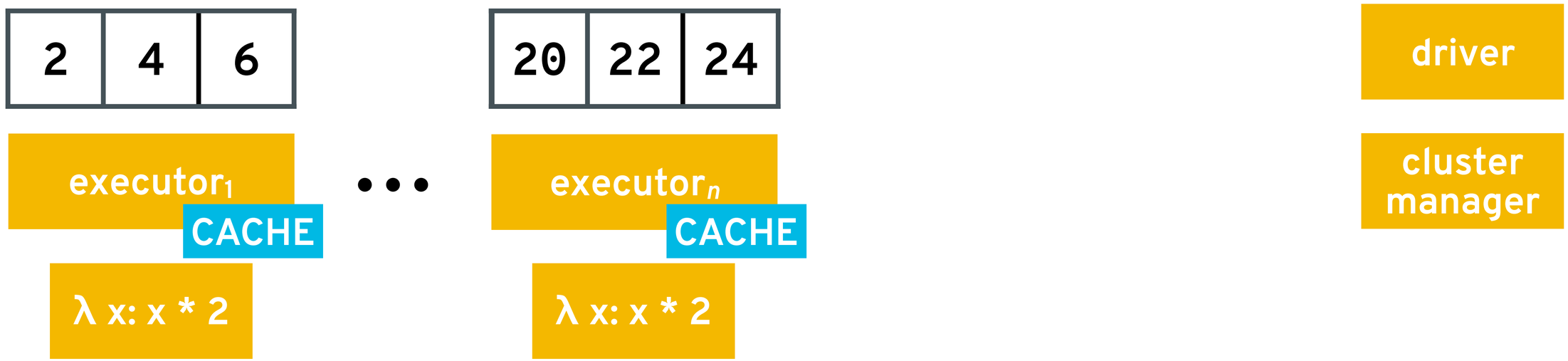

3
λ x: x % 2 != 0
λ x: x * 3
λ x: [x, x+1]
FILTER
MAP
FLATMAP


3
λ x: x % 2 != 0
FILTER
λ x: x * 3
MAP
λ x: [x, x+1]
FLATMAP
3
4
9
10
SAVE AS TEXT FILE


3
λ x: x % 2 != 0
FILTER
λ x: x * 3
MAP
λ x: [x, x+1]
FLATMAP
CACHE
3
4
9
10
SAVE AS TEXT FILE


1
2
3
10
11
12
executor1
executorn
driver
cluster manager


2
4
6
20
22
24
executor1
executorn
λ x: x * 2
λ x: x * 2
driver
cluster manager


2
4
6
20
22
24
executor1
CACHE
λ x: x * 2
executorn
CACHE
λ x: x * 2
driver
cluster manager

Example: word count

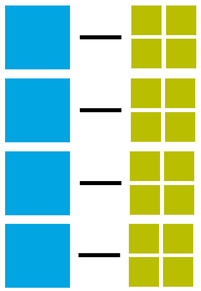
```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```

Example: word count

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```

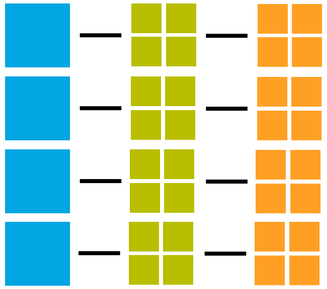
Example: word count

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
             .map(lambda w: (w, 1))
             .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```



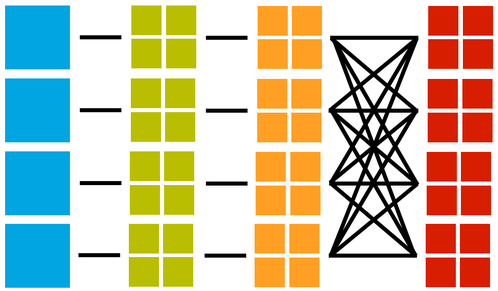
Example: word count

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```



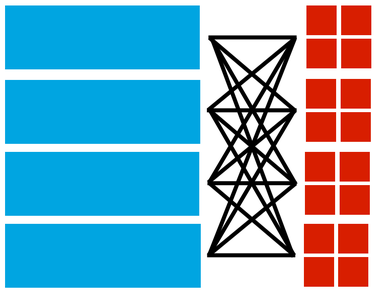
Example: word count

```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```



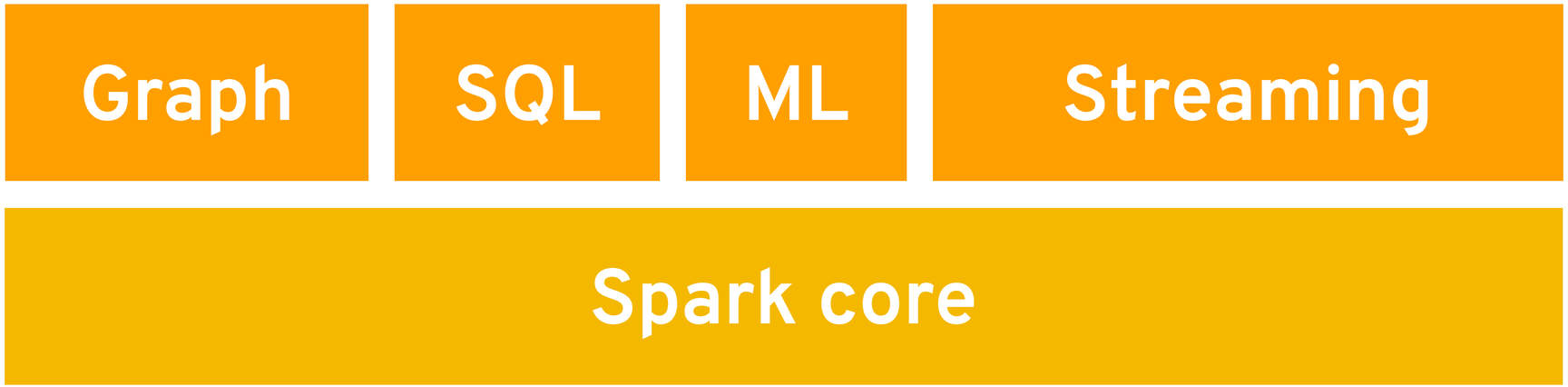
Example: word count

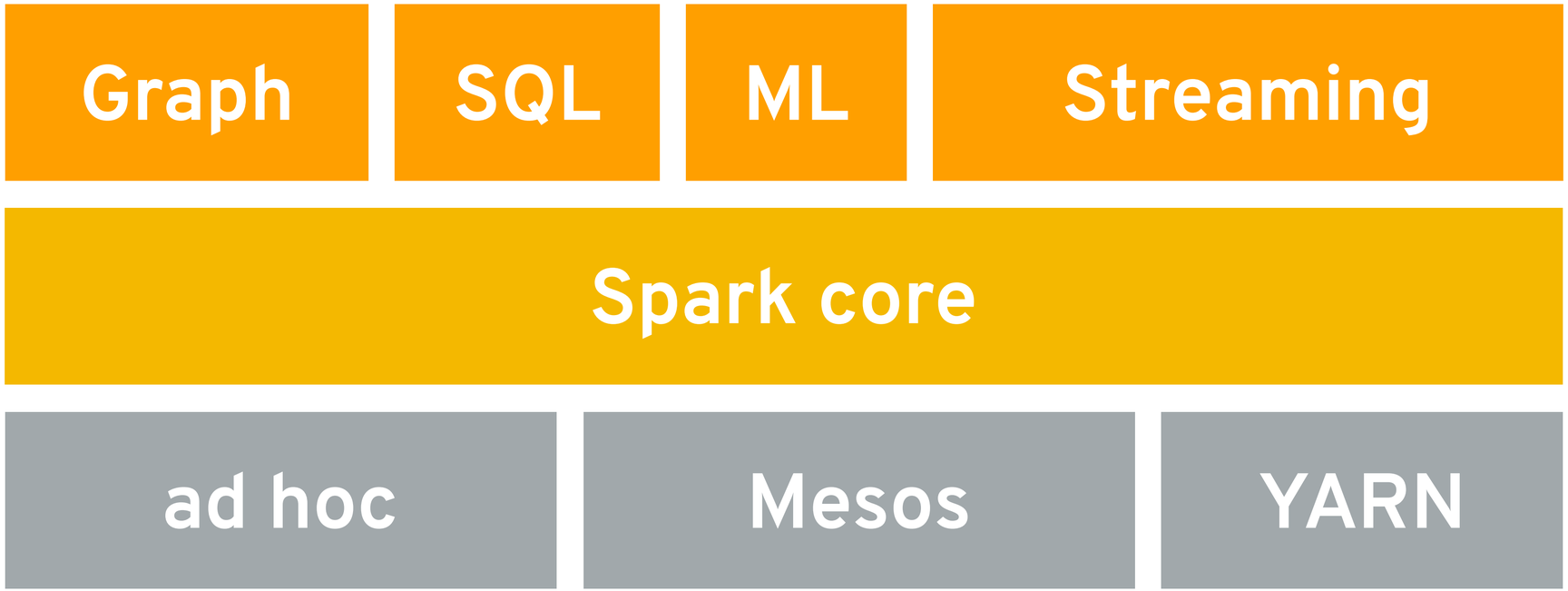
```
file = sc.textFile("file://...")

counts = file.flatMap(lambda l: l.split(" "))
              .map(lambda w: (w, 1))
              .reduceByKey(lambda x, y: x + y)

# computation actually occurs here
counts.saveAsTextFile("file://...")
```

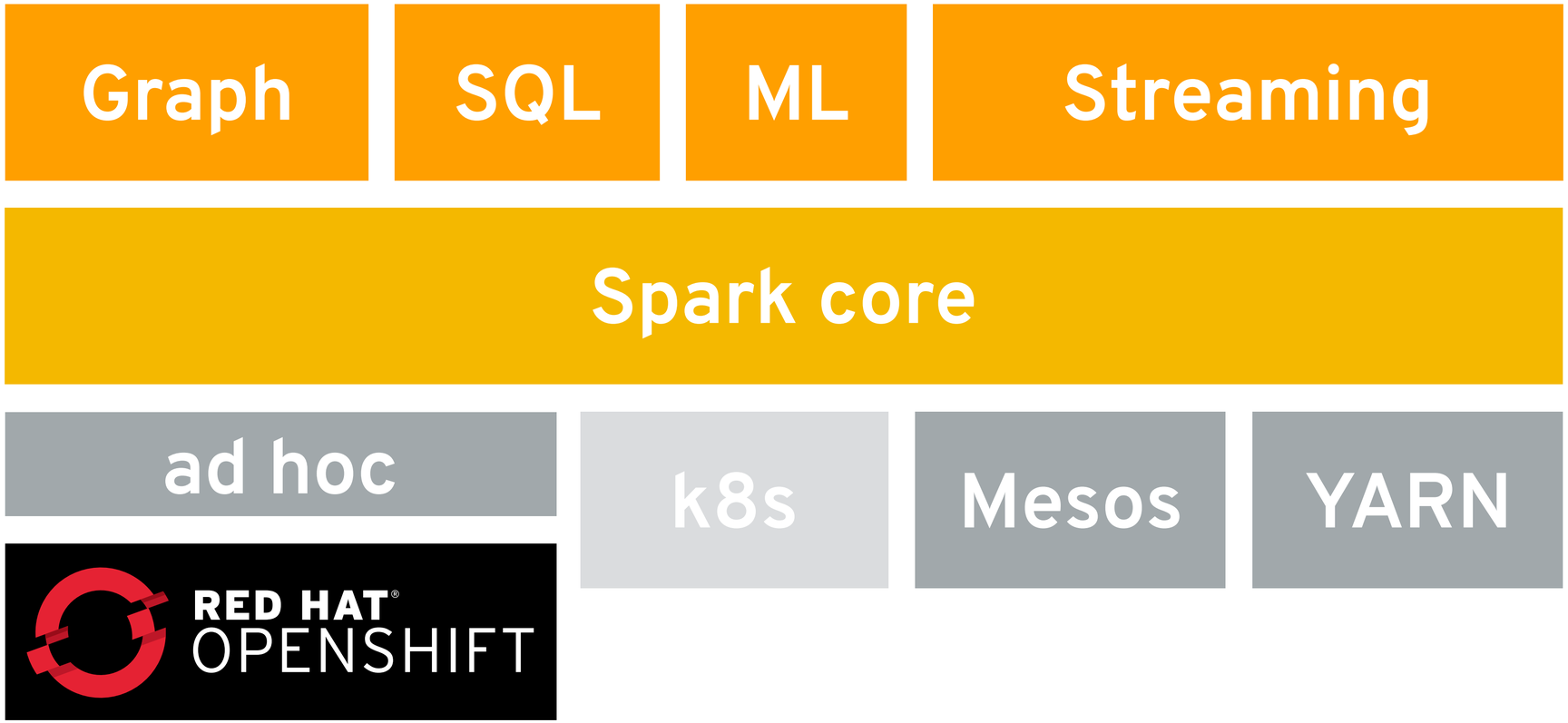


BEYOND THE RDD

Spark core


Graph
SQL
ML
Streaming
Spark core


Graph
SQL
ML
Streaming
Spark core
ad hoc
Mesos
YARN


Graph
SQL
ML
Streaming
Spark core
ad hoc
k8s
Mesos
YARN
RED HAT®
OPENSHIFT

Machine learning with Spark

Support code for **feature engineering** and **learning pipelines.**

Many **parallel implementations** of **classic algorithms** for machine learning tasks: **dimensionality reduction, classification, regression, clustering, recommendation engines,** etc.

Parallel optimization primitives (gradient descent, etc.) and **linear algebra** to **implement your own algorithms.**

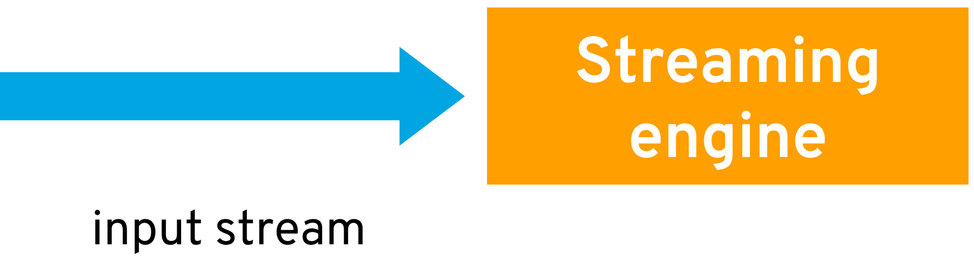
Streaming data

Goal: use the **same abstraction** for batch and “streaming” (micro-batch) data by **dividing a stream** into **many small RDDs.**

input stream

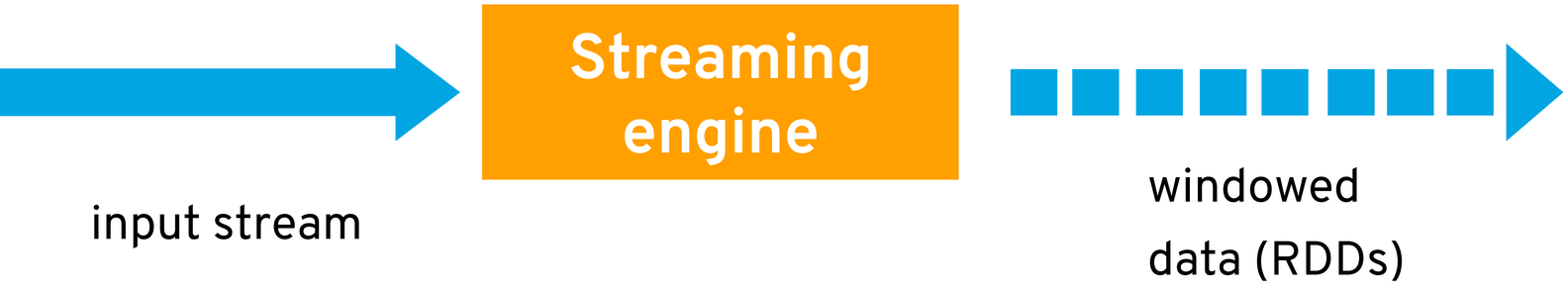
Streaming data

Goal: use the **same abstraction** for batch and “streaming” (micro-batch) data by **dividing a stream** into **many small RDDs.**



Streaming data

Goal: use the **same abstraction** for batch and “streaming” (micro-batch) data by **dividing a stream** into **many small RDDs.**

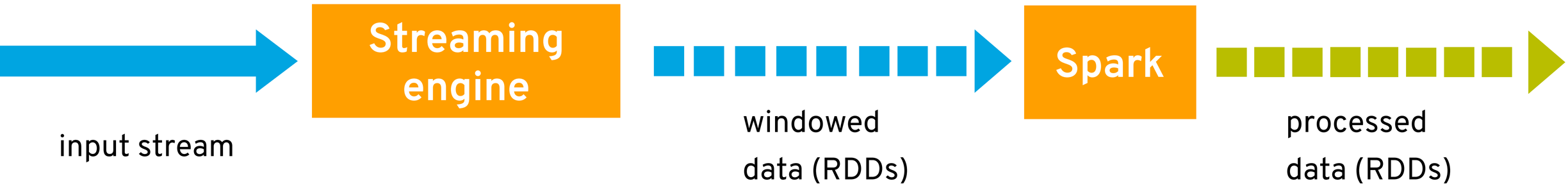


Streaming data

Goal: use the **same abstraction** for batch and "streaming" (micro-batch) data by **dividing a stream** into **many small RDDs.**

Streaming data

Goal: use the **same abstraction** for batch and "streaming" (micro-batch) data by **dividing a stream** into **many small RDDs.**



Structured queries

The capacity to run arbitrary code in RDDs is powerful but comes with **an important tradeoff**: Spark can't rearrange RDD programs to improve their performance.

Writing Spark programs with a **query DSL** allows Spark to generate **optimized execution plans**.

Query planning

```
hugeCollection
    .join(anotherHugeCollection)
    .filter(lambda (n, (a, b)): ultraRare(a) and ultraRare(b))
```

Query planning

```
hugeCollection
    .join(anotherHugeCollection)
    .filter(lambda (n, (a, b)): ultraRare(a) and ultraRare(b))
```

```
hugeCollection.filter(lambda a: ultraRare(a))
    .join(anotherHugeCollection.filter(lambda a: ultraRare(a)))
```

Structured query in Spark

SQL interface (unchecked syntax or semantics)

```
SELECT word, COUNT(*) FROM words GROUP BY word
```

Data frame interface (semantics checked at run-time)

```
words.groupBy('word').count()
```

Dataset interface (mostly checked at compile-time)

Questions & hands-on exercises