UNIVERSIDAD DE CHILE

Minería de Datos

Welcome to the Machine Learning class

Valentin Barriere

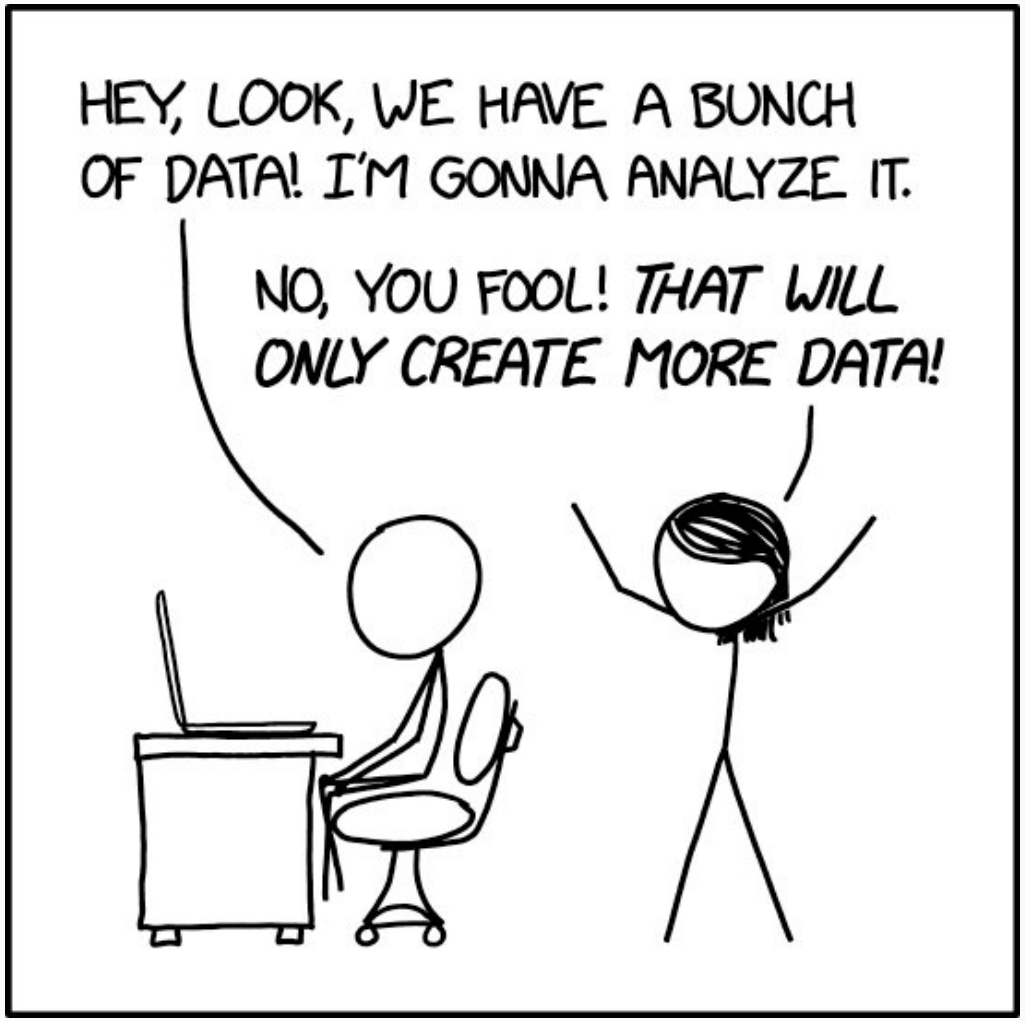
Universidad de Chile – DCC

CC5205, Fall 2025

Data I

Outline: Data

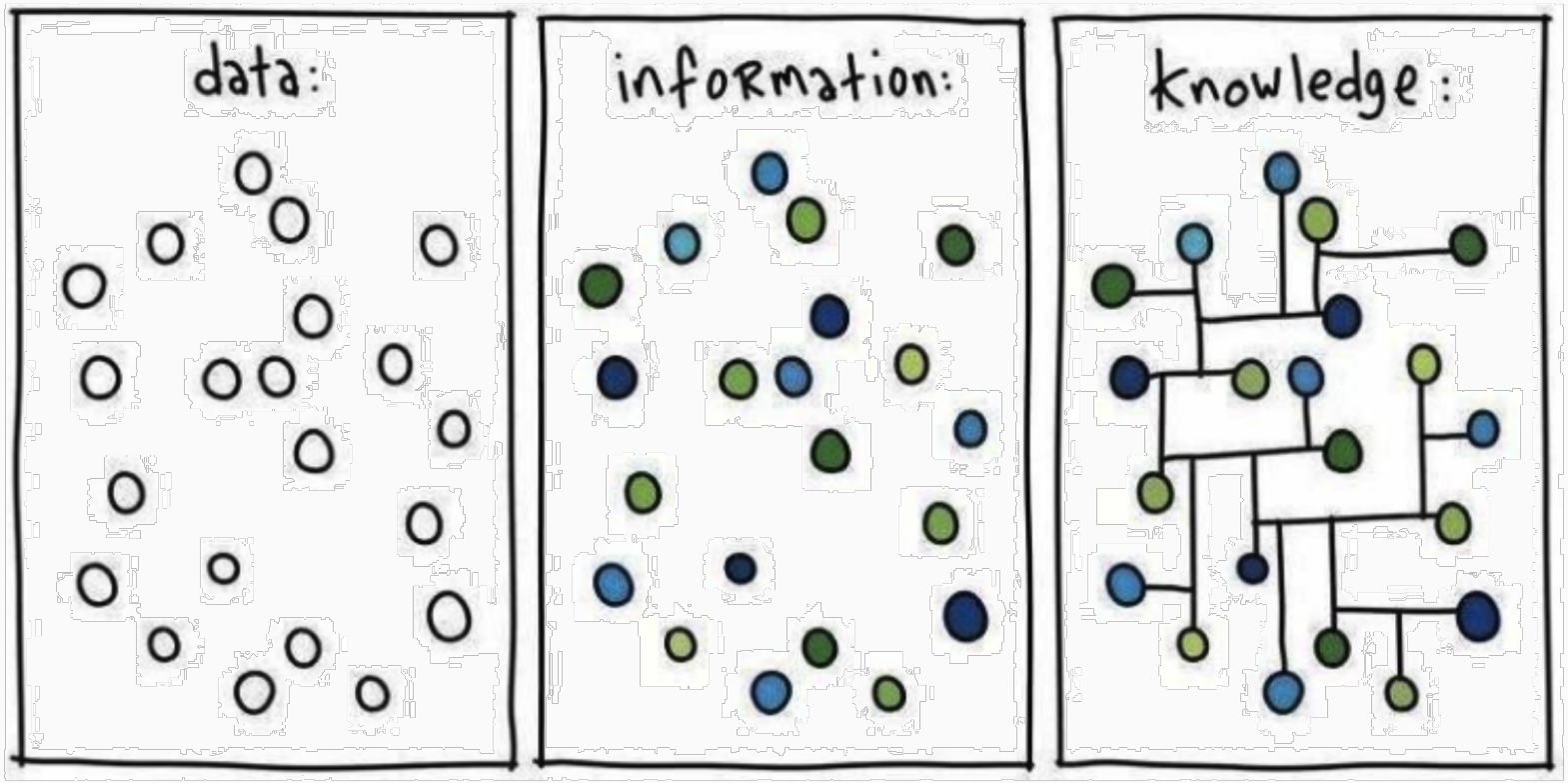
Data


HEY, LOOK, WE HAVE A BUNCH OF DATA! I'M GONNA ANALYZE IT.
NO, YOU FOOL! THAT WILL ONLY CREATE MORE DATA!

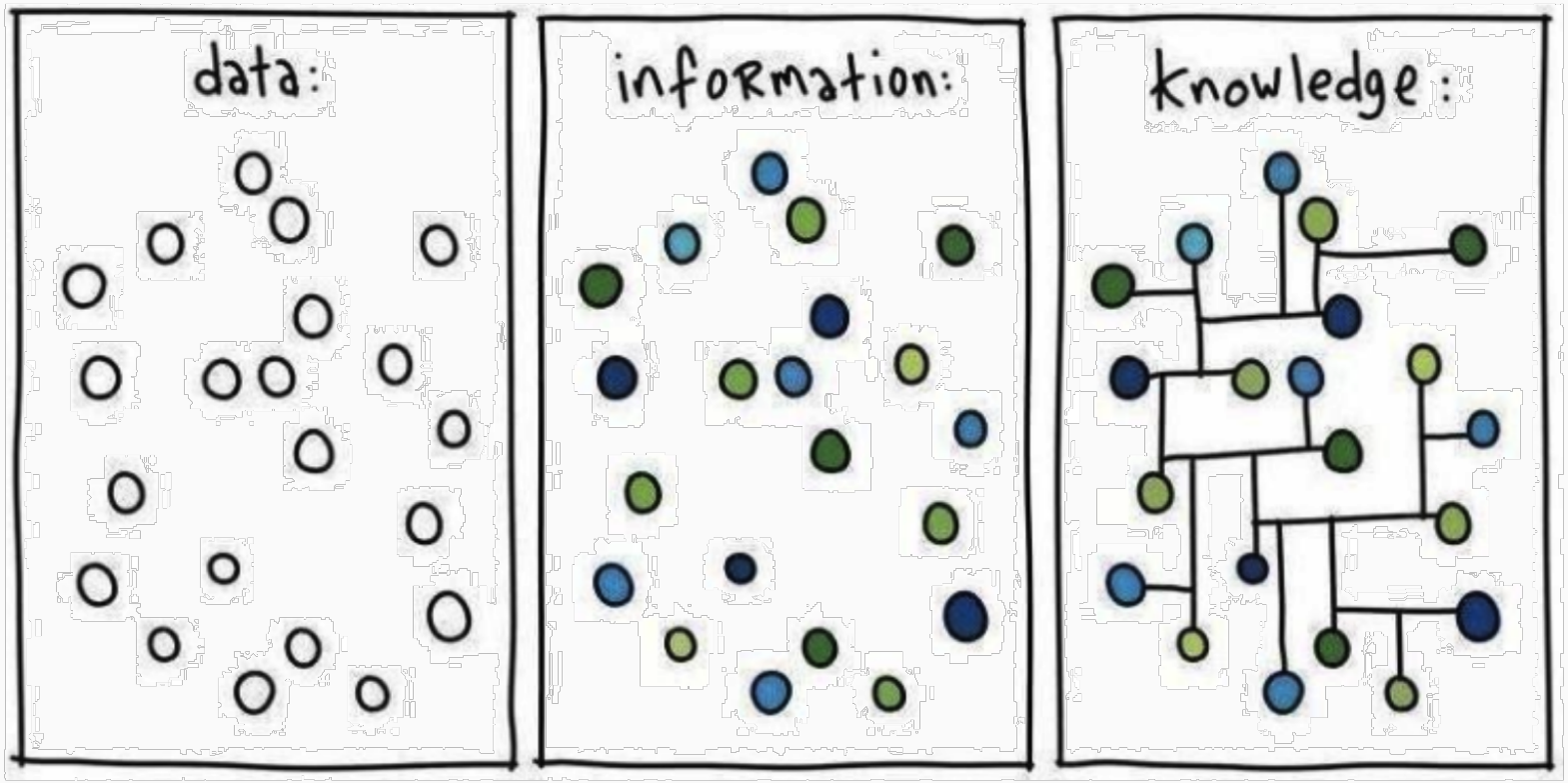
Generalities

Outline : Generalities

Data: What Do We Mean by Data?



Data: What Do We Mean by Data?

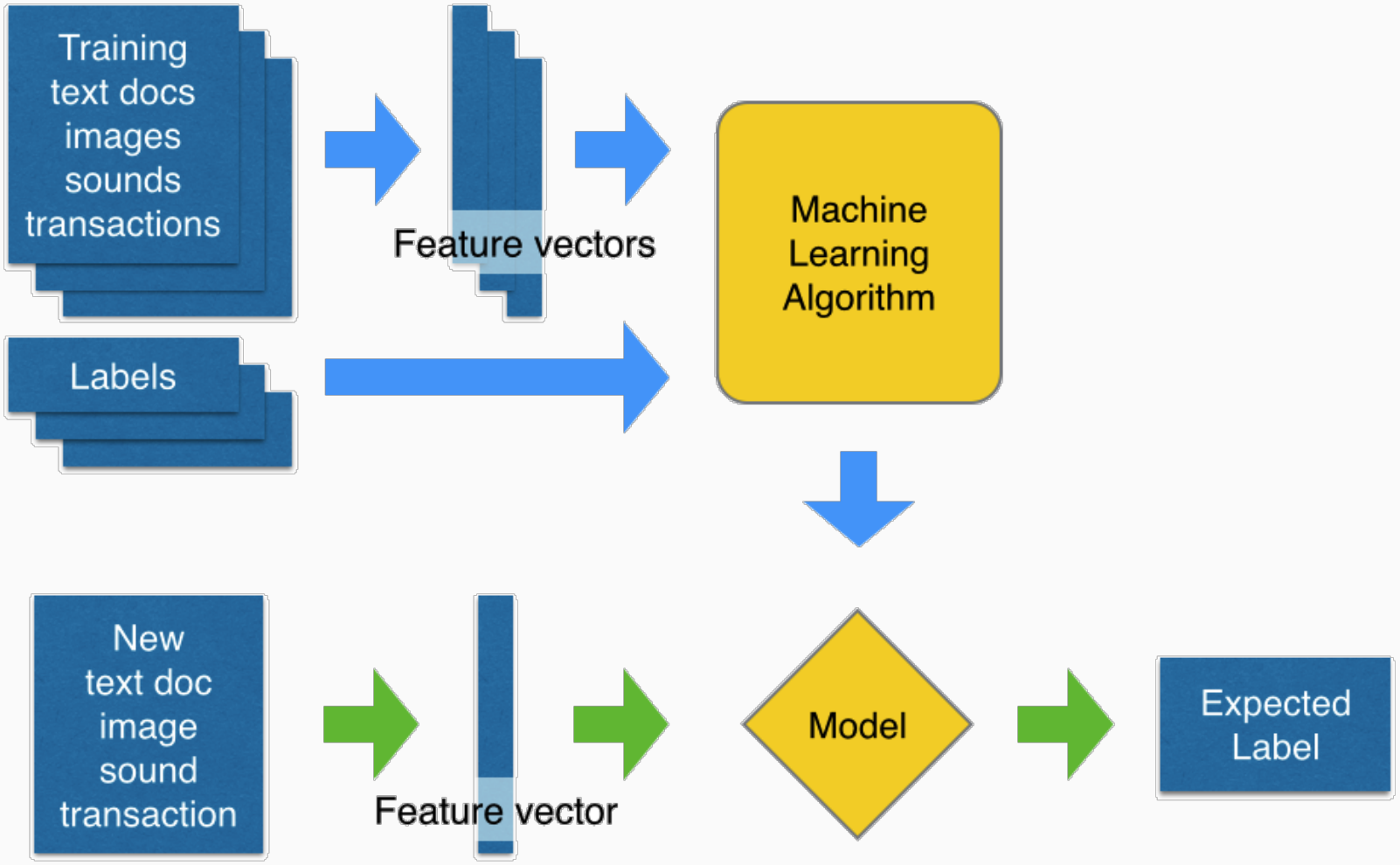


Juan Pérez obtuvo un 5.8 en la segunda prueba de historia

Juan Pérez obtuvo mejor nota que el 75% del resto del curso

Juan Pérez es un buen alumno

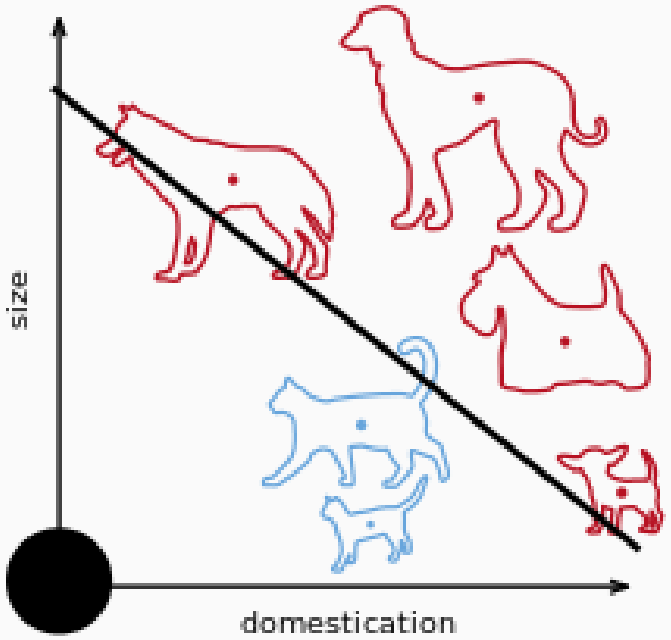
Supervised Learning



Predictive Modeling Data Flow

Vectorization

- To detect structures in a space, the data must be in vector form, i.e., a set of numerical variables (one per dimension).
- It is necessary to extract encoded representations from each document to transform the documents into vectors.



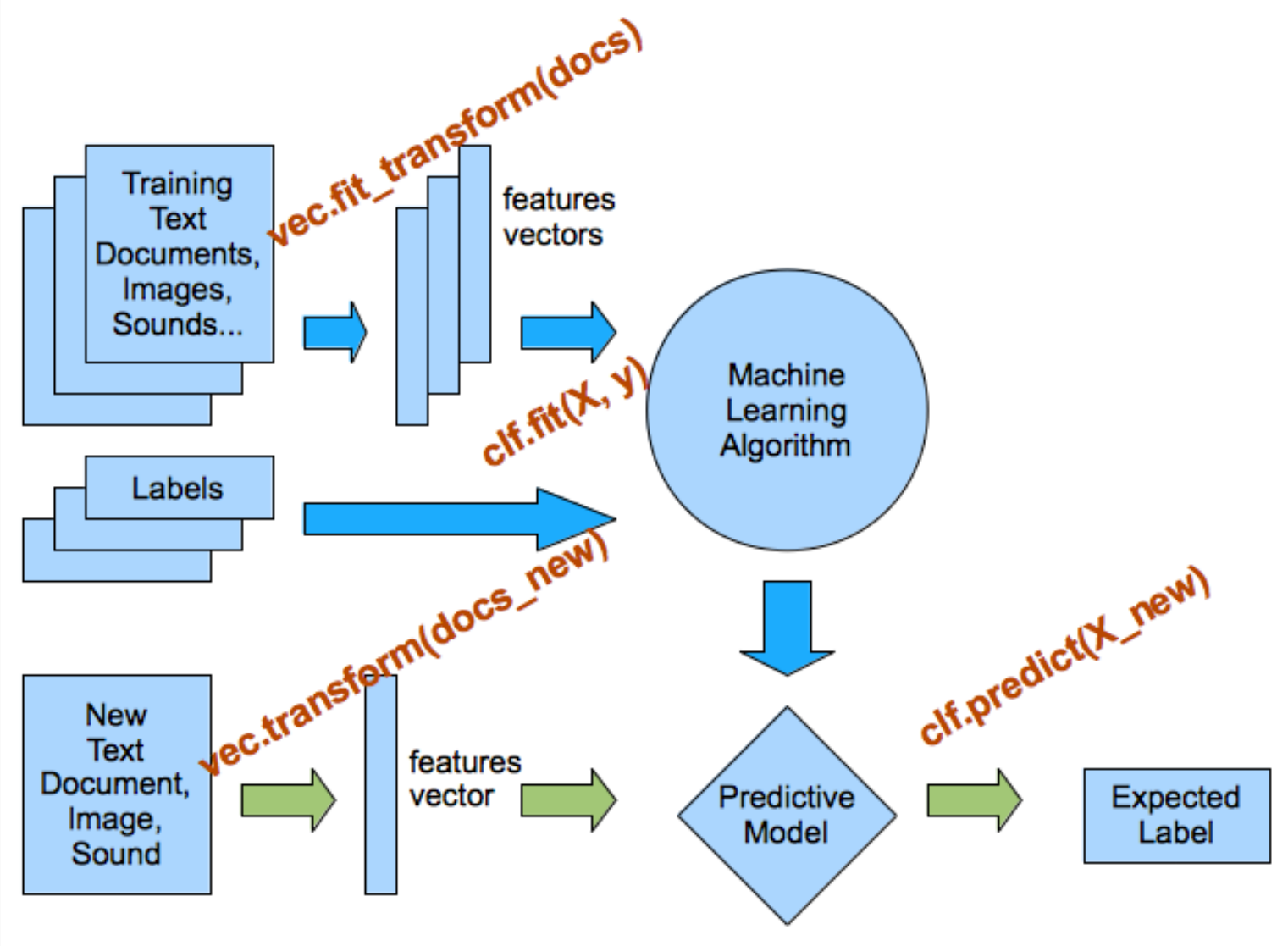
Scikit-learn: ML Library in Python

A very user-friendly machine learning library:
https://scikit-learn.org/

A cool user manual:
https://scikit-learn.org/stable/user_guide.html

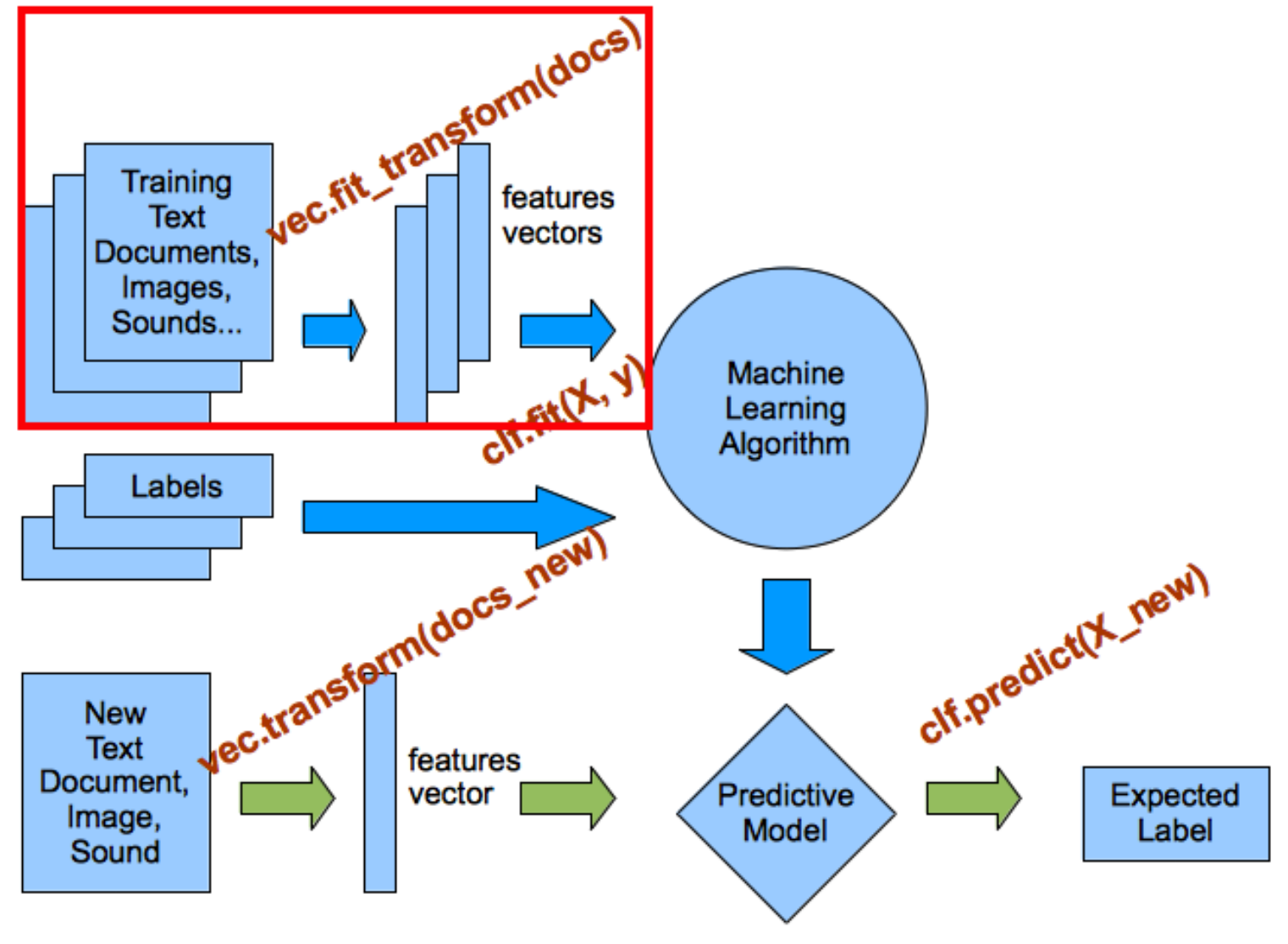
Scikit-learn: General Workflow

Normalized functions to learn quickly with clear code:

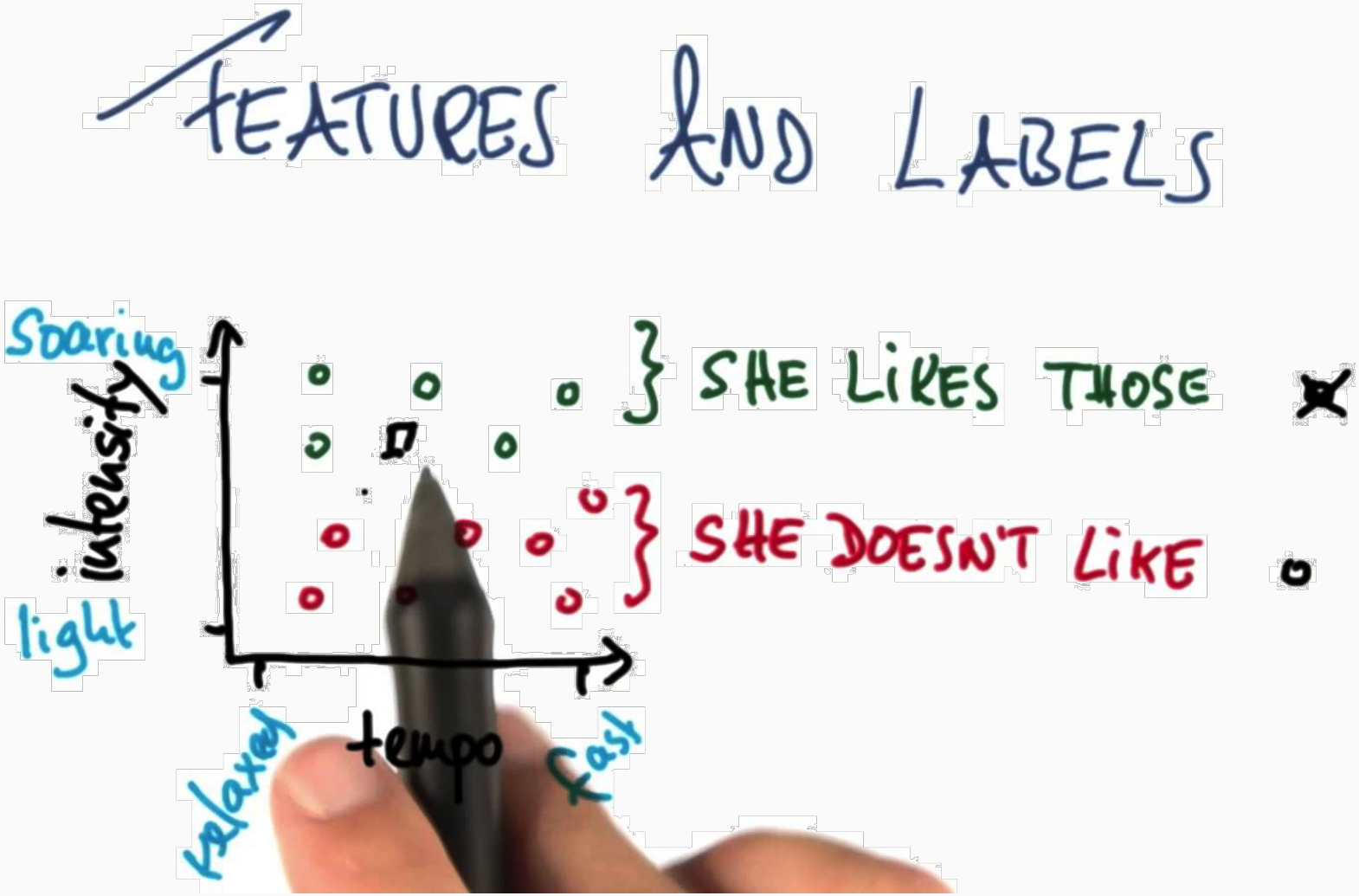


Scikit-learn: General Workflow

Normalized functions to learn quickly with clear code:

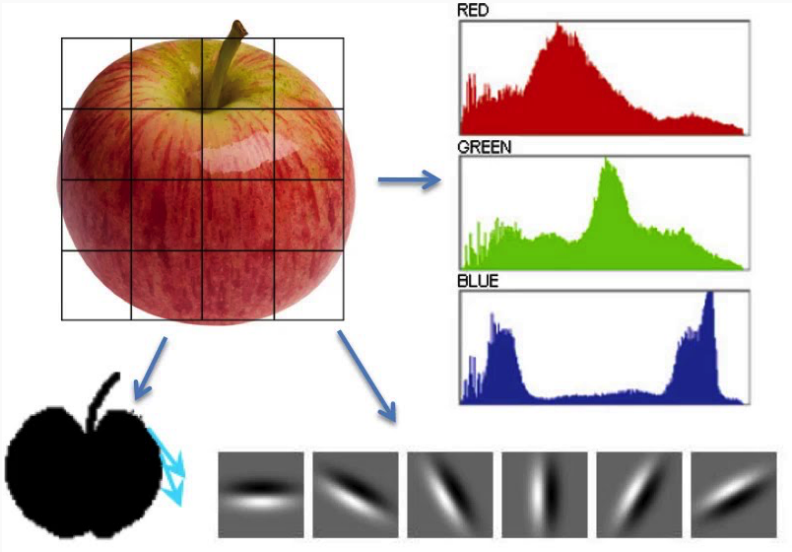


Features and Labels

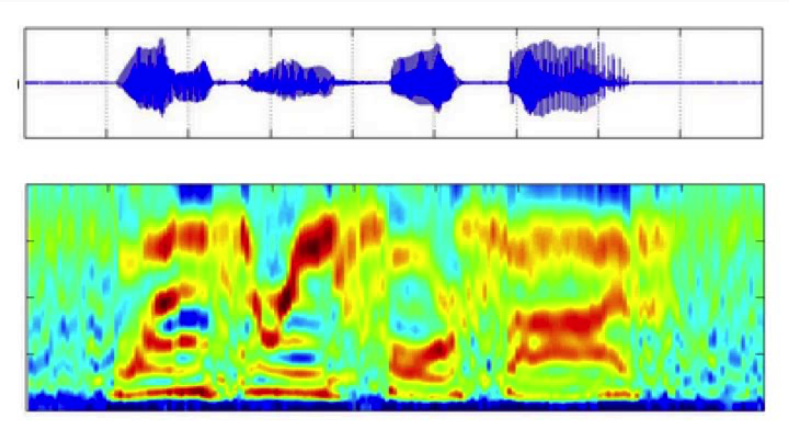


Features

- Images: Color histogram



- Sounds: Time-frequency representation



Tabular Data

Today, we will mainly discuss structured data, i.e., in tabular form.

- A collection of facts such as numbers, words, measurements, or descriptions of things.
- Generally used for analysis.

Types of Data

Outline : Types of Data

Outline : Qualitative vs. Quantitative

Quantitative vs. Qualitative Data

What are the benefits of qualitative versus quantitative data?

Quantitative vs. Qualitative Data

What are the benefits of qualitative versus quantitative data?

Easier to interpret (can be higher-level) but with some loss of information

An Example with Tabular Data: Titanic Dataset

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Figure 1: A structured database

Types of Tabular Data

Attribute Type		Description	Examples	Operations
Categorical (Qualitative)	Nominal	The values of a nominal attribute are just different names; i.e., nominal values provide only enough information to distinguish one object from another. $(=, \neq)$	zip codes, employee ID numbers, eye color, gender	mode, entropy, contingency correlation, χ^2 test
	Ordinal	The values of an ordinal attribute provide enough information to order objects. $(<, >)$	hardness of minerals, $\{good, better, best\}$, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Numeric (Quantitative)	Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. $(+, -)$	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, t and F tests
	Ratio	For ratio variables, both differences and ratios are meaningful. $(\times, /)$	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Outline : Feature Extraction with scikit-learn

Feature Extraction

The Concept of Transforming Data into Vectors

Different types of possible extraction:

- It is nominal (a single class, multi-class: a tag)
- It is an ordinal value (normalization)

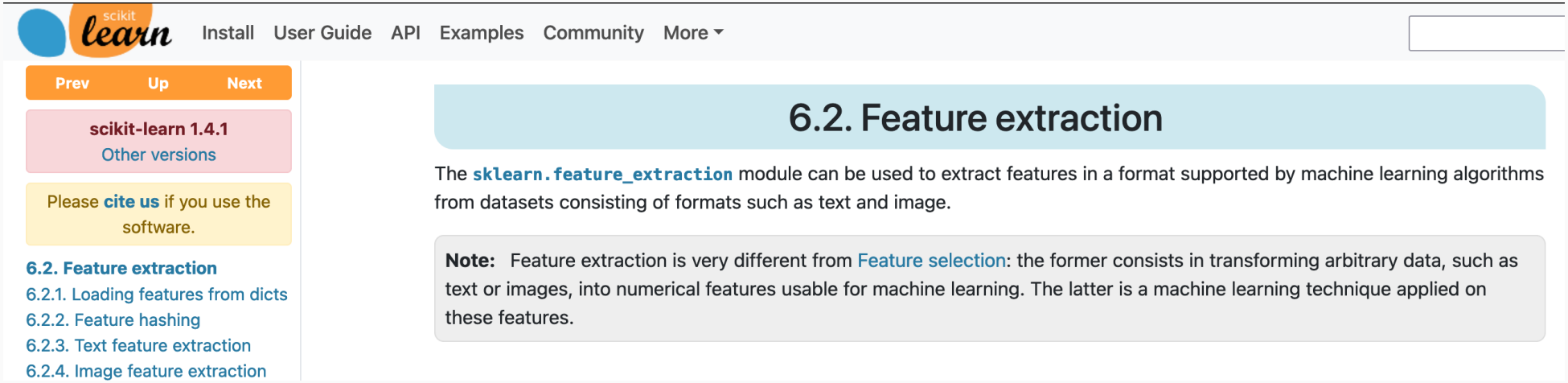
```
>>> measurements = [
...     {'city': 'Dubai', 'temperature': 33.},
...     {'city': 'London', 'temperature': 12.},
...     {'city': 'San Francisco', 'temperature': 18.},
... ]

>>> from sklearn.feature_extraction import DictVectorizer
>>> vec = DictVectorizer()

>>> vec.fit_transform(measurements).toarray()
array([[ 1.,  0.,  0., 33.],
       [ 0.,  1.,  0., 12.],
       [ 0.,  0.,  1., 18.]])

>>> vec.get_feature_names_out()
array(['city=Dubai', 'city=London', 'city=San Francisco', 'temperature'], ...)
```

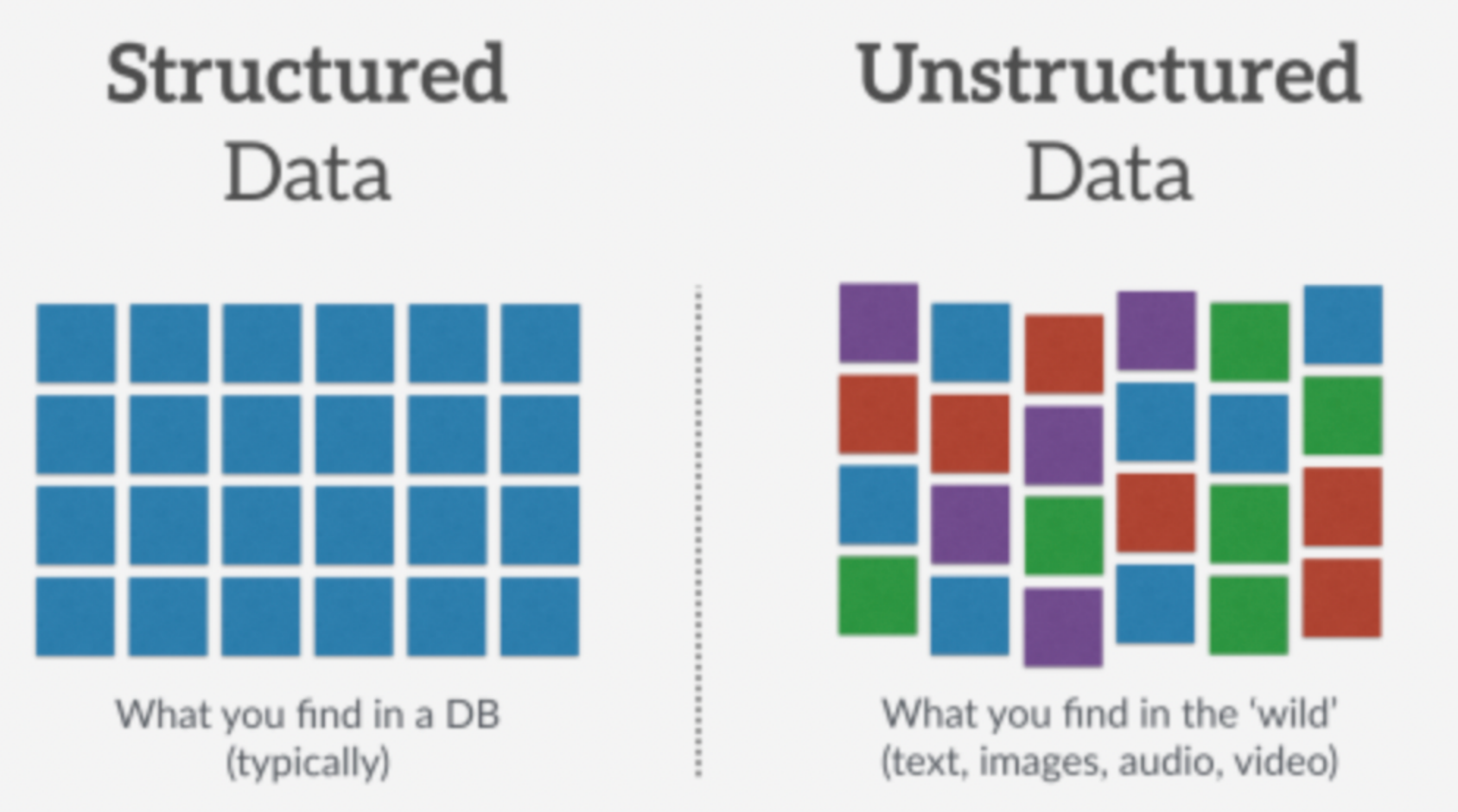
Feature Extraction in scikit-learn



scikit-learn class:
https://scikit-learn.org/stable/modules/feature_extraction.html

Outline : Structured vs. Unstructured Data

Unstructured Data



Unstructured Data

Nombre mascota	Especie	Raza	Edad	Sexo	Peso
Doris	Canino	Schnauzer	3	Hembra	8
Clotilde	Canino	Mestizo	14	Hembra	7

Se ha ingresado el paciente **Doris**, de especie **canino**. Nació aproximadamente el 2017, por lo que tiene **3 años**. Su manto es color **sal y pimienta** y patas blancas. Este es su primer control, pesando **8kg** aproximadamente.

General Characteristics

Dimensionality

Number of features; the curse of dimensionality relates to problems when working with many dimensions (preprocessing: dimensionality reduction)

Sparsity

Most dimensions are 0 in the data, which can be advantageous as you only need to store the non-zero (1) values. (e.g., the Web graph and its links).

Resolution

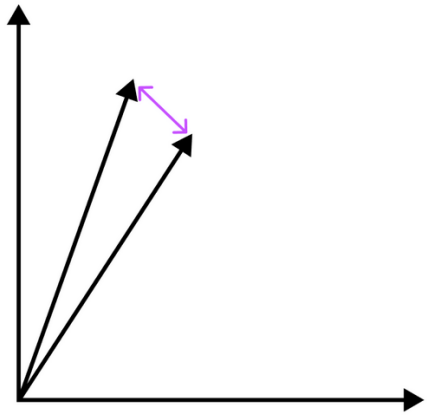
The term speaks for itself (e.g., variations in atmospheric pressure are noticeable hourly but not over months).

Outline : Distance in the Space

Distances and Similarities

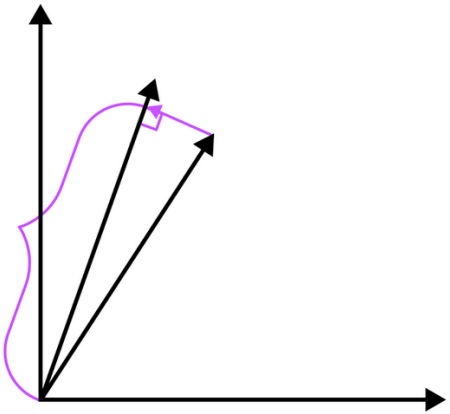
Euclidean Distance

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$



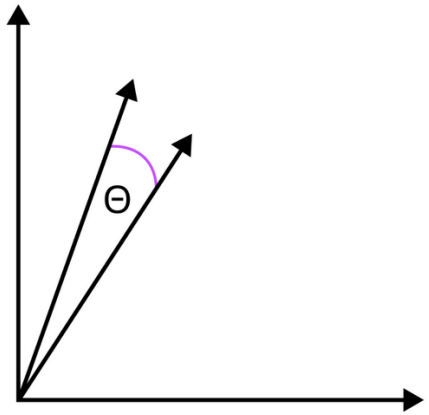
Inner Product

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$



Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$



Pearson Correlation and Dot Product

A similarity measure can be computed with a dot product:

$$\cos(\mathbf{X}, \mathbf{X'}) = \frac{< \mathbf{X} | \mathbf{X'} >}{||\mathbf{X}|| \cdot ||\mathbf{X'}||} = \frac{1}{||\mathbf{X}|| \cdot ||\mathbf{X'}||} \sum_{i=1}^{N} X_i * X_i'$$

Big Five Personality Traits

5 dimensions: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. Rated from -1 to 1 in each dimension. More info about the Big 5

User Recommendation

Each user is represented as an N-dimensional vector, with each dimension corresponding to a product. For every product purchased/liked/viewed, the vector will have a 1 in that dimension.

Pearson correlation: cosine of the centered data: $\cos(\mathbf{X}_{\text{cent}}, \mathbf{X}'_{\text{cent}})$

Pearson Correlation and Dot Product

A similarity measure can be computed with a dot product:

$$\cos(\mathbf{X}, \mathbf{X'}) = \frac{< \mathbf{X} | \mathbf{X'} >}{||\mathbf{X}|| \cdot ||\mathbf{X'}||} = \frac{1}{||\mathbf{X}|| \cdot ||\mathbf{X'}||} \sum_{i=1}^{N} X_i * X_i'$$

Big Five Personality Traits

5 dimensions: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. Rated from -1 to 1 in each dimension. More info about the Big 5

User Recommendation

Each user is represented as an N-dimensional vector, with each dimension corresponding to a product. For every product purchased/liked/viewed, the vector will have a 1 in that dimension.

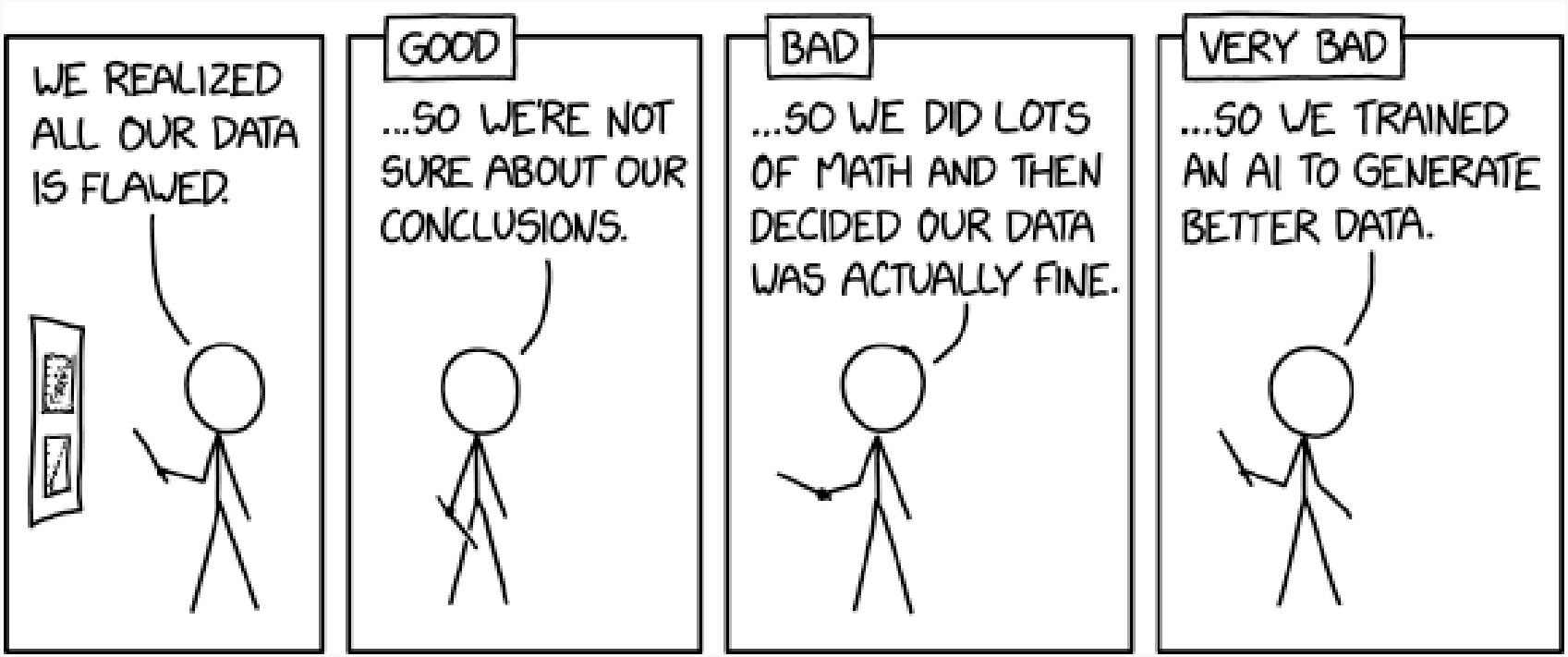
Pearson correlation: cosine of the centered data: $\cos(\mathbf{X}_{\text{cent}}, \mathbf{X}'_{\text{cent}})$

Think further: what else could we do with text?

Quality

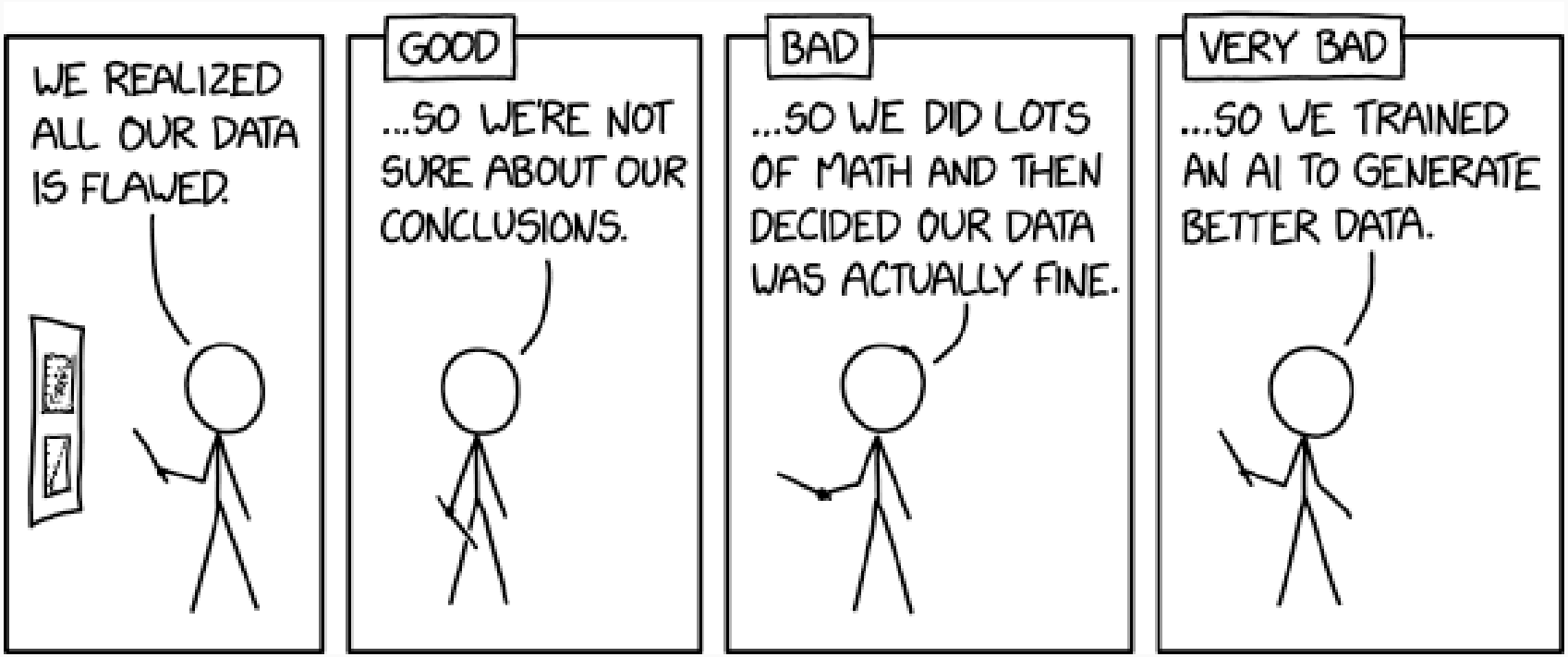
Outline : Quality

Data Quality



Types of errors: noise and outliers, missing values, duplicate data

Data Quality



Types of errors: noise and outliers, missing values, duplicate data

There are solutions!

Outline : Noise

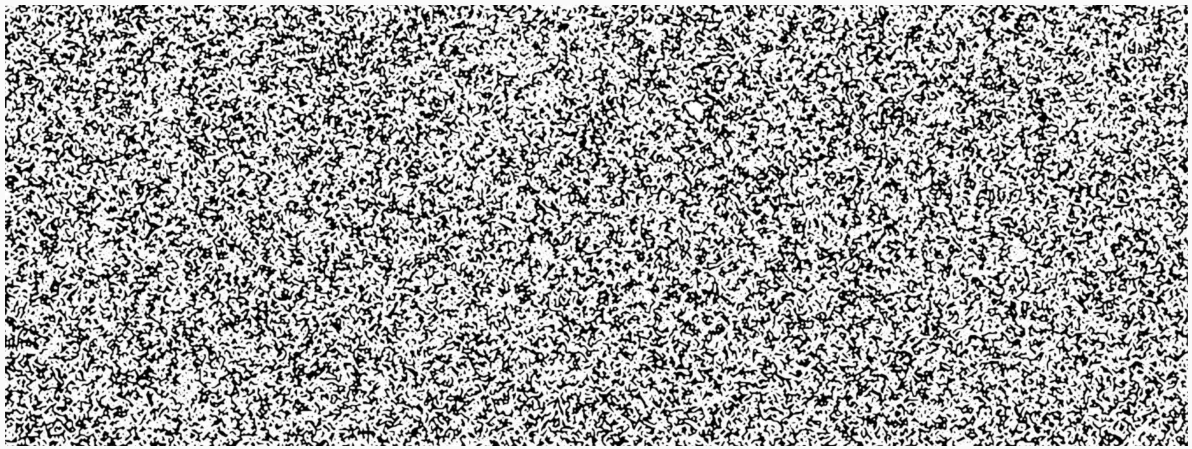
Statistical Noise

Definition

Random irregularity found in any real-world data. It has no pattern. These errors are generally inevitable and unpredictable.

It generally consists of errors and residuals:

- Errors can be due to measurement or sampling; they are avoidable and repeat, distorting the data.
- The residual comes from the modeling of the data.



Statistical Noise: Residual

Residual

The residual of the observed data is the difference between your observed value (again, that measured data point) and the predicted value; not the "true" value itself, but the point in space where your theory suggests the data point should lie.

- It is difficult (or impossible) to represent the entire reality with a finite set of observations.
- We represent a phenomenon with a finite-size vector, which may be a reduction of the initial phenomenon—an approximation of reality.
- **There will always be a component of noise that cannot be modeled.**

Example of Noise

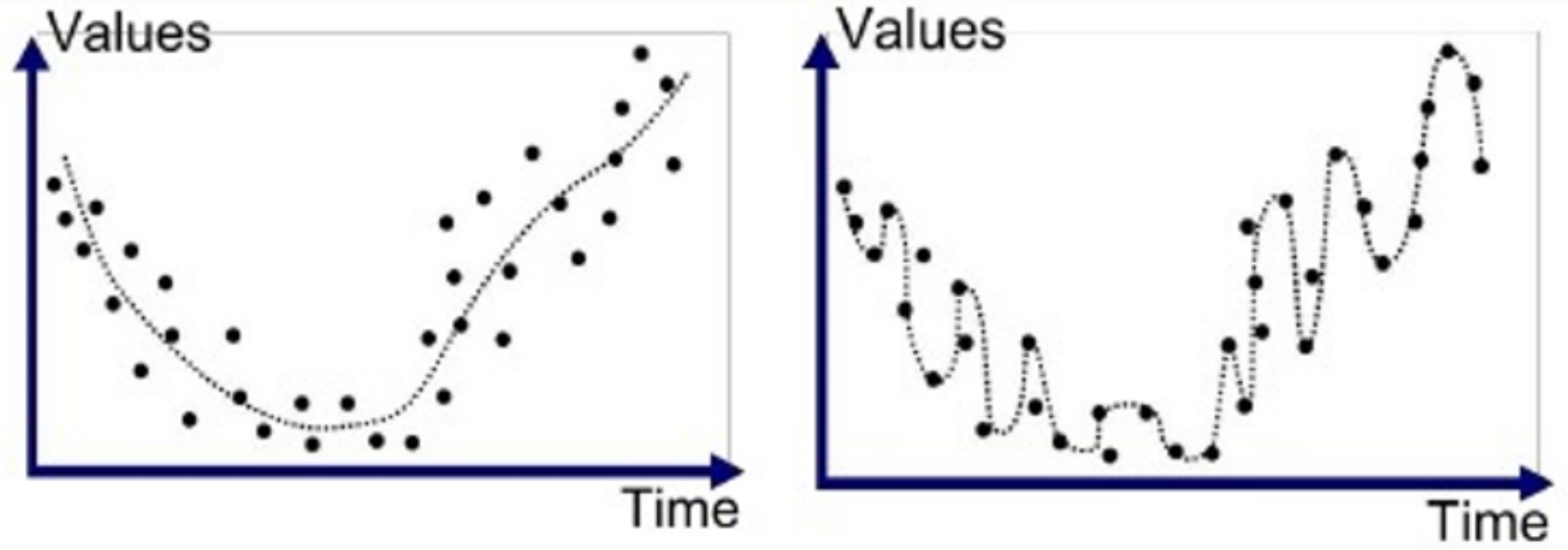


Figure 2: Noise in a function

If we try to model $Y = 3X_1 - 2X_2^2 + \epsilon$ using X_1 and X_2, we will not succeed.

Denoising Example

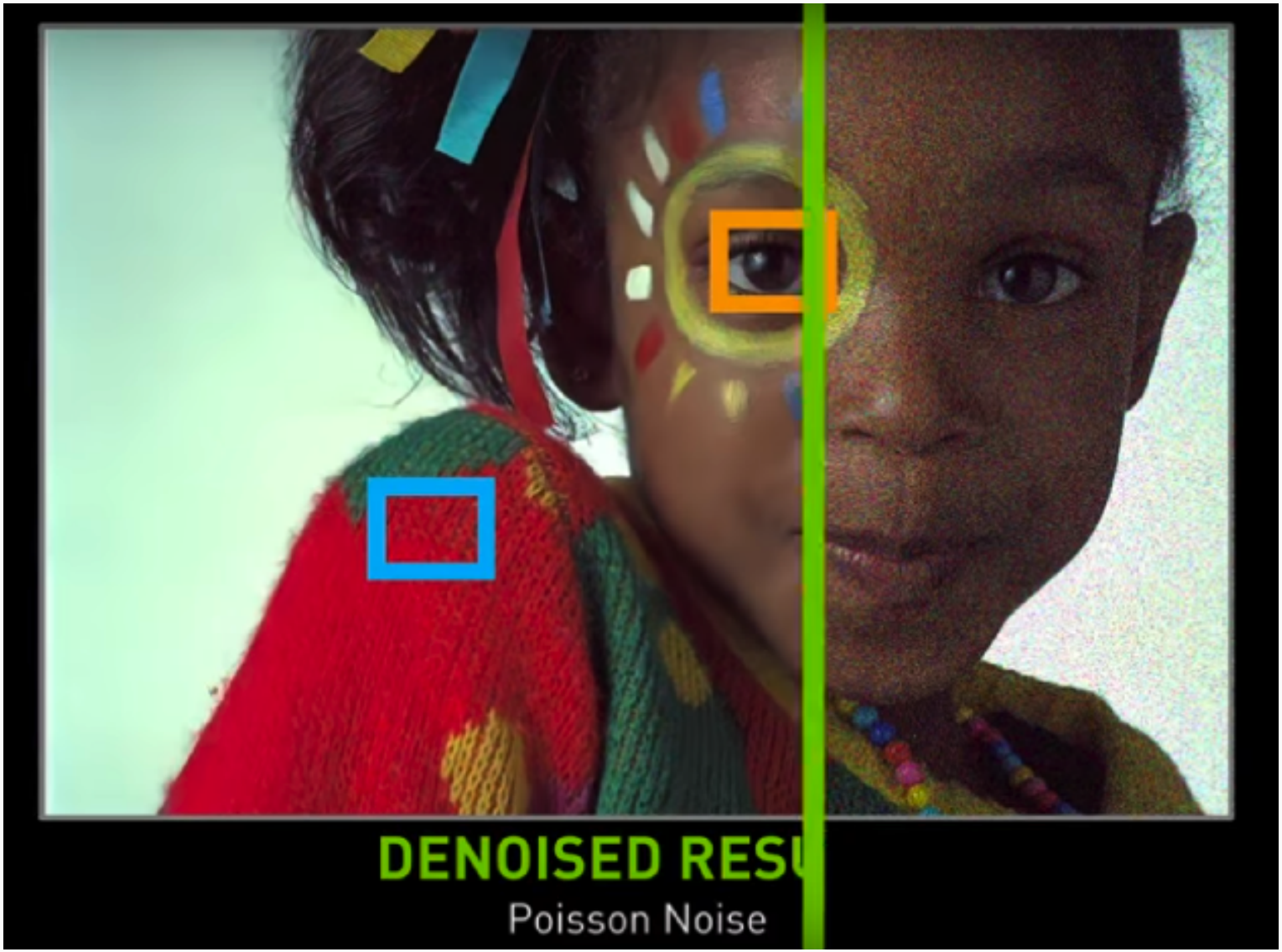


Figure 3: An application of an Image Denoiser

Outline : Outliers

Outliers

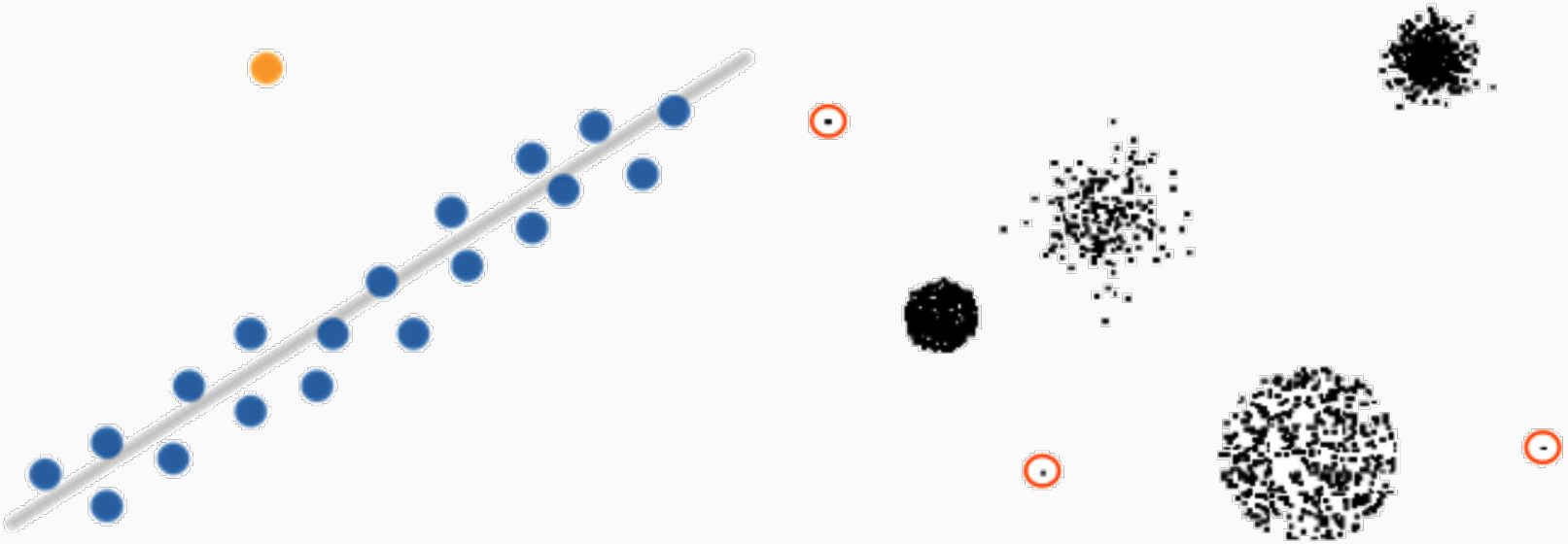


Figure 4: Outlier data points in various distributions

Outliers

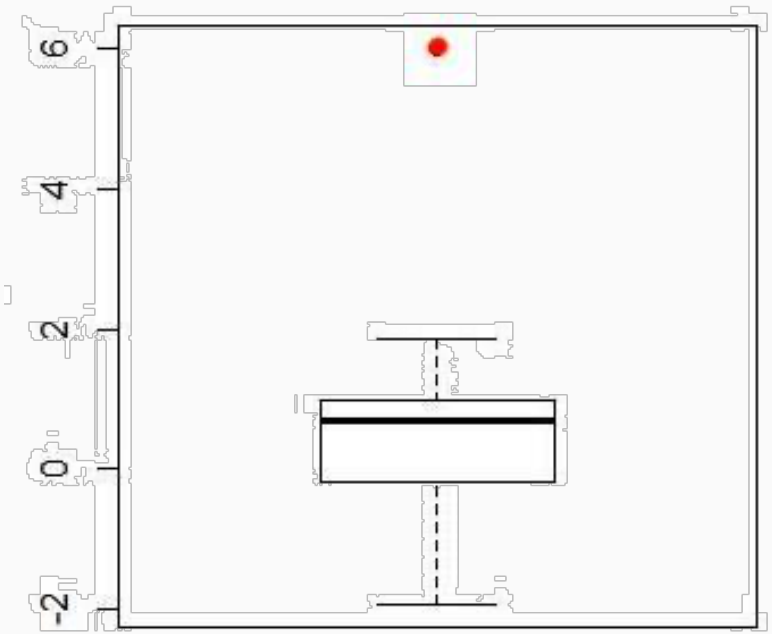


Figure 4: Outlier data points in various distributions

Objects with characteristics considerably different from the majority

Outliers

Noisy Outliers

- Natural variability: tail of the distribution
- Measurement errors: faulty sensor, typos, incorrect entry, etc.
- Exceptional events: unusual or extreme events that do not follow the typical pattern of the data (and are not interesting to model for this distribution)

Good Outliers

Rare events—anomalies that we want to predict! Earthquake, epileptic crisis, machine maintenance, product anomaly, etc.

Detecting an Outlier

- Data visualization (limited)
- Statistical methods: such as the interquartile range (IQR), standard deviation, or z-scores
- Machine learning techniques: anomaly detection algorithms
- Model-based methods: examining points with large errors
- Clustering techniques: Outliers can appear as points that do not cluster well with others. Observing isolated points or very small clusters can help identify outliers.
- Domain validation: Domain validation is crucial. Sometimes what appears as an outlier may actually be a rare but valid data point. It is important to consider the context and nature of the data when interpreting detected outliers.

Anomaly Detection

It can be more precise than detecting an entire sample, though only a part of this example:

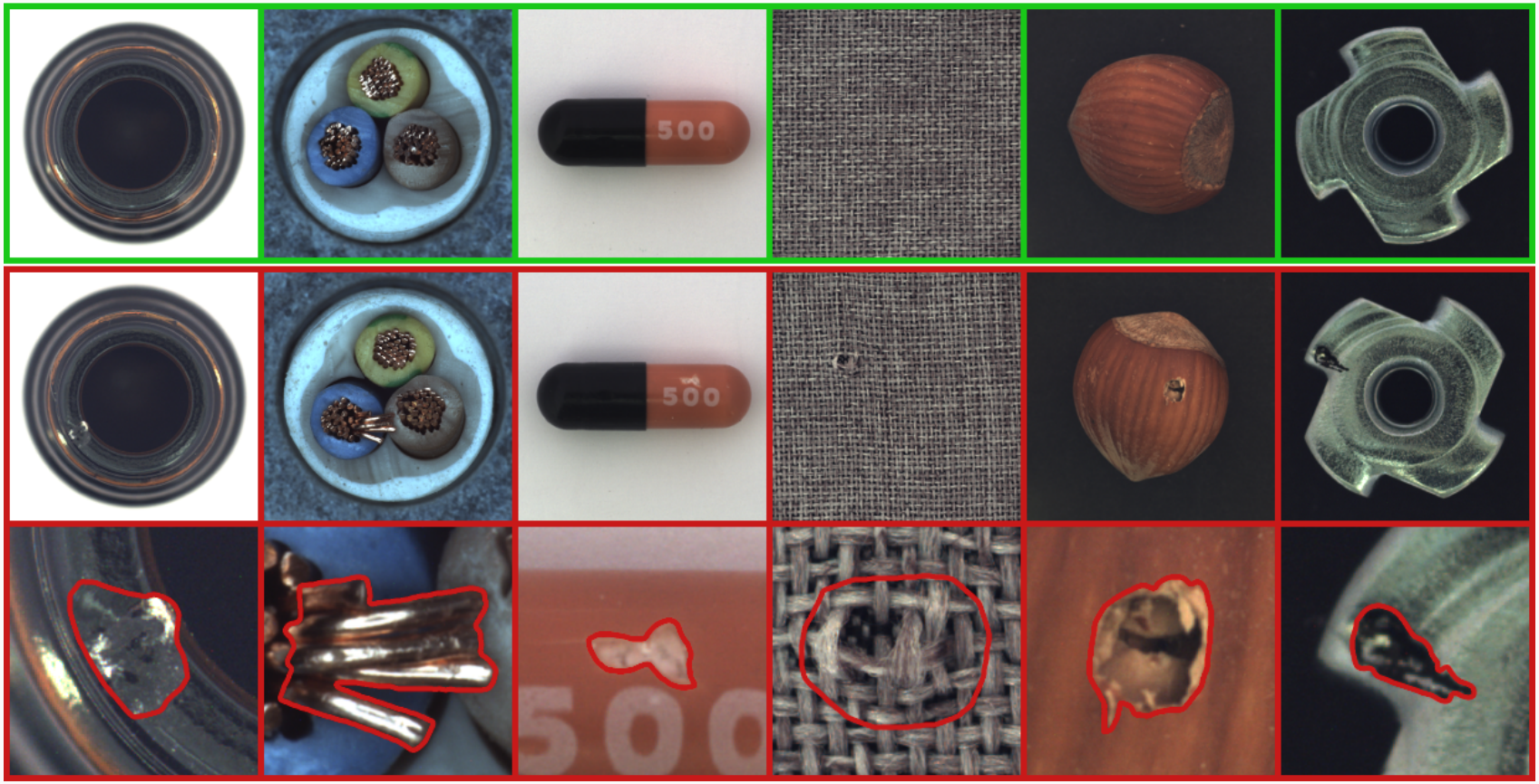


Figure 5: Examples from the MVTEC Anomaly Detection Dataset

CleanLab Will Detect Label Errors

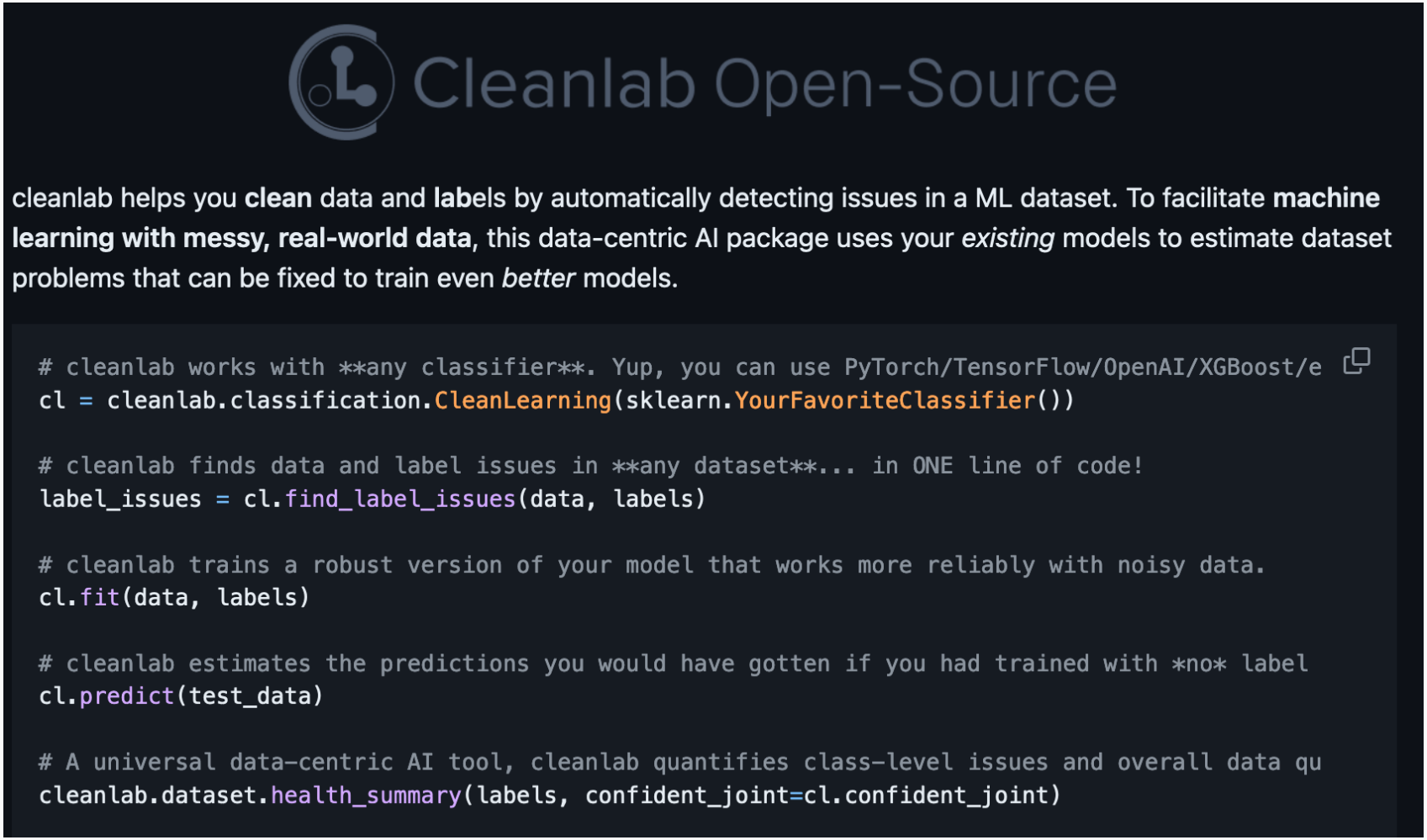


Figure 6: Outlier Label Detection. CleanLab was able to detect many label errors in the Imagenet dataset.

Outline : Missing Values

Missing Values

There are often missing values, and you must address them:

- Removing the record
- Estimating values: using a constant, the mean, the values from neighbors (KNN), or other methods (learning them)
- Ignoring: some algorithms can handle missing values

https://scikit-learn.org/stable/modules/impute.html

Missing Values: with scikit-learn

Functions `SimpleImputer` or `KNNImputer`:

```
>>> import numpy as np
>>> from sklearn.impute import SimpleImputer
>>> imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
>>> # Alternatively: imp_mean = SimpleImputer(missing_values=np.nan,
# strategy='constant', fill_value=0)
>>> imp_mean.fit([[7, 2, 3], [4, np.nan, 6], [10, 5, 9]])
SimpleImputer()
>>> X = [[np.nan, 2, 3], [4, np.nan, 6], [10, np.nan, 9]]
>>> print(imp_mean.transform(X))
[[ 7.   2.   3. ]
 [ 4.   3.5  6. ]
 [10.   3.5  9. ]]
```

`KNNImputer` will take the attribute values from the N nearest examples in the feature space.

Missing Values: Estimators That Handle Them

6.4.7. Estimators that handle NaN values

Some estimators are designed to handle NaN values without preprocessing. Below is the list of these estimators, classified by type (cluster, regressor, classifier, transform):

- **Estimators that allow NaN values for type `cluster`:**
  - HDBSCAN
- **Estimators that allow NaN values for type `regressor`:**
  - BaggingRegressor
  - DecisionTreeRegressor
  - HistGradientBoostingRegressor
  - RandomForestRegressor
  - StackingRegressor
  - VotingRegressor
- **Estimators that allow NaN values for type `classifier`:**
  - BaggingClassifier
  - DecisionTreeClassifier
  - HistGradientBoostingClassifier
  - RandomForestClassifier
  - StackingClassifier
  - VotingClassifier

Figure 7: There are models that handle missing values

Missing Values: An Example

We can use complex algorithms to reconstruct a sample with missing values

Missing Values: An Example

We can use complex algorithms to reconstruct a sample with missing values: **Any ideas?**

Missing Values: An Example

We can use complex algorithms to reconstruct a sample with missing values: Any ideas?

Figure 8: An Image Reconstruction model based on neural networks

Outline : Duplicate Data

Duplicate Data

It is very common to find duplicate data because:

- Human error during data entry.
- It reflects reality (but is not necessarily useful)
- There is a failure in the data collection system: extraction, transformation, and loading tools
- Aggregation of different databases that contain duplicates

Quality controls regarding duplicates are needed during data collection and processing.

Use Case: Large Language Models

Scraping the Whole Web!

To train an LLM from scratch, it is necessary to collect a huge amount of data! That data cannot be completely clean!

- Data scraped from the web is extremely noisy and must be cleaned.
- Markups, syntax errors, etc.—all non-natural language text—are harmful and can prevent convergence.
- Deduplication has been identified as playing a significant role in improving language models (Allamanis, 2019; Lee et al., 2022).
- Repeated data has been shown to be increasingly harmful to model quality as parameter count increases (Hernandez et al., 2022):
  - For a 1B-parameter model, a hundred duplicates are harmful;
  - At 175B, even a few duplicates could have a disproportionate effect.

Cleaning and Preprocessing

Outline : Cleaning and Preprocessing

Outline : Transformations

Data Preprocessing

- Feature creation
- Selection of a subset of features
- Aggregation
- Normalization
- Sampling
- Dimensionality reduction
- Discretization and binarization
- Transformation

Feature Extraction: OneHotEncoder

```
>>> genders = ['female', 'male']
>>> locations = ['from Africa', 'from Asia', 'from Europe', 'from US']
>>> browsers = ['uses Chrome', 'uses Firefox', 'uses IE', 'uses Safari']
>>> enc = preprocessing.OneHotEncoder(categories=[genders, locations, browsers])
>>> # Note that for the 2nd and 3rd features there are missing categorical valu
>>> X = [['male', 'from US', 'uses Safari'],
>>> ['female', 'from Europe', 'uses Firefox']]
>>> enc.fit(X)
OneHotEncoder(categories=[['female', 'male'],
                          ['from Africa', 'from Asia', 'from Europe',
                           'from US'],
                          ['uses Chrome', 'uses Firefox', 'uses IE',
                           'uses Safari']])
>>> enc.transform([['female', 'from Asia', 'uses Chrome']]).toarray()
array([[1., 0., 0., 1., 0., 0., 1., 0., 0., 0.]])
```

Because:

```
>>> enc.categories_
[array(['female', 'male'], dtype=object),
array(['from Europe', 'from US'], dtype=object),
array(['uses Firefox', 'uses Safari'], dtype=object)]
```

More on encoding categorical features

Data Transformations: On the Distribution

Standardization

An estimator may perform poorly if the individual features do not resemble standard normally distributed data: a Gaussian with **zero mean and unit variance**.

Scaling Features to a Range

Generally between 0 and 1. This scaling is motivated by robustness to very small standard deviations in features and the preservation of zero entries in sparse data.

Normalization

Normalization is the process of **adjusting individual samples so that they have unit norm**.

More info here

Data Transformations: Histogram Stretching

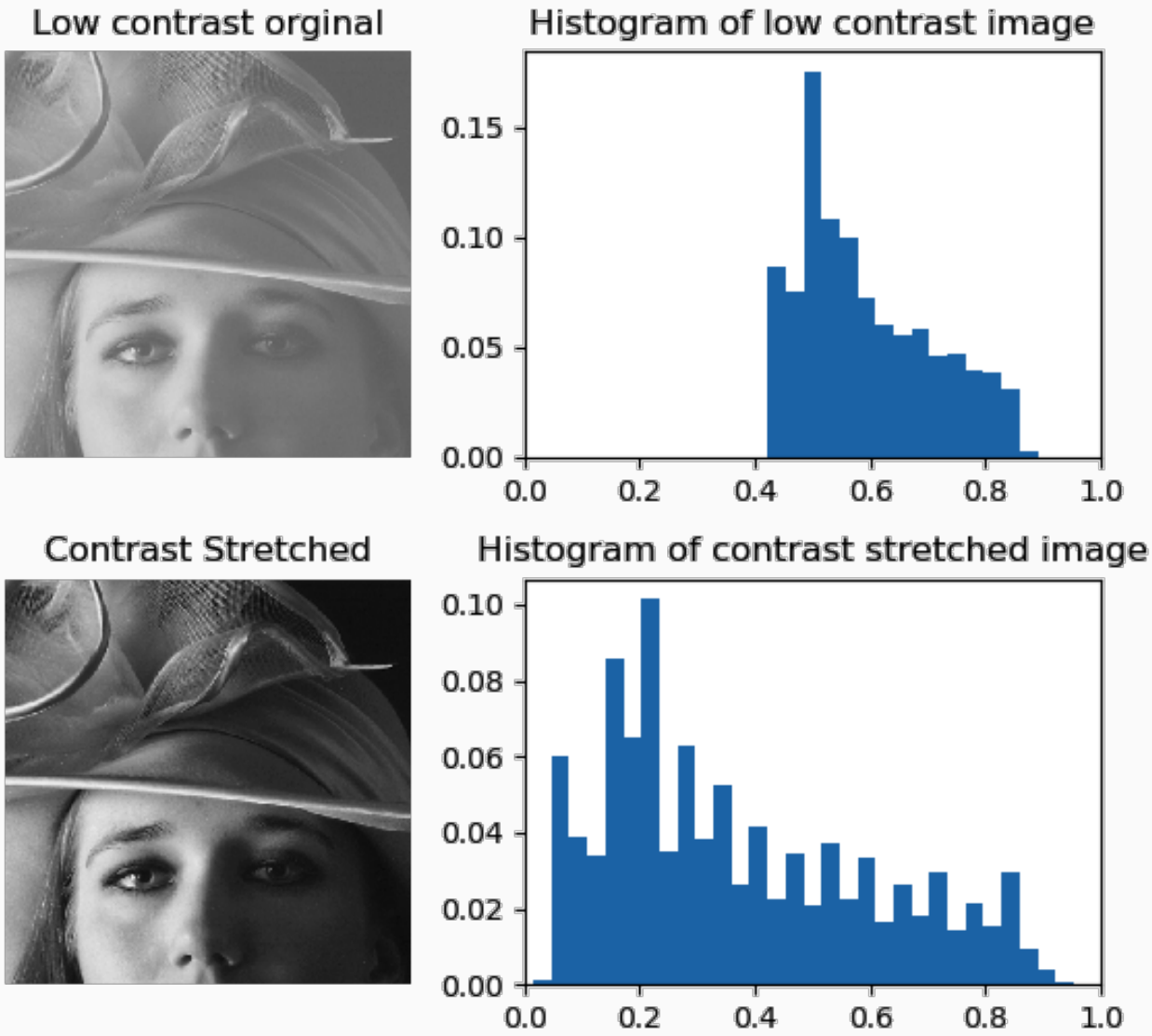
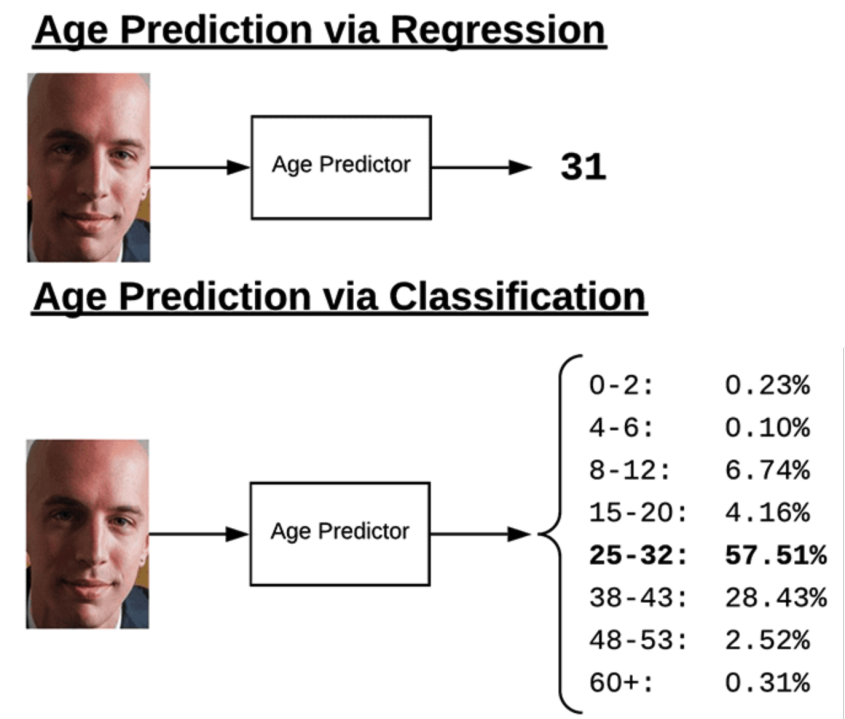


Figure 9: Changing the range of histogram values helps reveal the image

Other Data Transformations

Discretization

Discretization (also known as quantization or binning) allows continuous features to be divided into discrete values (classes). One-hot encoding of discretized features can make a model more expressive while maintaining interpretability.

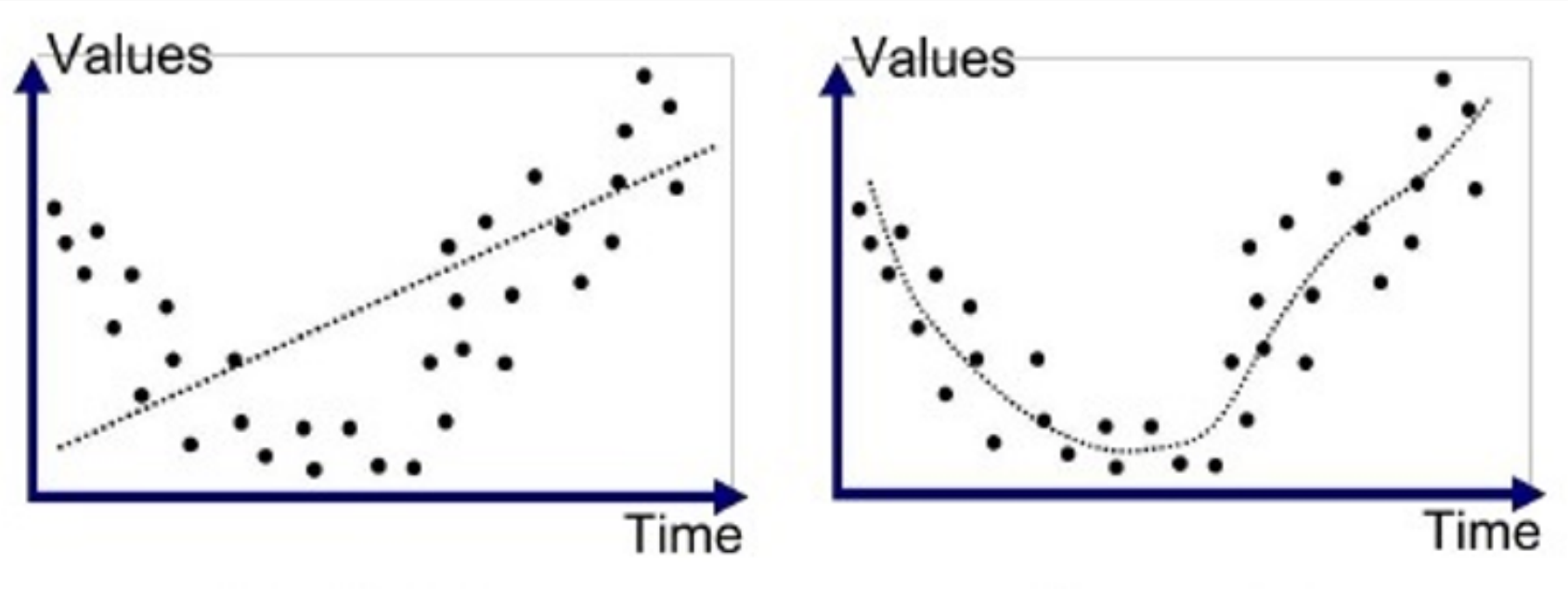


More information here

Other Data Transformations

Polynomial Features

It is often useful to add complexity to a model by considering non-linear features of the input data.



More information [here]

Special Transformations I

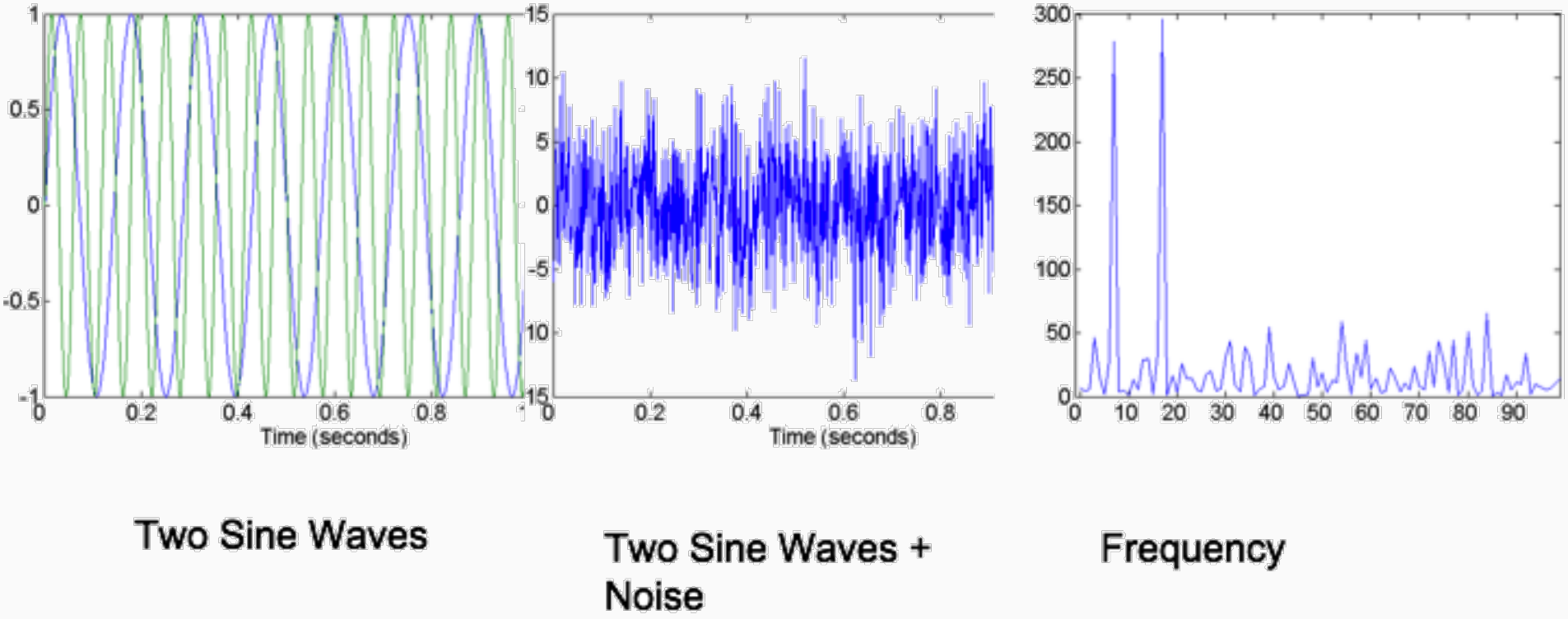


Figure 10: Mapping to a new space that makes more sense can help better characterize the data

Special Transformations II

6.3.8. Custom transformers

Often, you will want to convert an existing Python function into a transformer to assist in data cleaning or processing. You can implement a transformer from an arbitrary function with **FunctionTransformer**. For example, to build a transformer that applies a log transformation in a pipeline, do:

```
>>> import numpy as np
>>> from sklearn.preprocessing import FunctionTransformer
>>> transformer = FunctionTransformer(np.log1p, validate=True)
>>> X = np.array([[0, 1], [2, 3]])
>>> # Since FunctionTransformer is no-op during fit, we can call transform directly
>>> transformer.transform(X)
array([[0.        , 0.69314718],
       [1.09861229, 1.38629436]])
```

Figure 11: Any method can be used to transform the data

https://scikit-learn.org/stable/modules/preprocessing.htmlcustom-transformers

Outline : Sampling

Sampling I

Similarly:

- If the observations we have are insufficient to represent reality, we have a problem.
- If the samples we have in the data do not adequately represent reality, we have a problem.

Sampling II

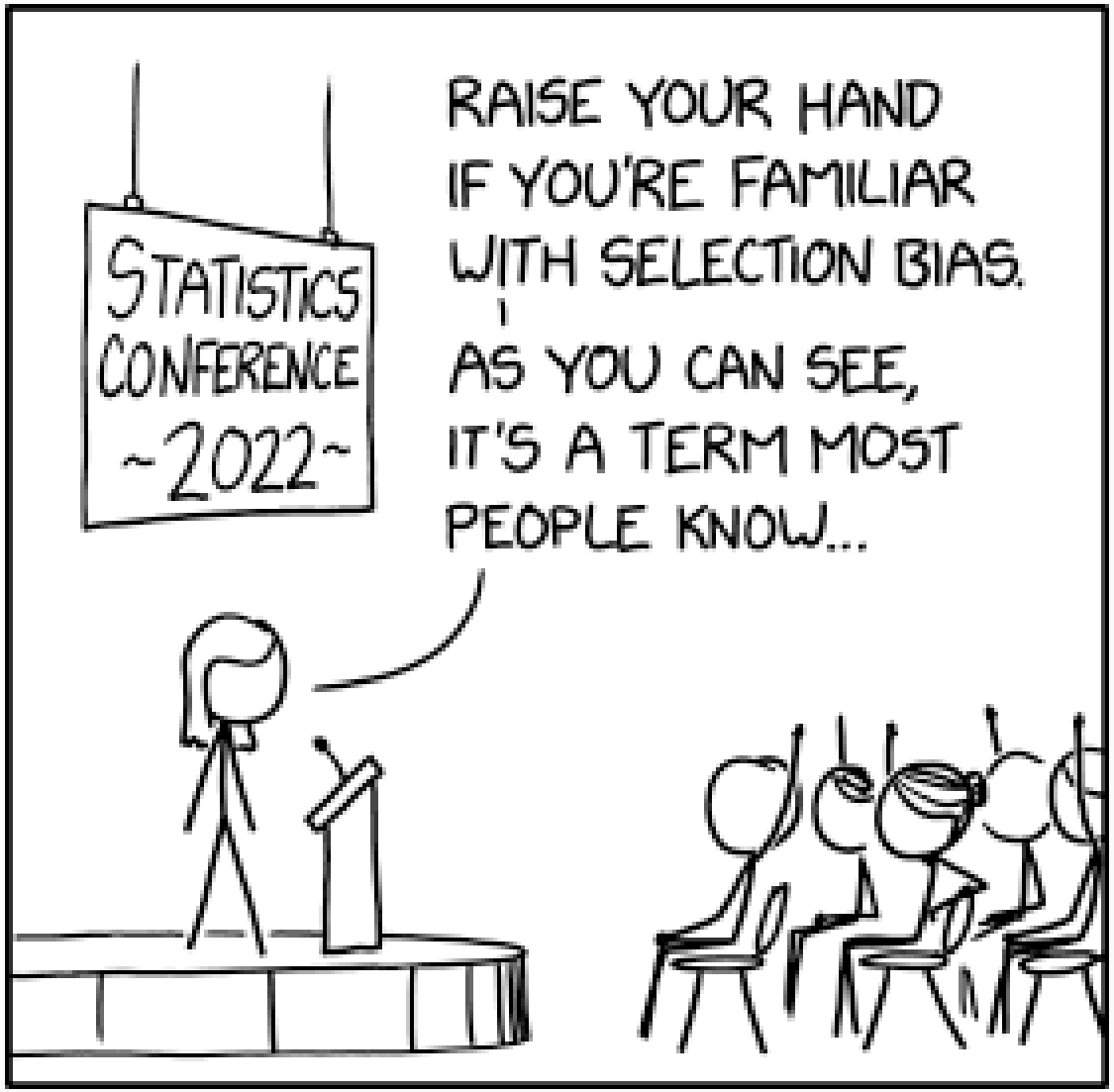


Figure 12: Poor sampling of data can lead to noise or biases

Sampling III: Stratified

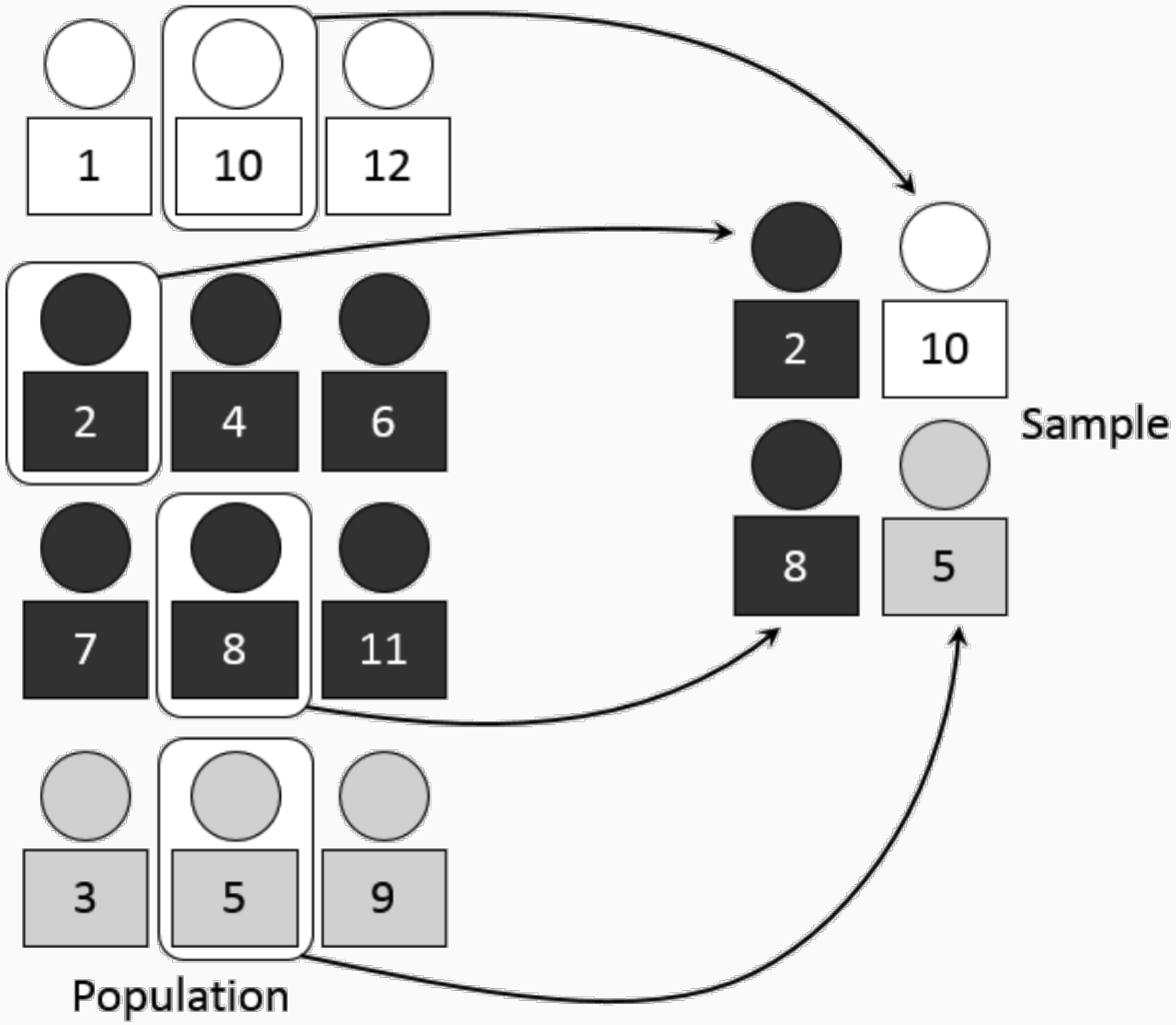


Figure 13: Stratified Sampling

Sampling III: Stratified

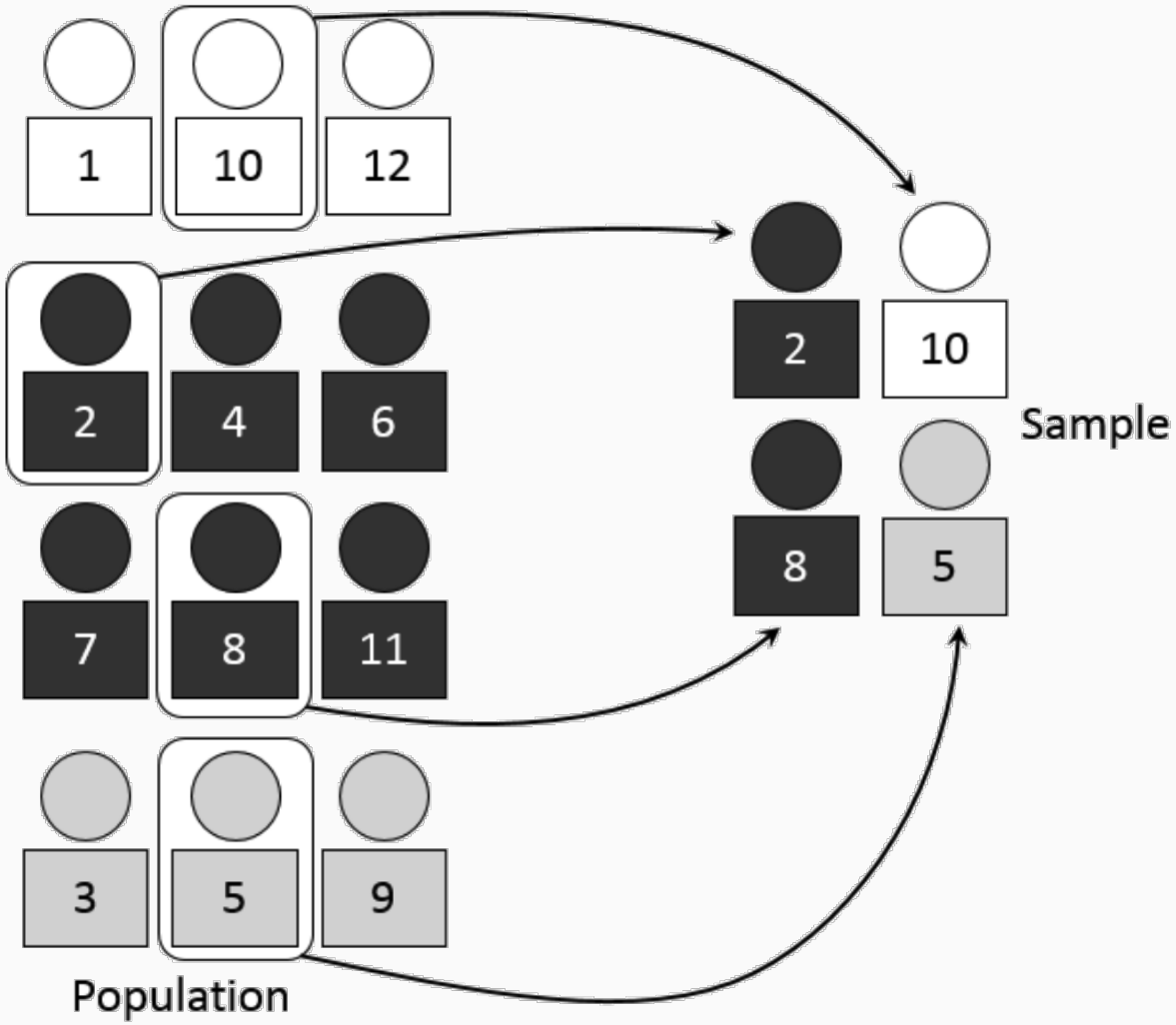


Figure 13: Stratified Sampling: **What are the advantages and disadvantages?**

Sampling IV: Systematic Random Sampling

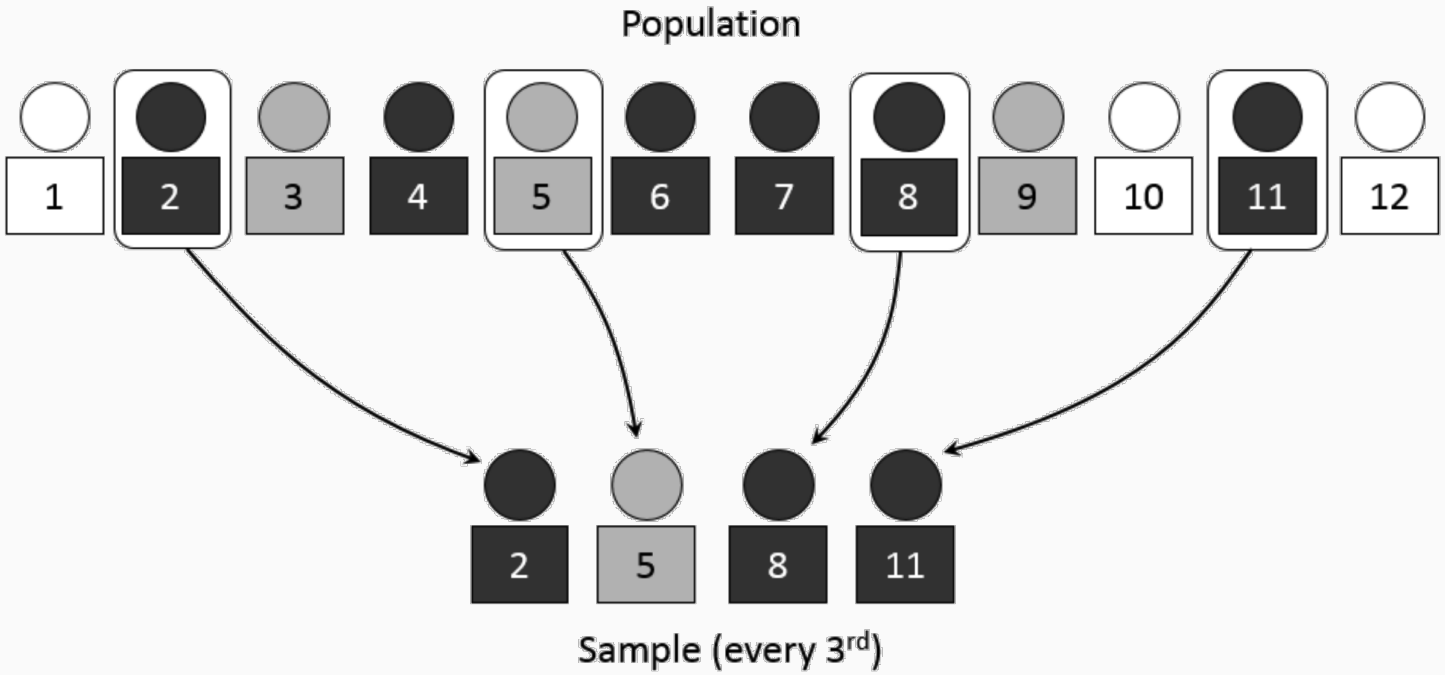


Figure 14: Selects elements from a population at regular intervals from a random starting point

Sampling V

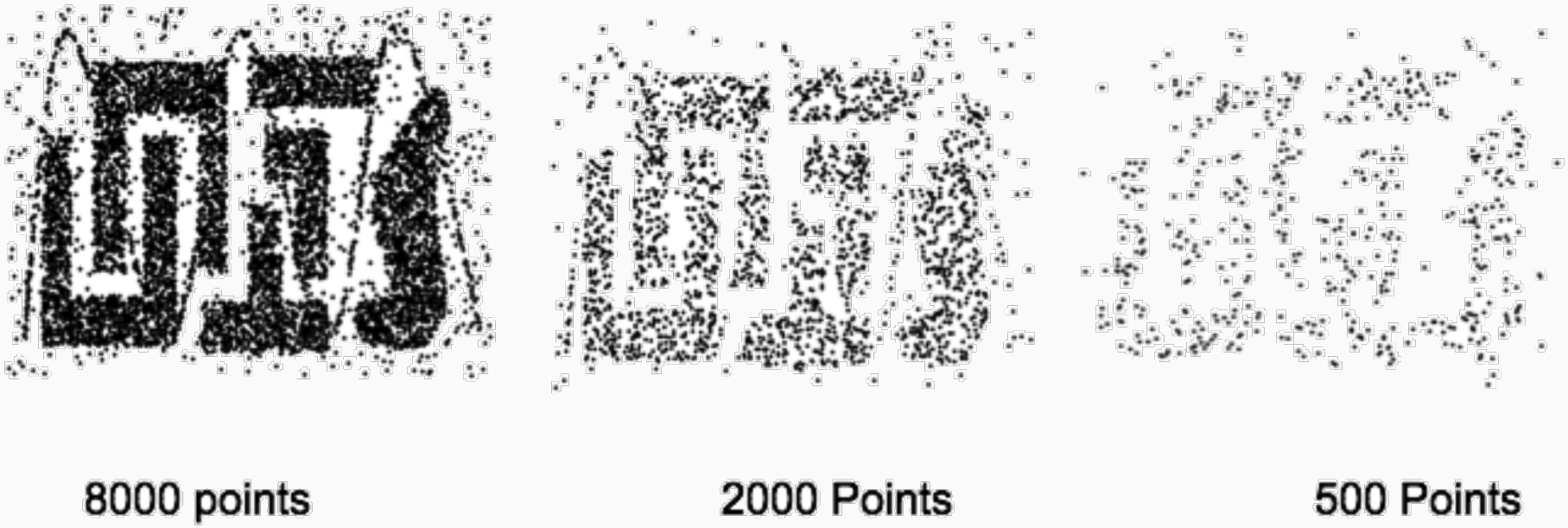


Figure 15: Which sampling method appears best for removing noise?

Outline : Aggregations

Aggregations

Combining two or more features (or objects) into a single feature (or object):

- Data reduction: fewer data points
- Scale change: using windows to temporally align data
- More stable data: less noise (e.g., averaging)

Intuition

Reduction of variability

Aggregations: Positive Emotion Analysis on Twitter

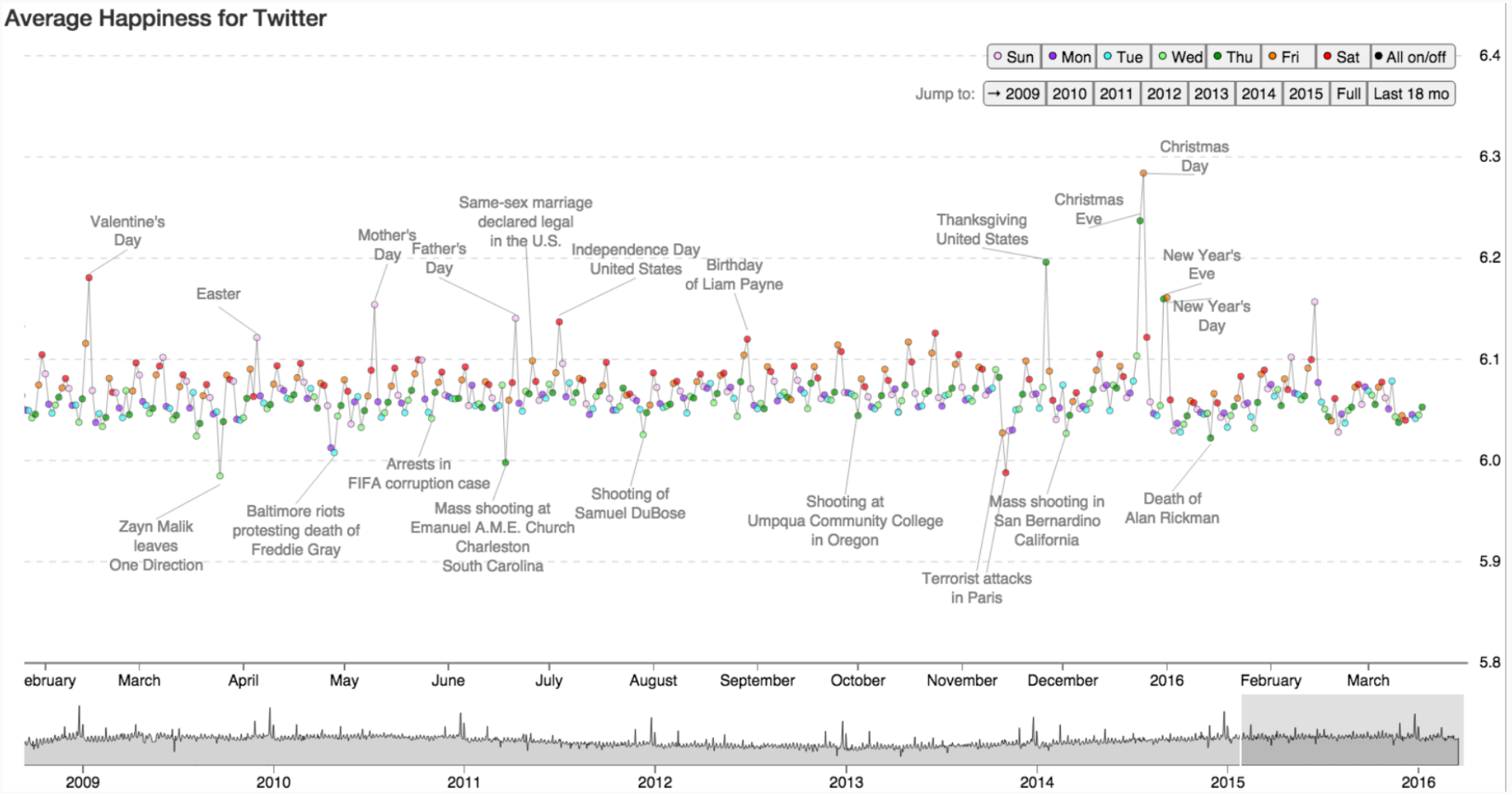


Figure 16: Aggregation over one day to measure overall sentiment

Aggregations: Positive Emotion Analysis on Twitter

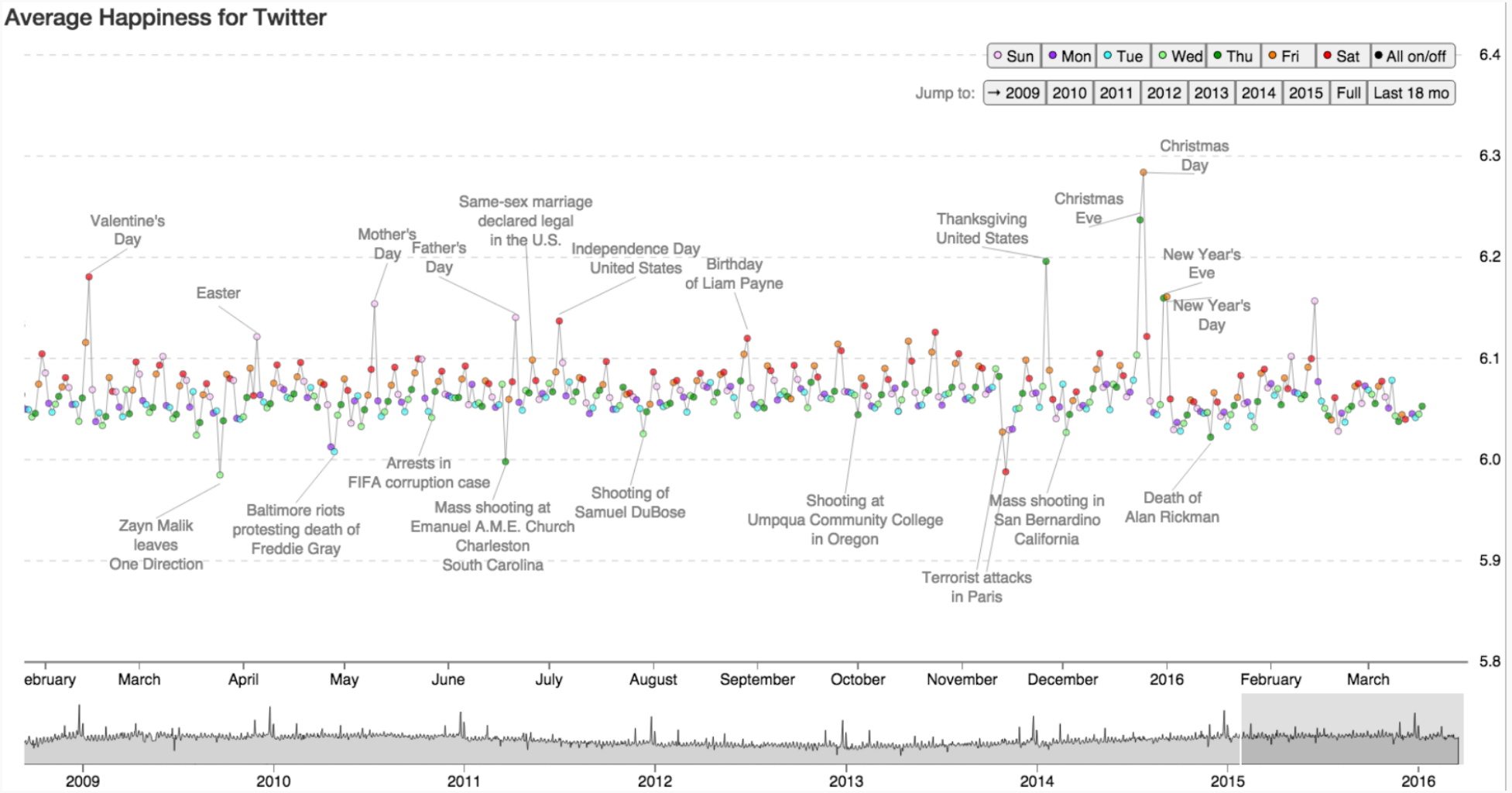


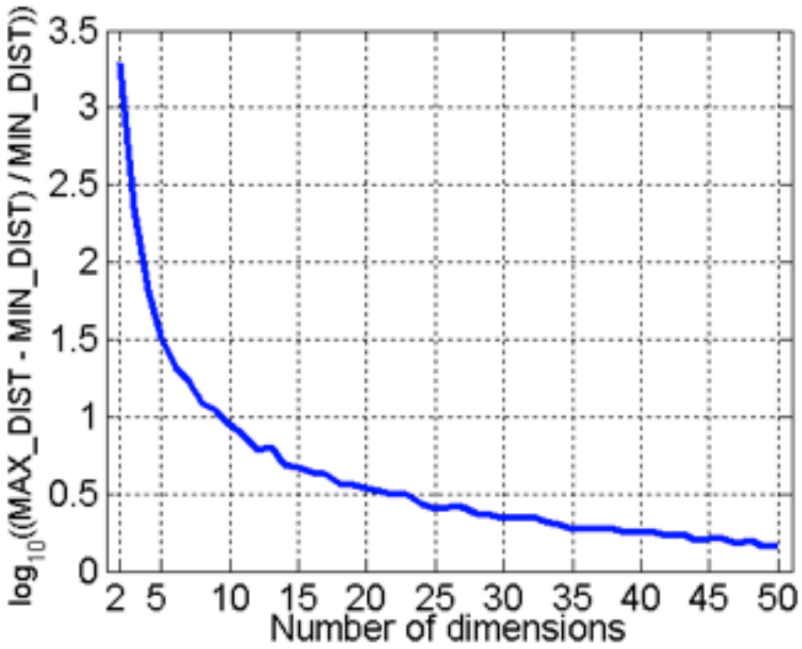
Figure 16: Aggregation over one day to measure overall sentiment

Advantages? Disadvantages?

Outline : Dimensionality: Curse and Reduction

Curse of Dimensionality

- As dimensionality increases, the data become more dispersed in the space (i.e., higher entropy)
- Measures such as density and distance between points lose meaning (affecting clustering and outlier detection)



Intuition

The higher the dimensionality, the more likely the points are to be uniformly distributed and the classes to converge. More dimensions mean more noise. We add features that do not serve the task and that can lead to spurious correlations. This increases the entropy (i.e., dispersion in the space) of the point cloud.

Dimensionality Reduction

There are techniques to reduce the dimensionality of the data:

- Feature selection: choosing the most "useful" and relevant features with minimal inter-correlation
- Principal Component Analysis: finding new (abstract) features that capture the maximum variance in the data

What Are They Used For?

(i) To avoid the curse of dimensionality, *(ii)* to reduce the costs associated with running algorithms (time, memory), *(iii)* to improve data visualization, and *(iv)* to help remove irrelevant or noisy features.

We will cover this later in the course!

Algorithms for feature selection can be found via Lasso

More about unsupervised dimensionality reduction here

Example of Feature Selection

One can use the `VarianceThreshold` function, which is a feature selector that removes all features with low variance.

```
>>> from sklearn.feature_selection import VarianceThreshold
>>> X = [[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]]
>>> selector = VarianceThreshold()
>>> selector.fit_transform(X)
array([[2, 0],
       [1, 4],
       [1, 1]])
```

Questions?

References i