Big Data and Data Mining

Week 2: Preprocess

Fenerbahce University

Instructors

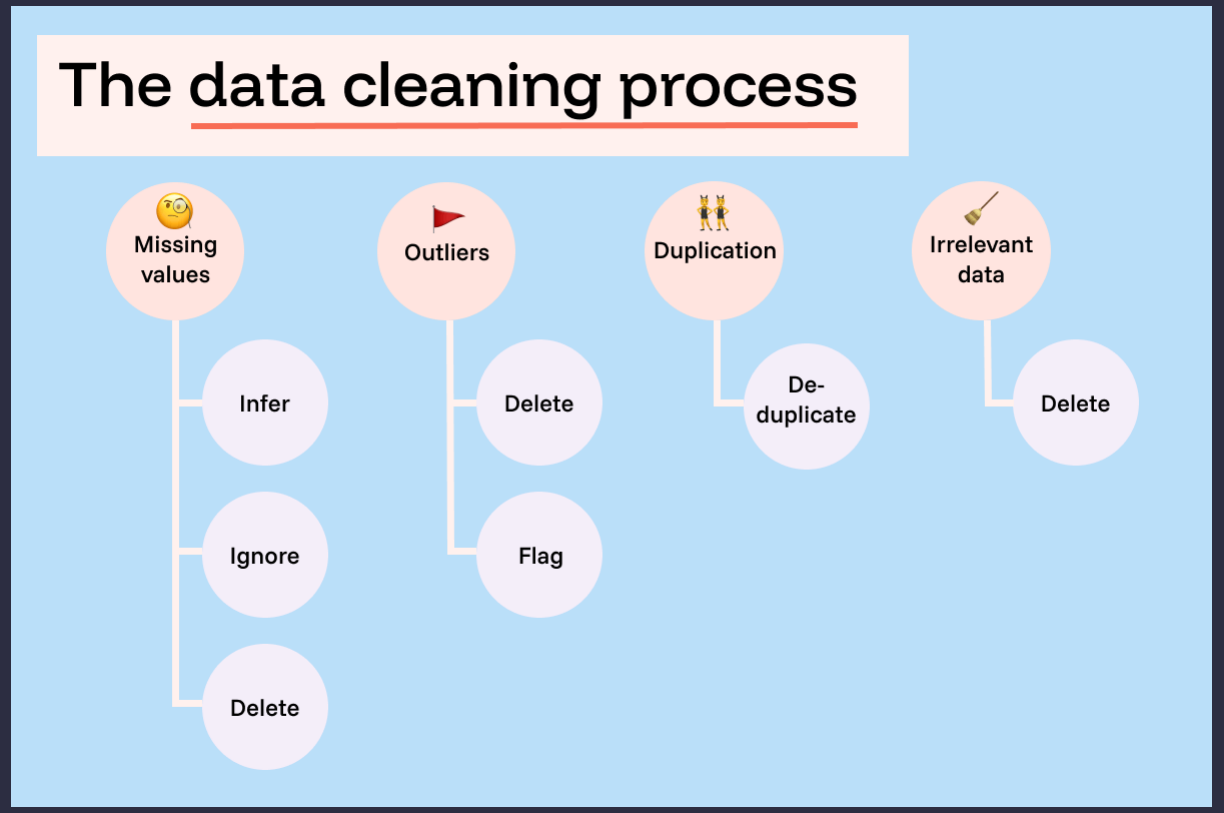
Assist. Prof. Vecdi Emre Levent

Office: 311

Email : emre.levent@fbu.edu.tr

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation



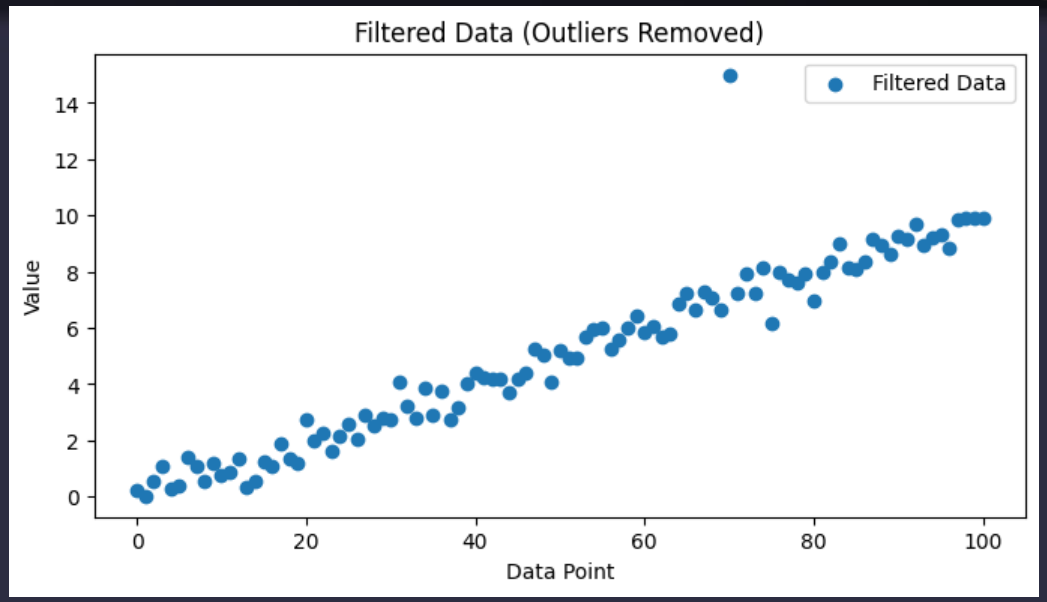
Why Data Preprocessing?

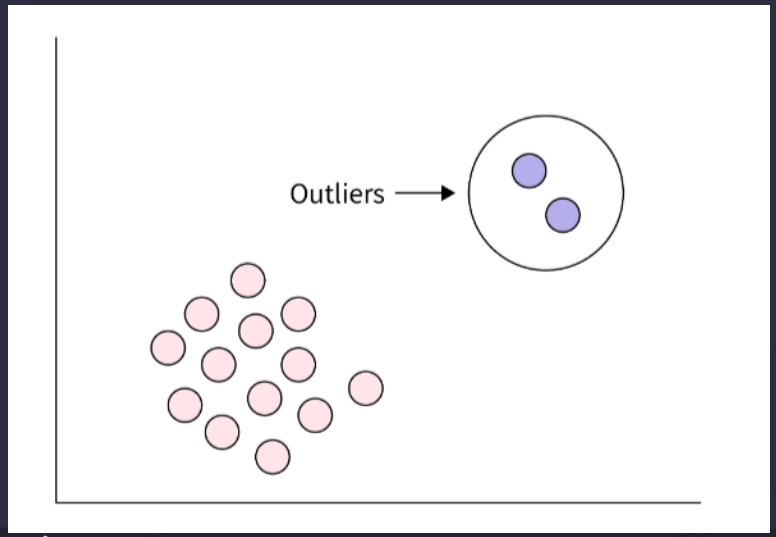
- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“”

ID	Color	Weight	Broken	Class
1	Black	80	Yes	1
2	Yellow	100	No	2
3	Yellow	120	Yes	2
4	Blue	90	No	2
5	Blue	85	No	2
6	**?**	60	No	1
7	Yellow	100	**?**	2
8	**?**	40	**?**	1

Why Data Preprocessing?

- Data in the real world is dirty
  - noisy: containing errors or outliers
    - e.g., Salary="-10"





Why Data Preprocessing?

- Data in the real world is dirty
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age="52" Birthday="03/07/1997"
    - e.g., Was rating "1,2,3", now rating "A, B, C"
    - e.g., discrepancy between duplicate records

ID	Name	Age	Salary
5218	Smith	38 (*qv*)	75,000 (*qv*)
5218	Smith	35 (*qv*)	null (*qv*)
5218	Smith	35 (*qv*)	77,000 (*qv*)
5218	Smith	38 (*qv*)	null (*qv*)
5218	Smith	40 (*qv*)	null (*qv*)
5218	Smith	45 (*qv*)	77,000 (*qv*)
5218	Smith	60 (*qv*)	null (*qv*)
5218	Smith	57 (*qv*)	50,000 (*qv*)

Why Is Data Dirty?

- Incomplete data comes from
  - n/a data value when collected
  - different consideration between the time when the data was collected and when it is analyzed.
  - human/hardware/software problems
- Noisy data comes from the process of data
  - collection
  - entry
  - transmission
- Inconsistent data comes from
  - Different data sources
  - Functional dependency violation

Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse.

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Cleaning

- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data
  - Resolve redundancy caused by data integration

Missing Data

- Data is not always available
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
- Missing data may need to be inferred.

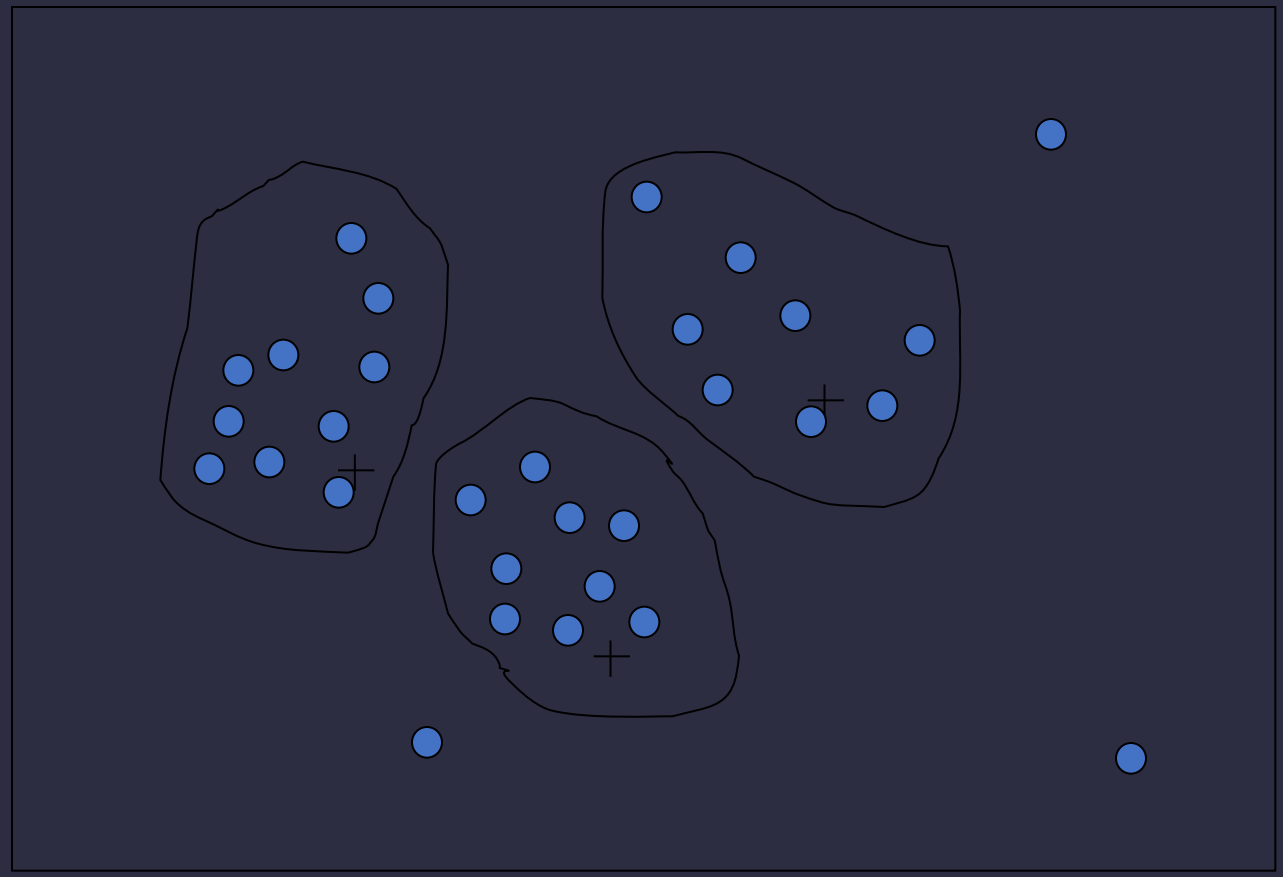
How to Handle Missing Data?

- Ignore the data: usually done when class label is missing (assuming the tasks in classification).
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter

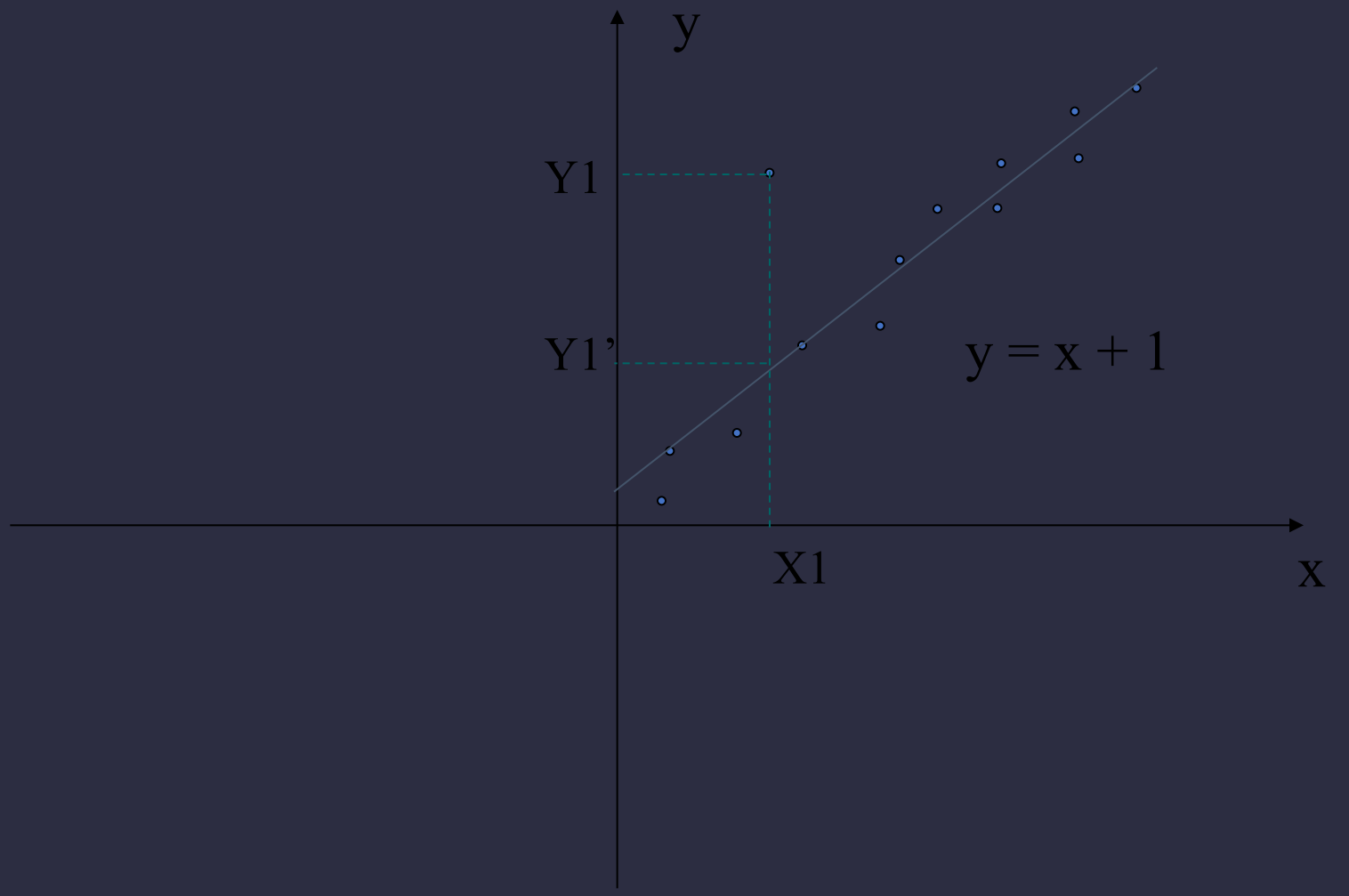
How to Handle Noisy Data?

- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)
- Regression
  - smooth by fitting the data into regression functions

Cluster Analysis



Regression

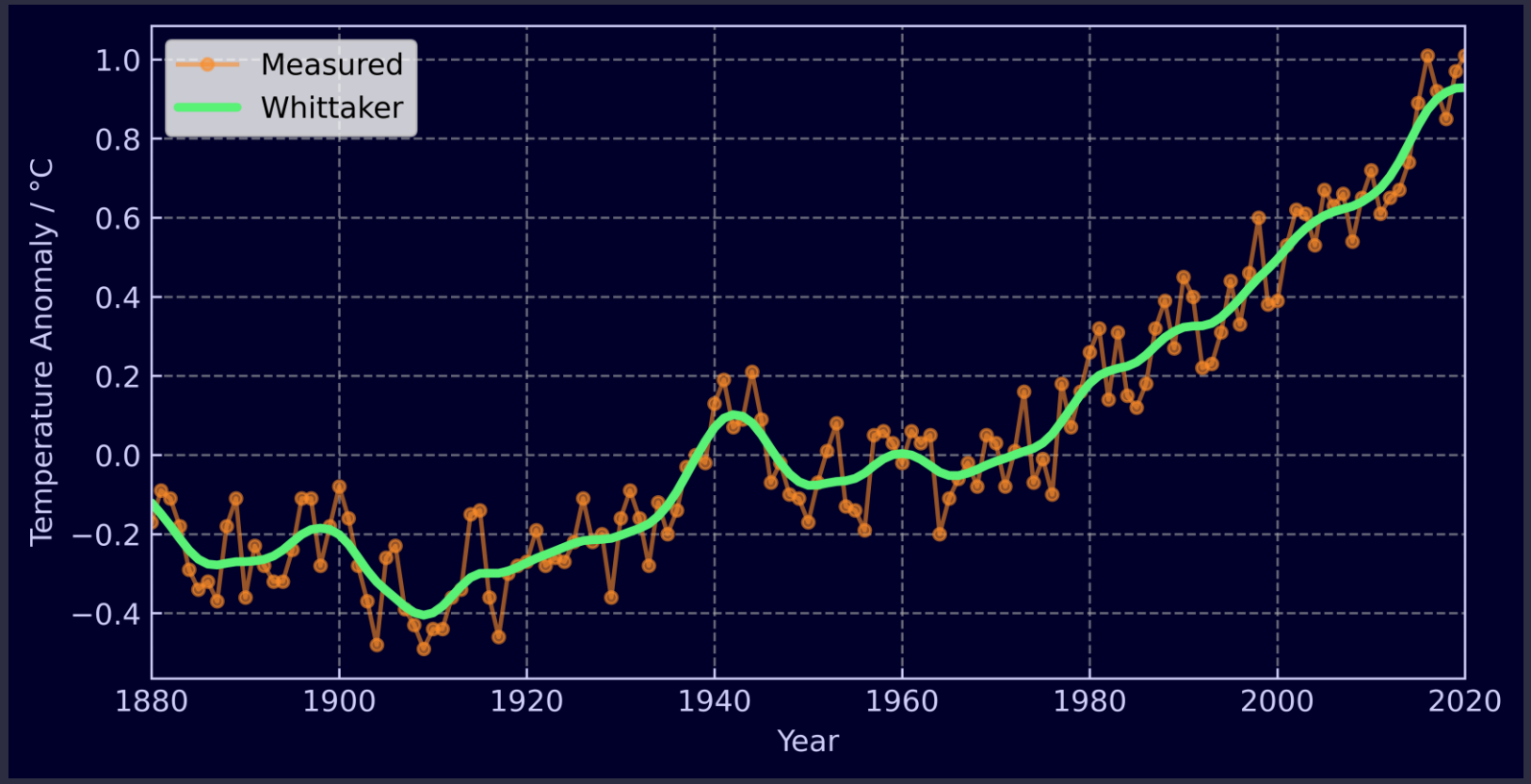


Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

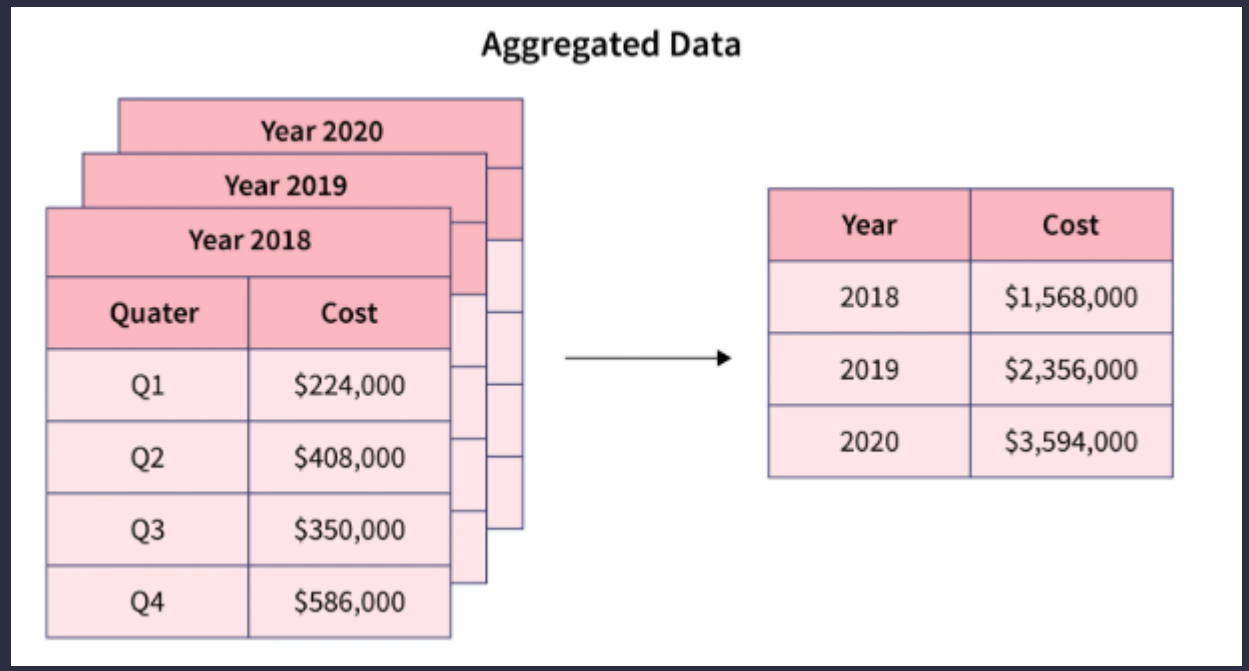
Data Transformation

- Smoothing: remove noise from data



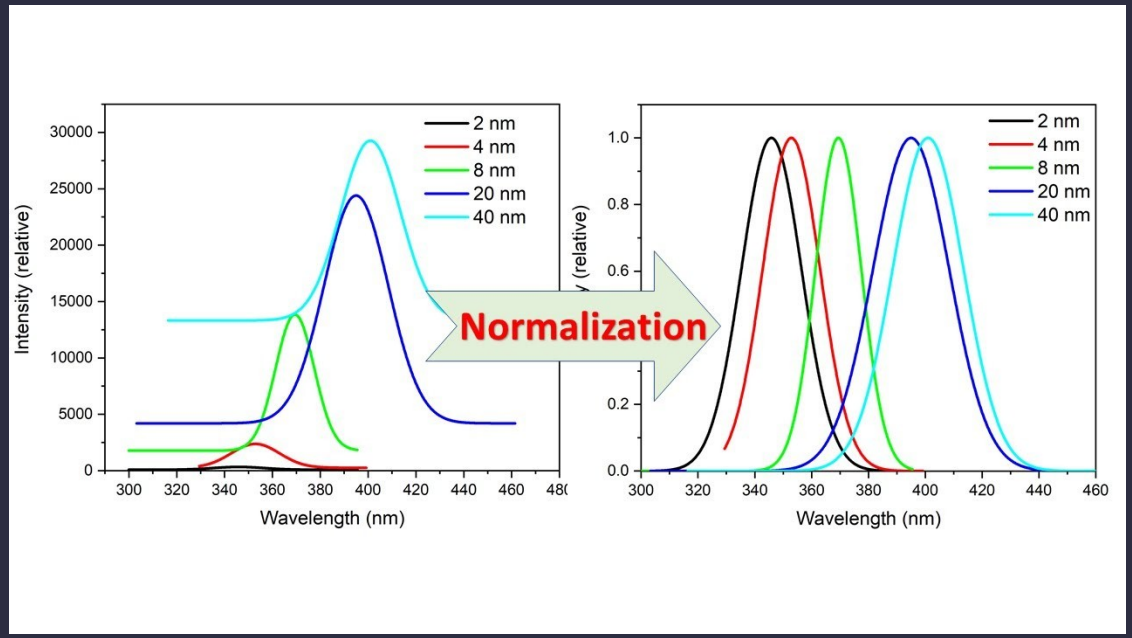
Data Transformation

- Aggregation: summarization



Data Transformation

- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling



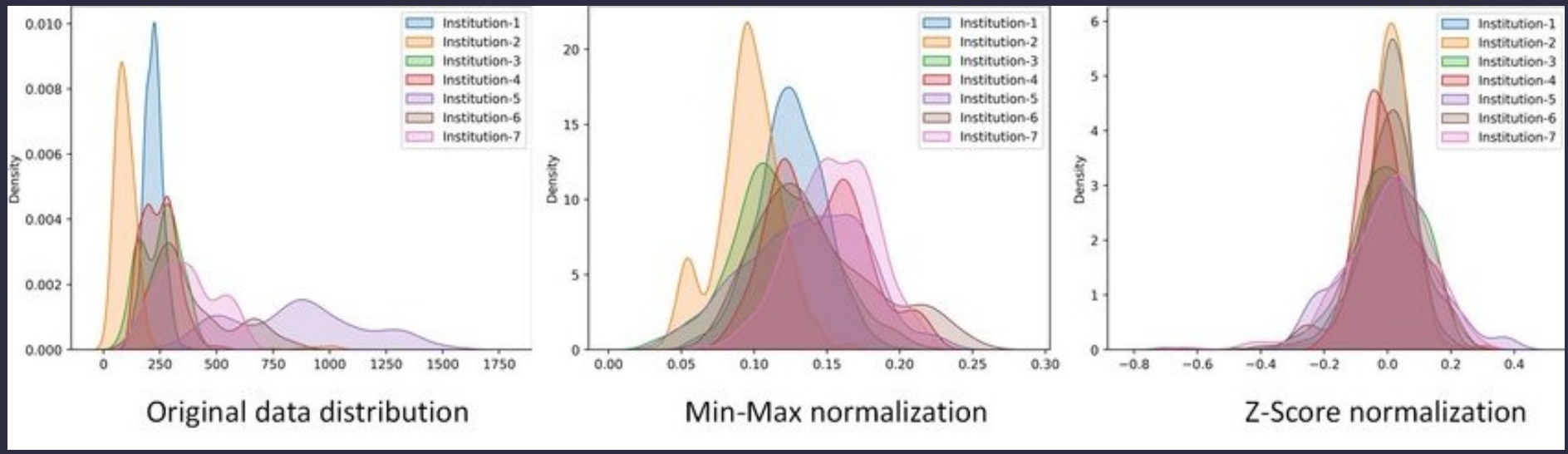
Data Transformation: Normalization

- min-max normalization

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

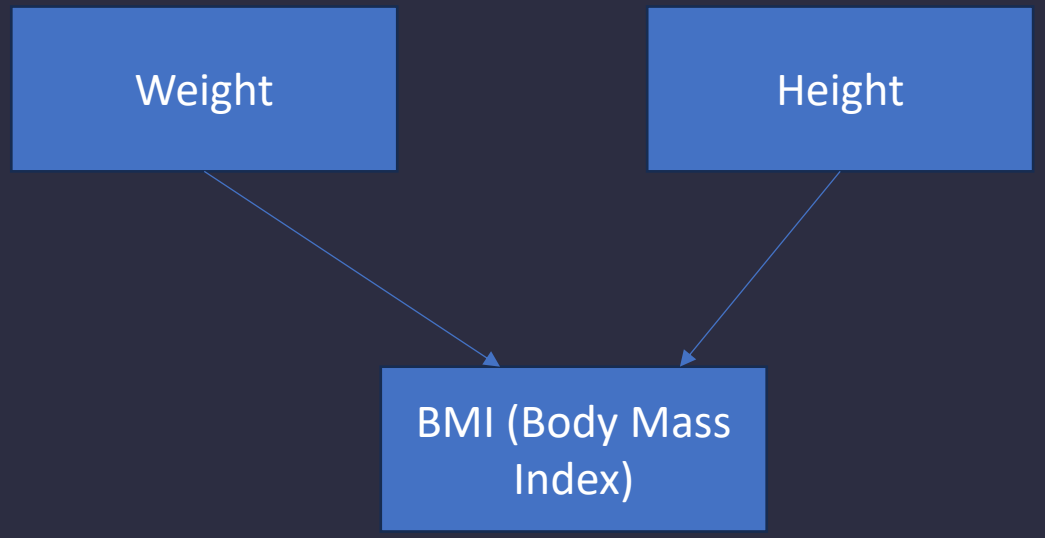
- z-score normalization

$$v' = \frac{v - mean_A}{stand\_dev_A}$$



Data Transformation

- Attribute/feature construction
  - New attributes constructed from the given ones



Data Preprocessing

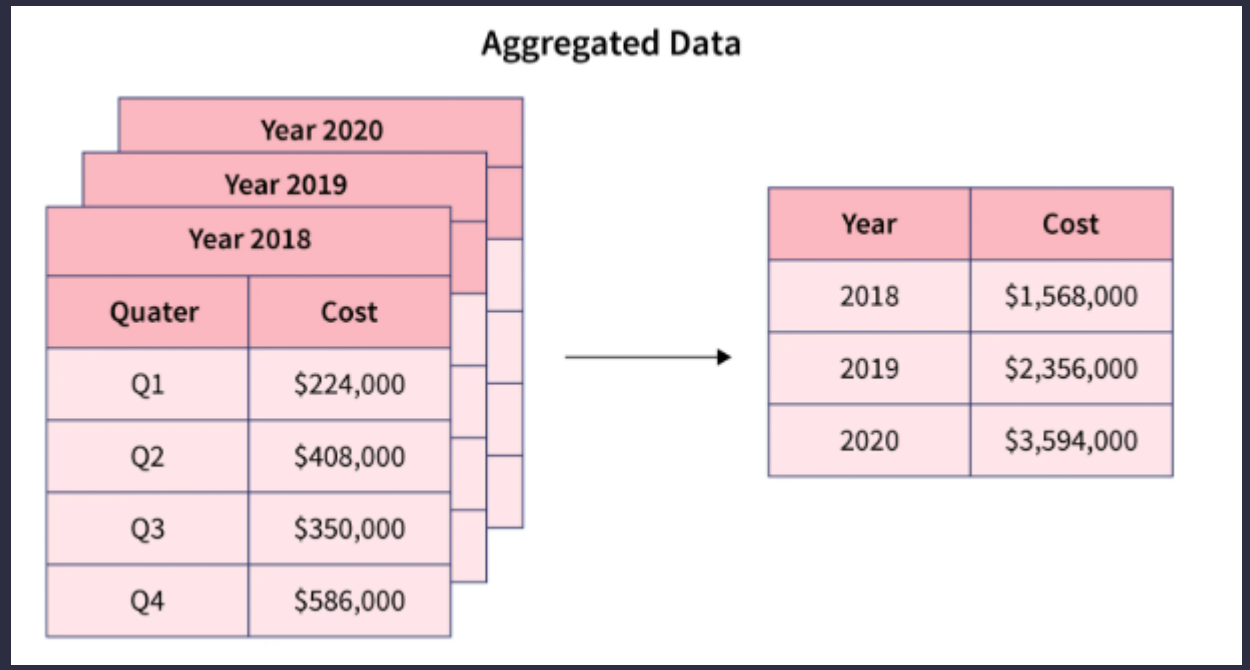
- Why preprocess the data?
- Data cleaning
- Data transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Reduction Strategies

- A data warehouse may store terabytes of data
  - Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
  - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results

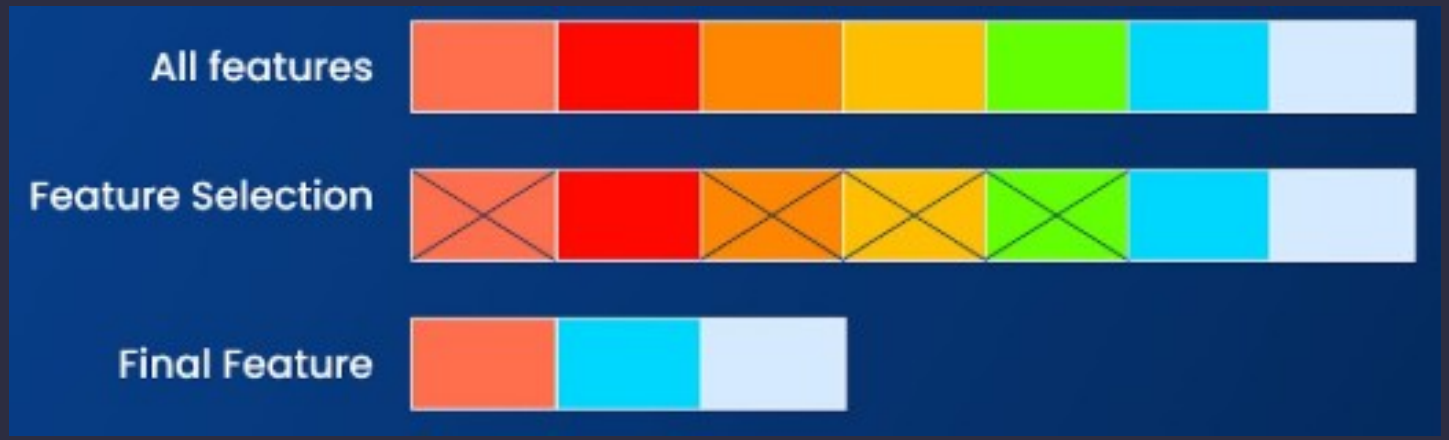
Data Cube Aggregation

- The lowest level of a data cube
  - the aggregated data for an individual entity of interest
  - e.g., a customer in a phone calling data warehouse.
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible



Dimensionality Reduction

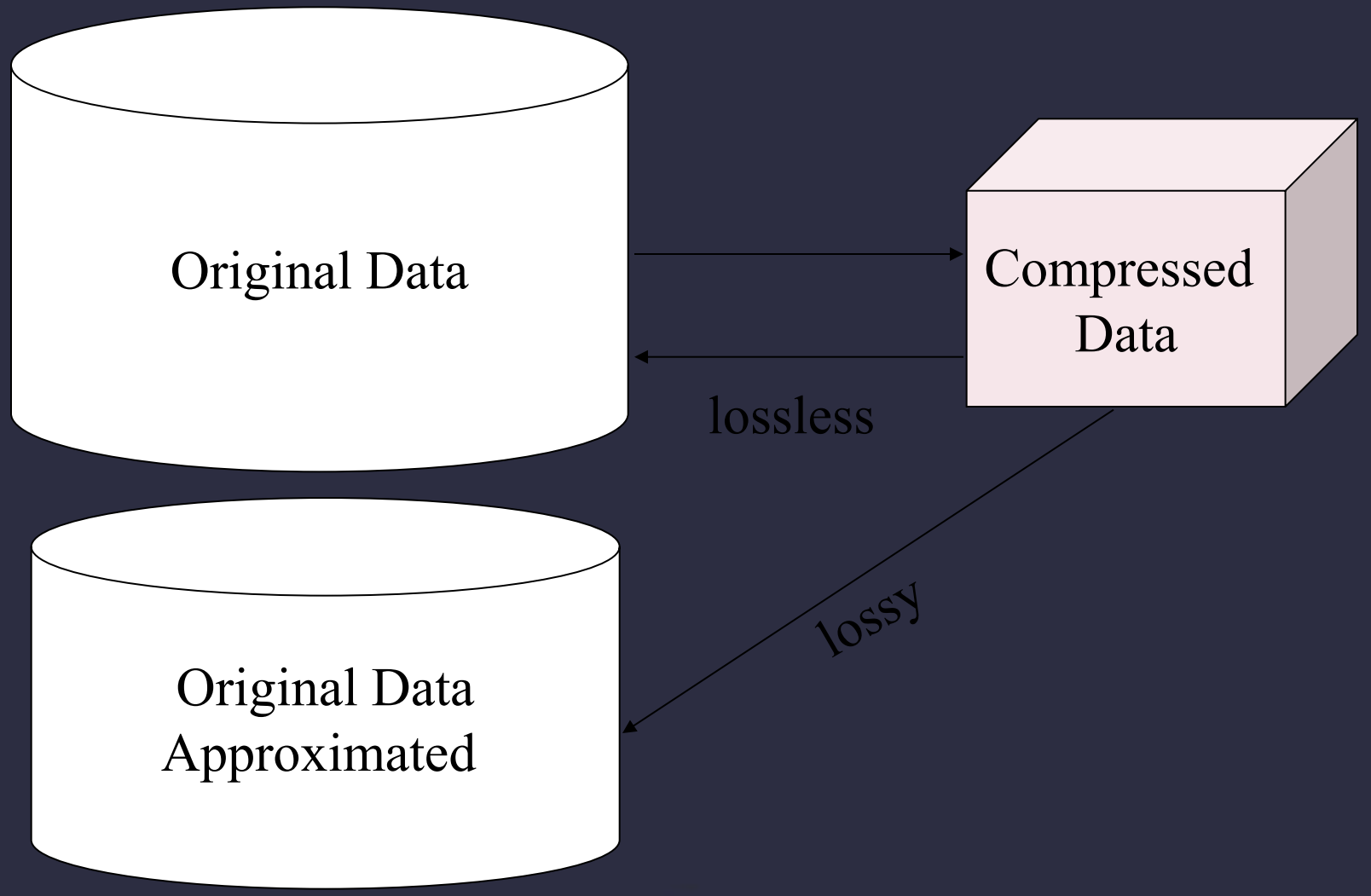
- Feature selection (i.e., attribute subset selection):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
  - reduce # of patterns in the patterns, easier to understand



Data Compression

- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless
  - But only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time

Data Compression

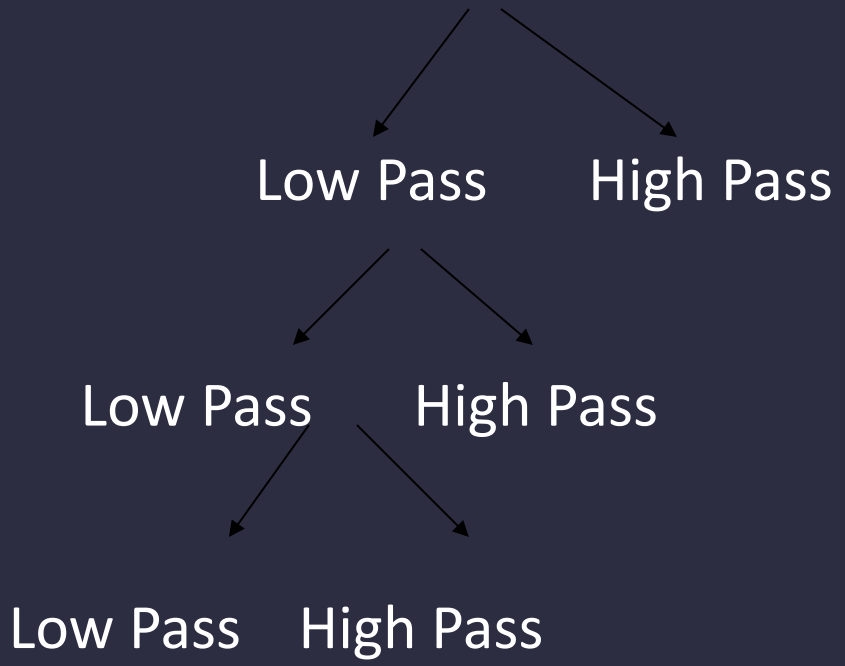


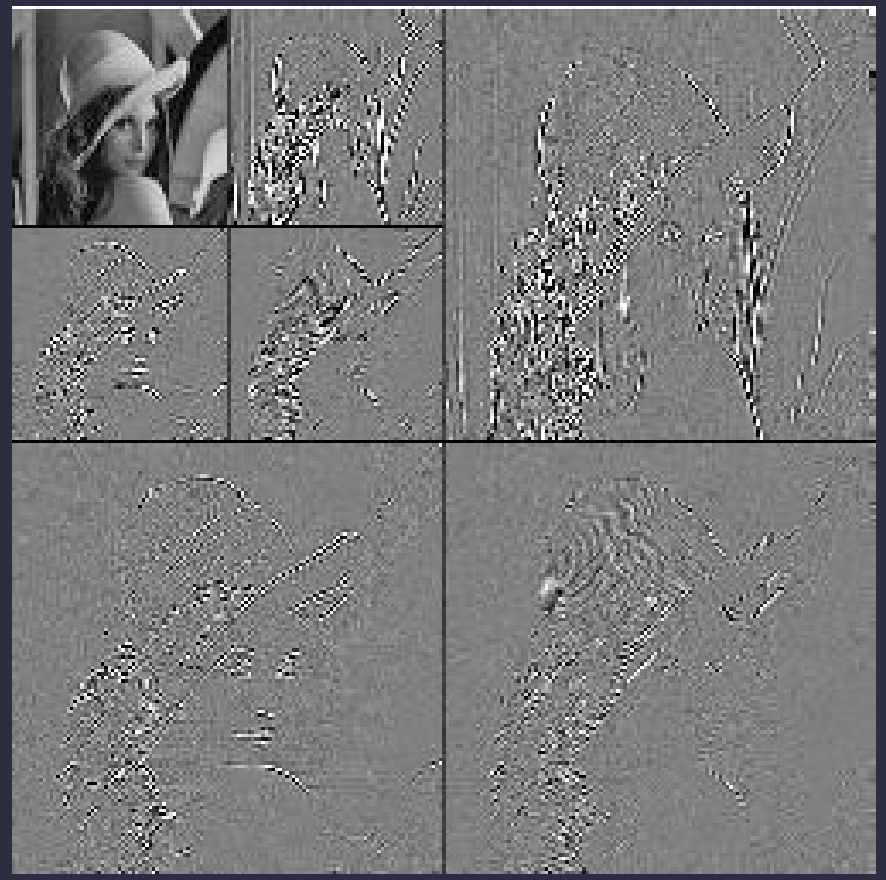
Wavelet Transformation

- Discrete wavelet transform (DWT): linear signal processing, multiresolutional analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space

DWT for Image Compression

- Image

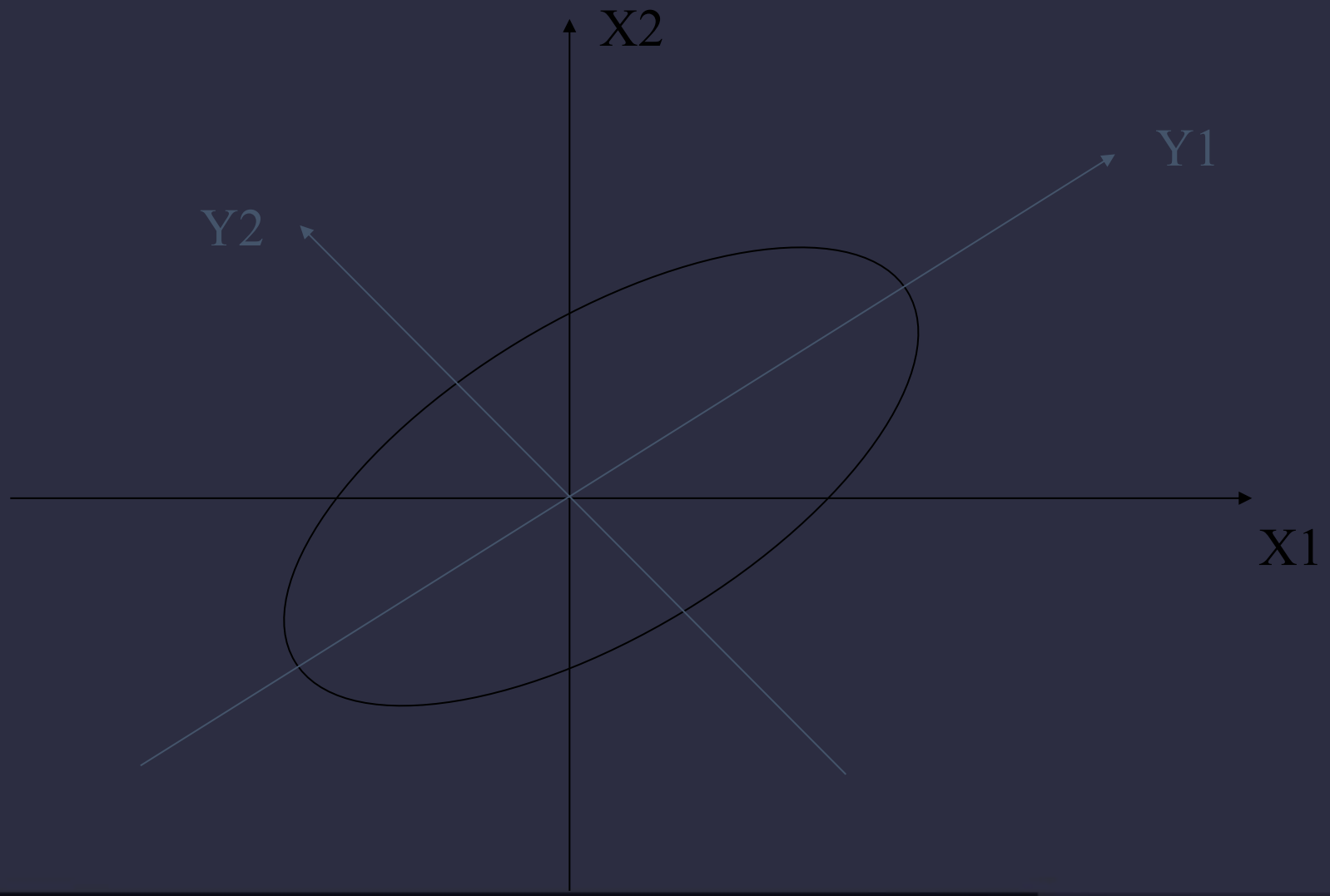


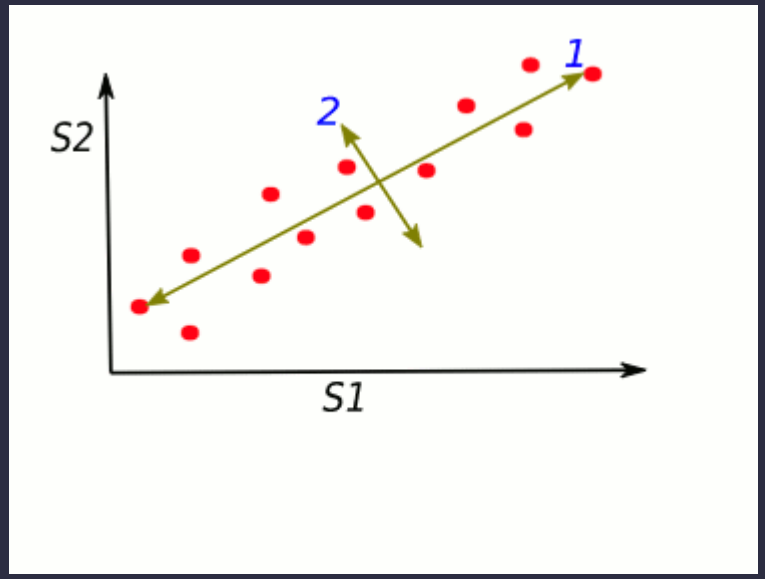


Principal Component Analysis

- Given N data vectors from k-dimensions, find c ≤ k orthogonal vectors that can be best used to represent data
  - The original data set is reduced to one consisting of N data vectors on c principal components (reduced dimensions)
- Each data vector is a linear combination of the c principal component vectors
- Works for numeric data only
- Used when the number of dimensions is large

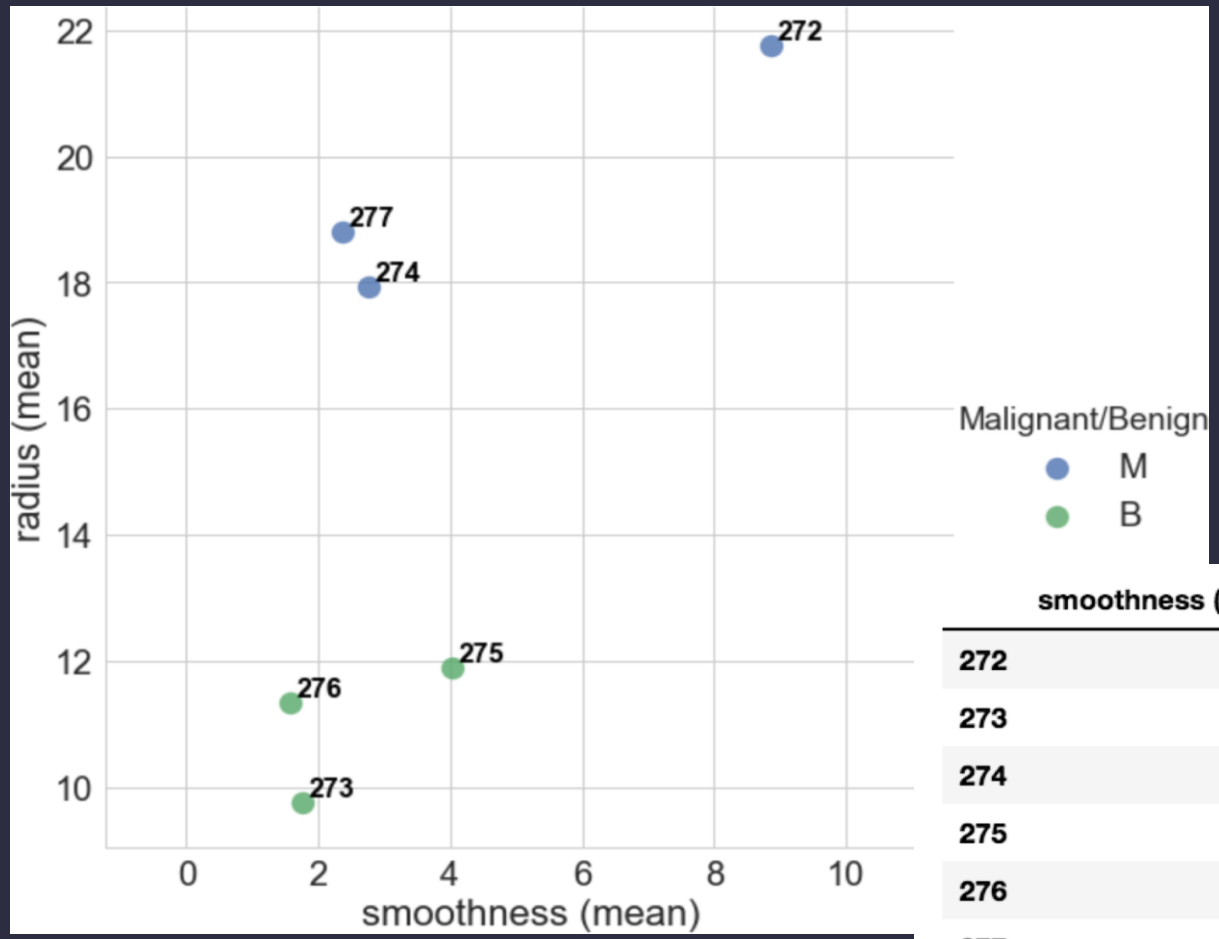
Principal Component Analysis





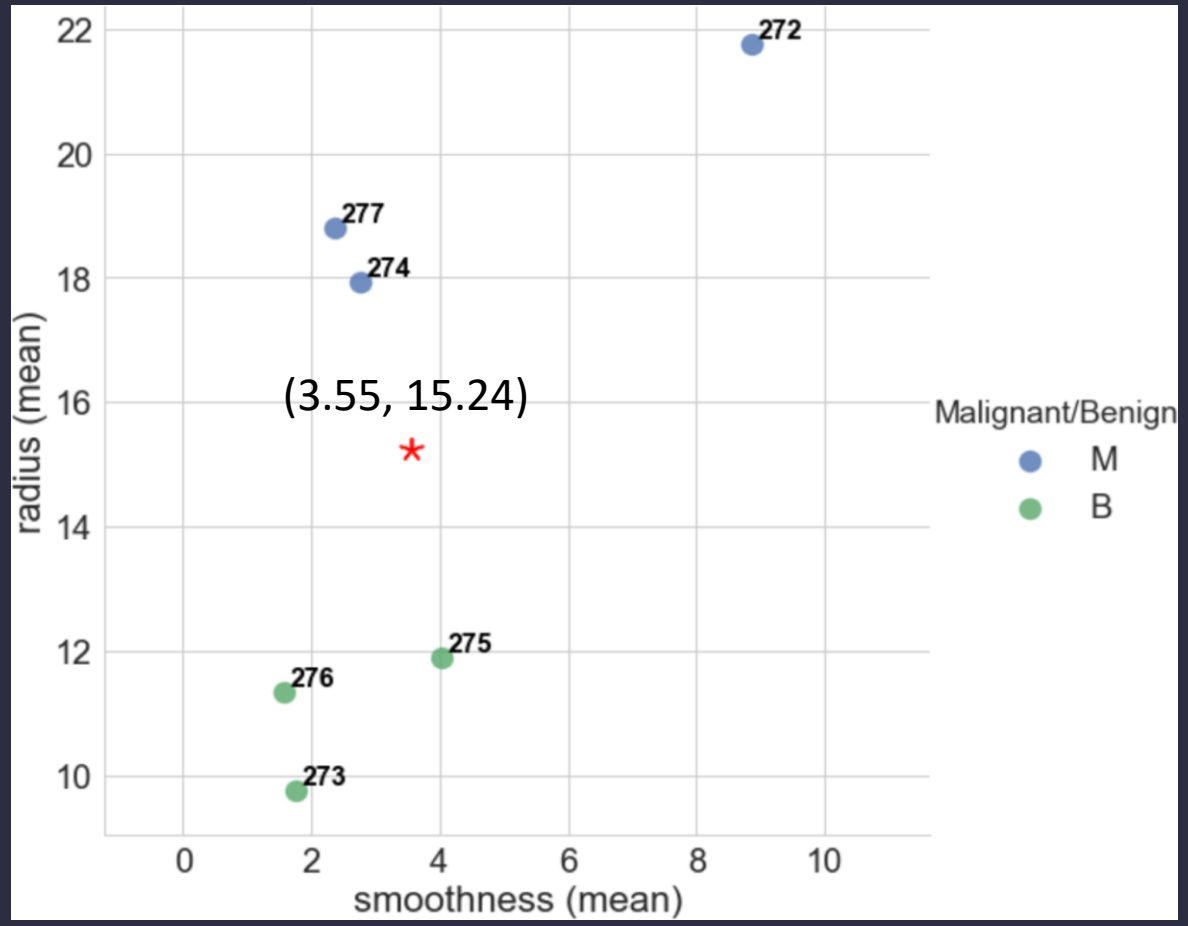
PCA Example

Let's start with a subset of 6 patients, and take a look at only two of the features: smoothness and radius

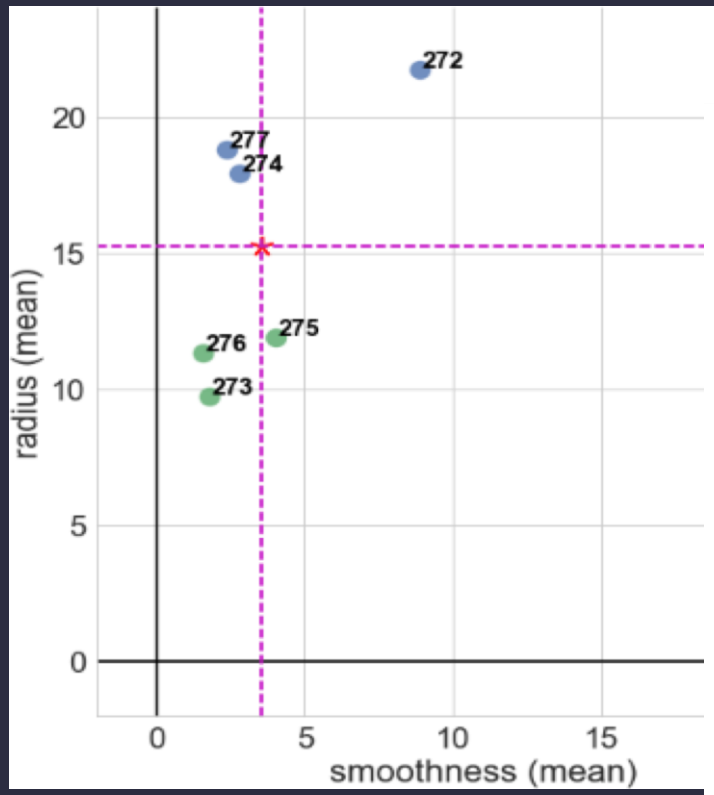


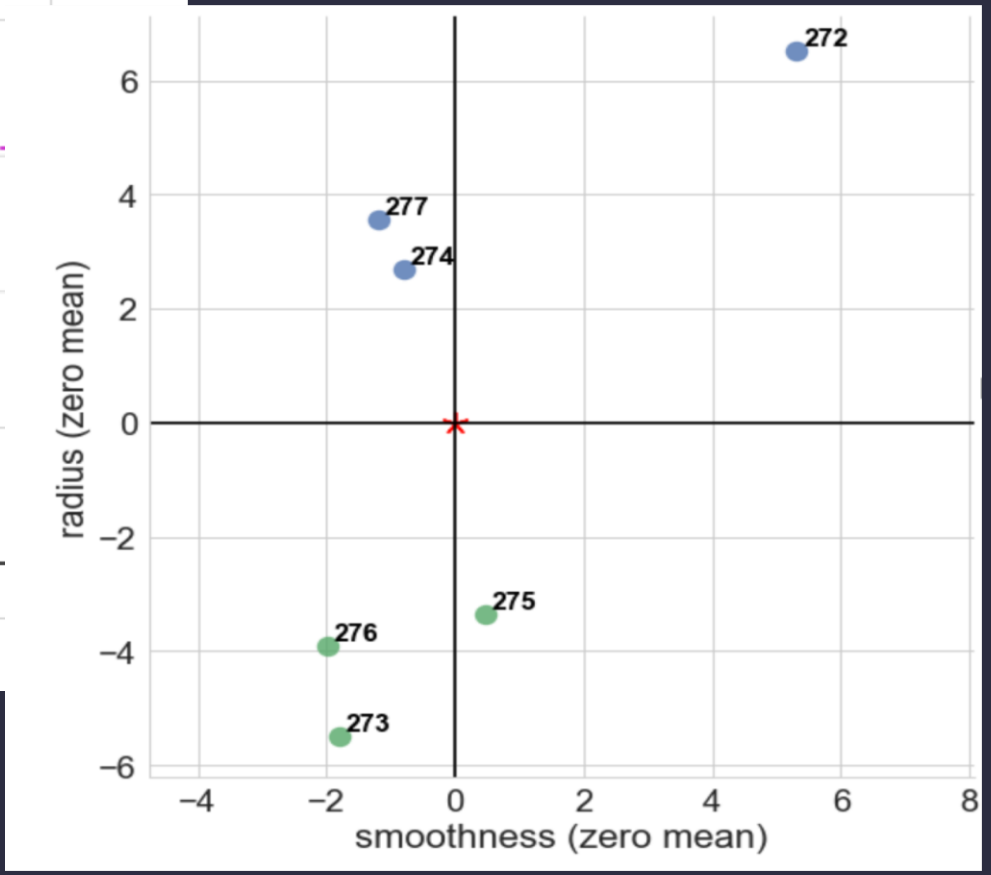
	smoothness (mean)	radius (mean)
272	8.867	21.750
273	1.750	9.742
274	2.765	17.930
275	4.021	11.890
276	1.565	11.330
277	2.363	18.810

Determine the “center” of the dataset – the mean value of each feature

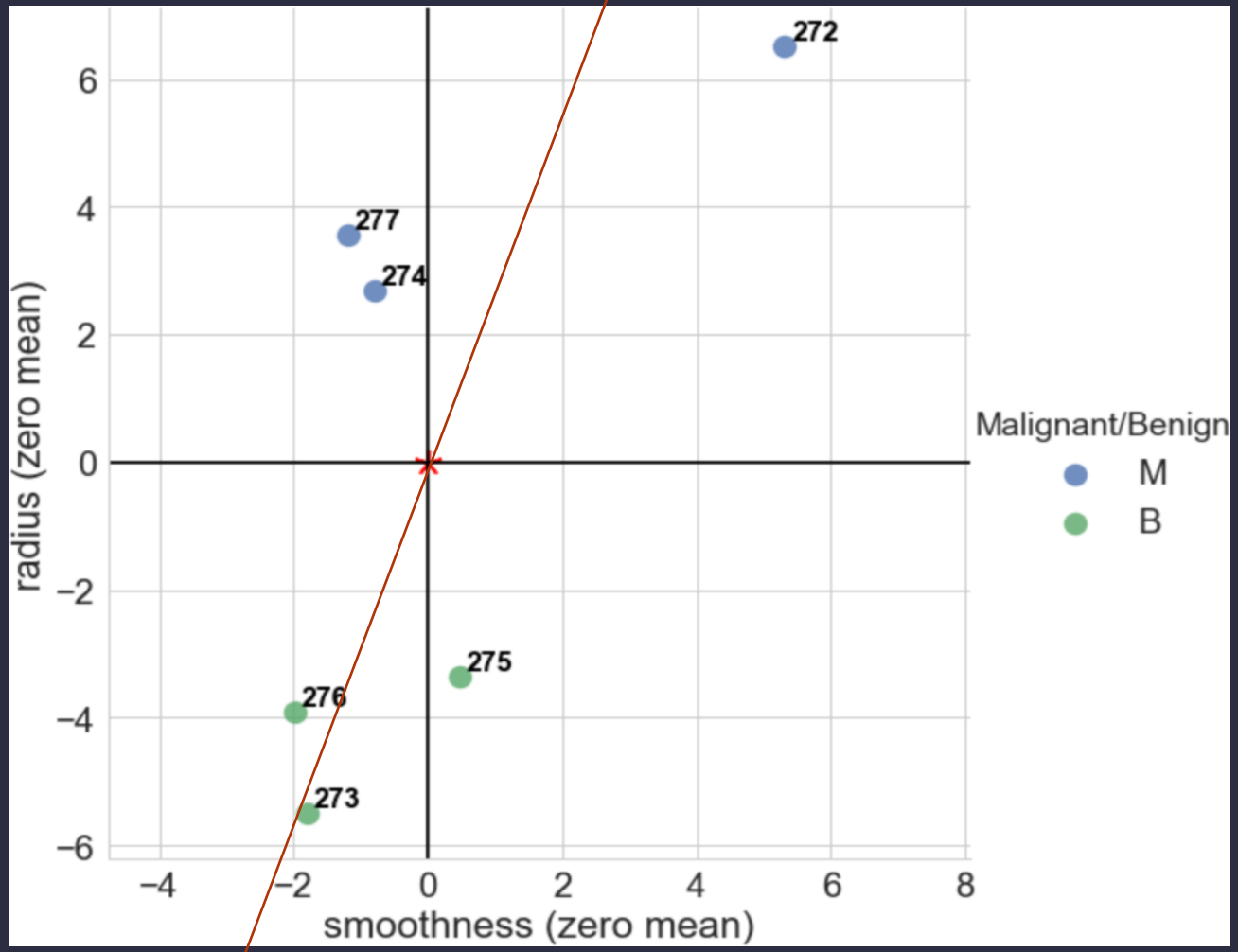


We will shift the dataset such that the “center” of the dataset (mean value) is at the origin (0,0) – the new dataset has zero mean value.

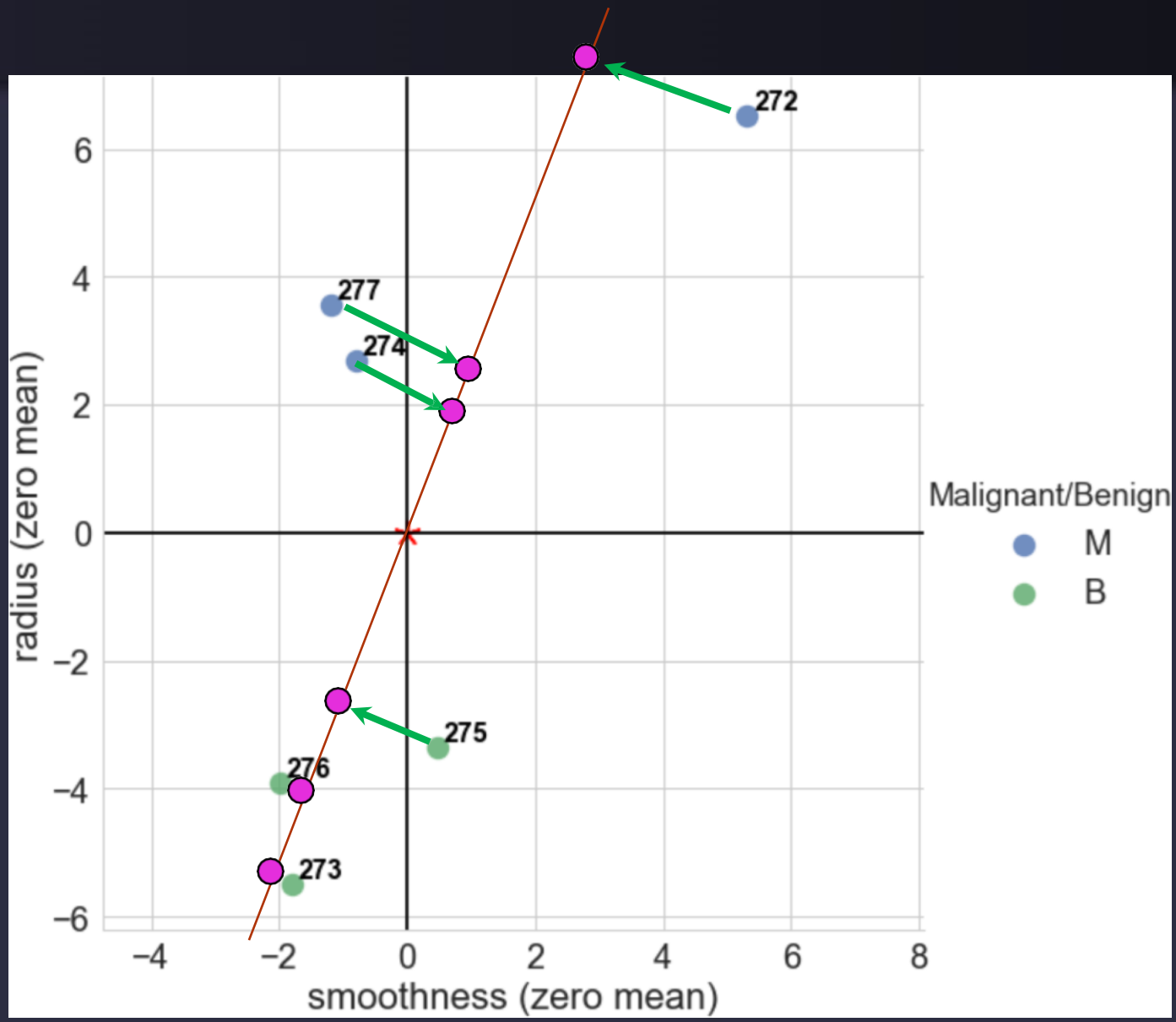




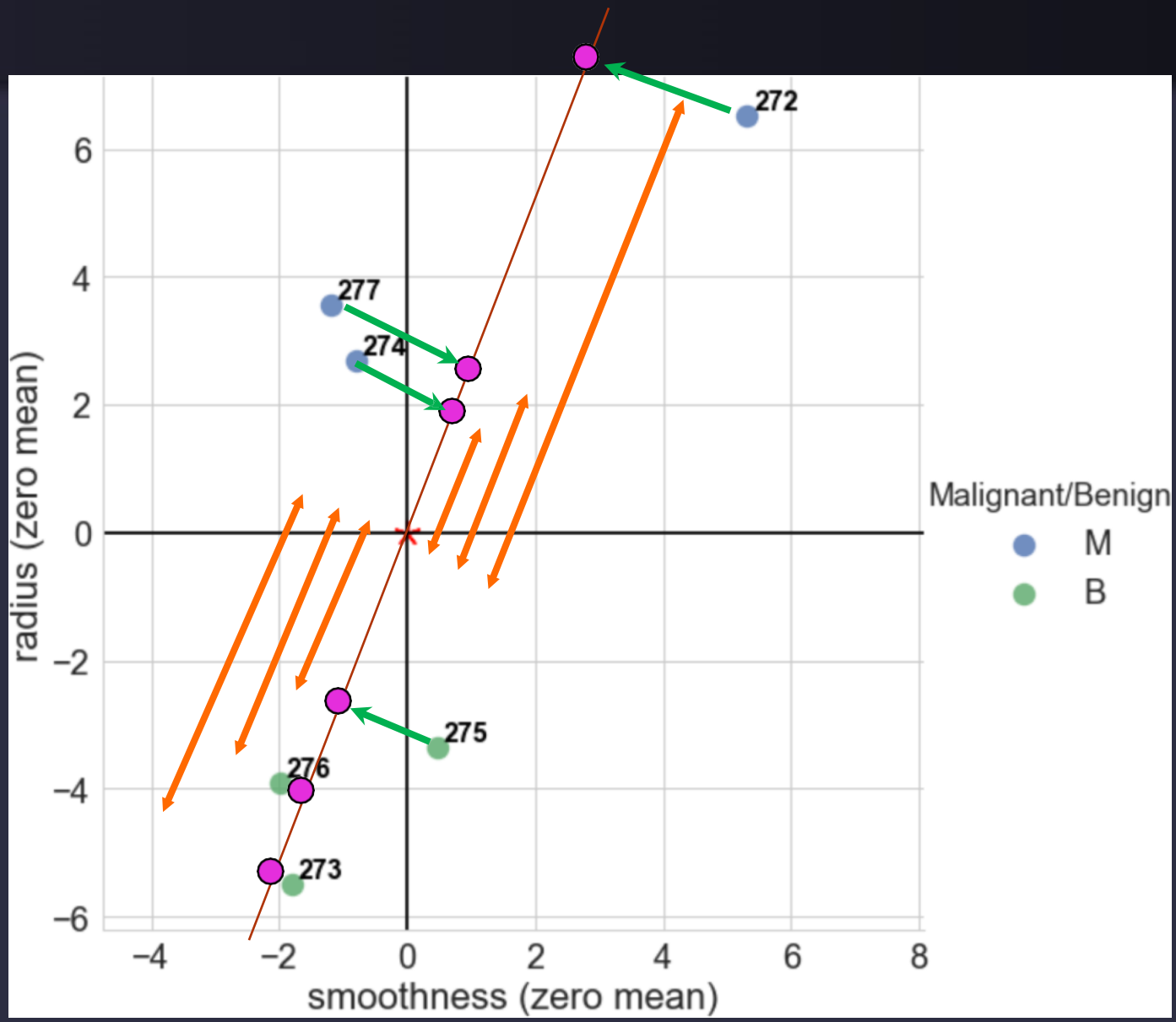
We want to find a straight line that fits the dataset.



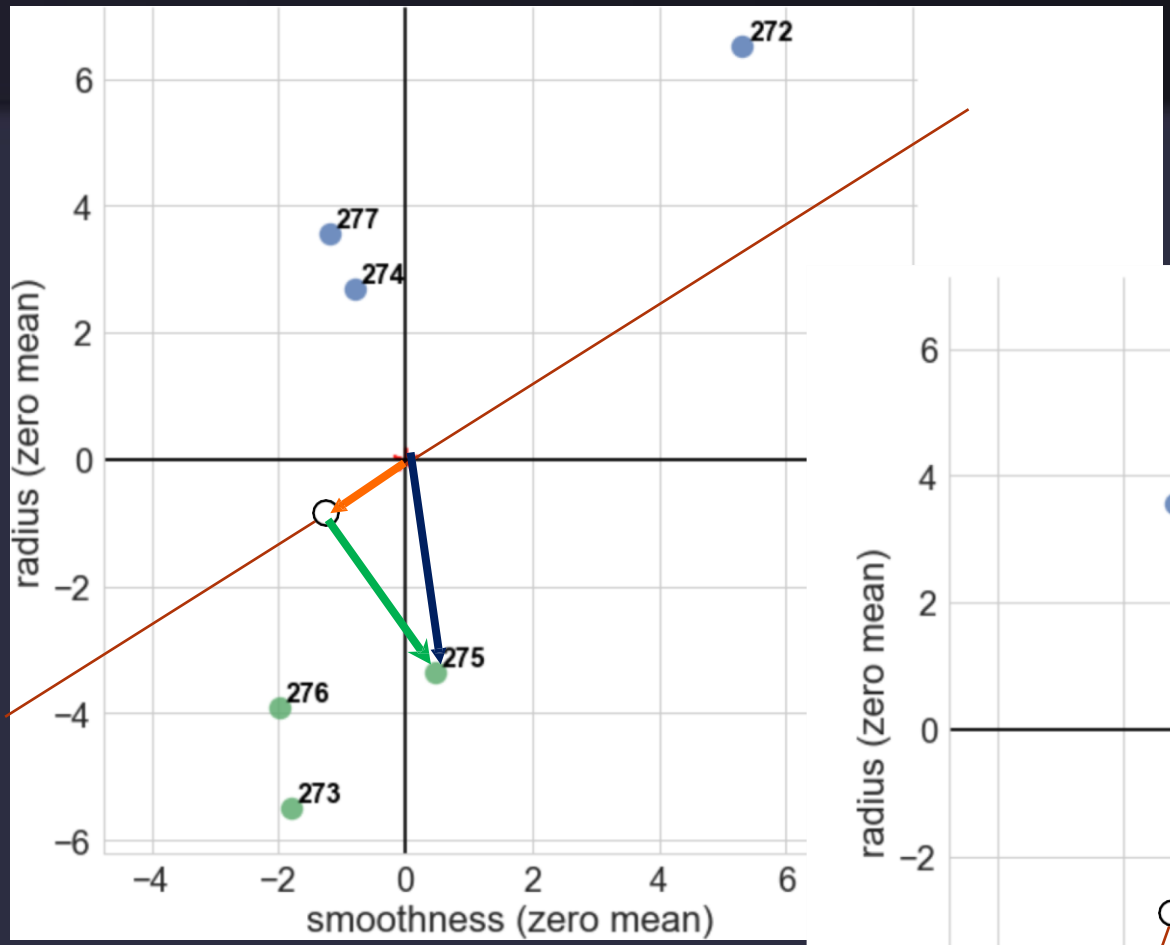
Let’s propose the red line below. To quantify how good the fit is, PCA projects the data onto the line. The best fit minimizes the distances from the points to the line (indicated in green below)…

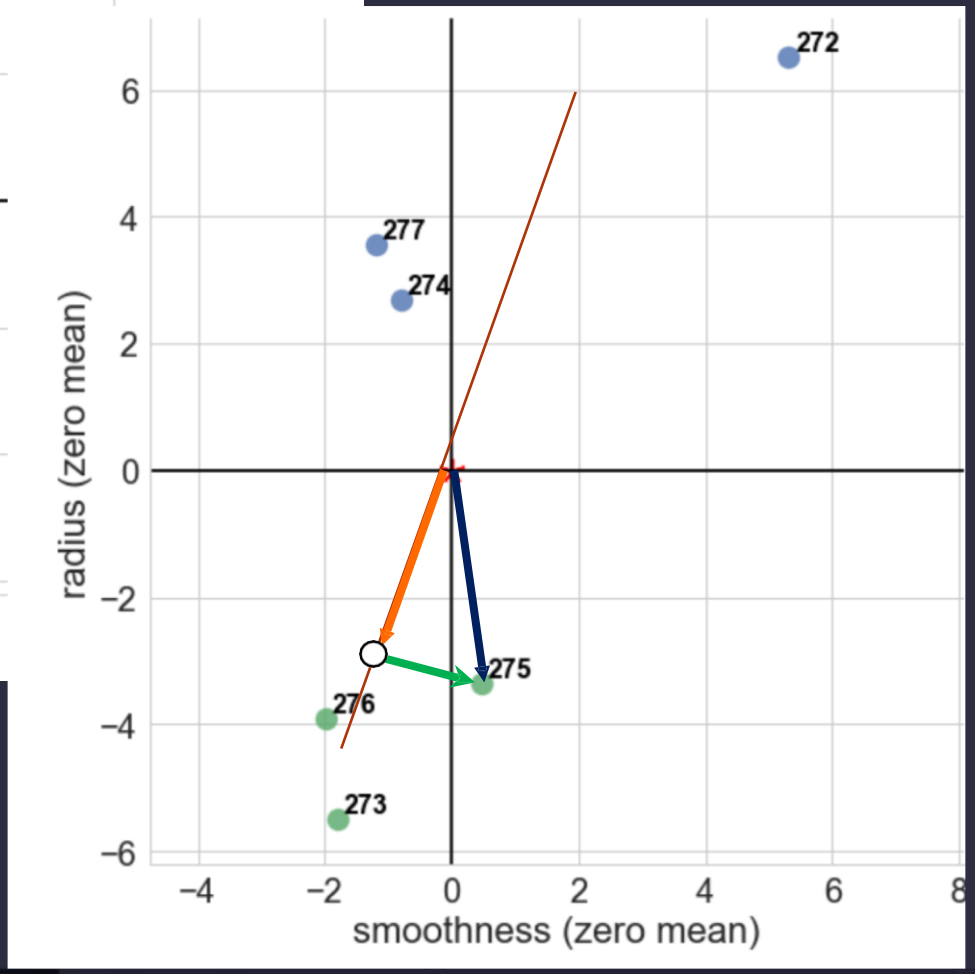


Or maximizes the distances from the projected points to the origin (indicated in orange)

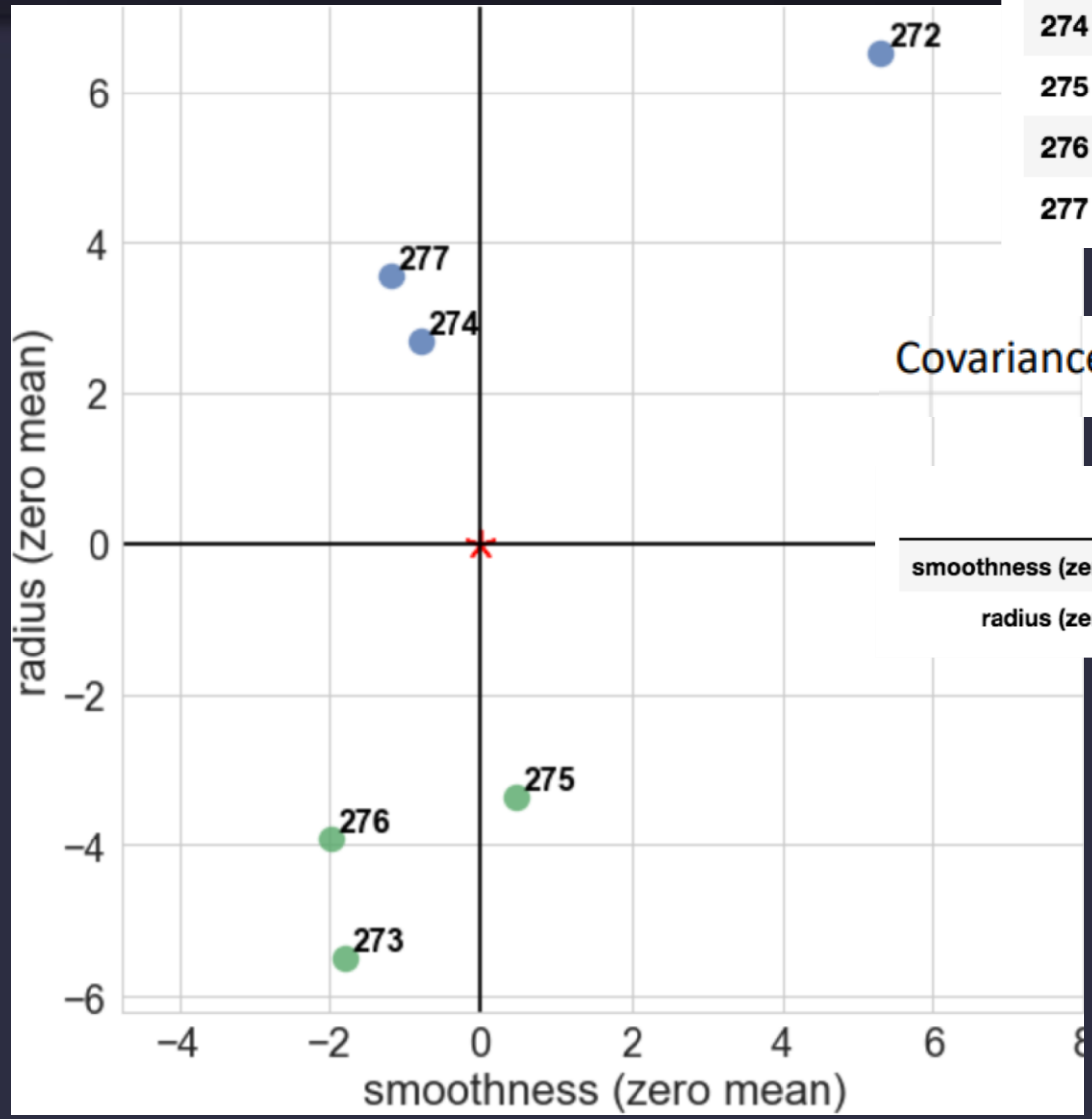


Why are they the same?
Take a look at what happens to the vectors below when we change the fit curve.





Let’s talk about the variance of the dataset



	smoothness (zero mean)	radius (zero mean)
272	5.311833	6.508
273	-1.805167	-5.500
274	-0.790167	2.688
275	0.465833	-3.352
276	-1.990167	-3.912
277	-1.192167	3.568

$!=$

Covariance matrix: $\frac{1}{(n-1)} A^T A$

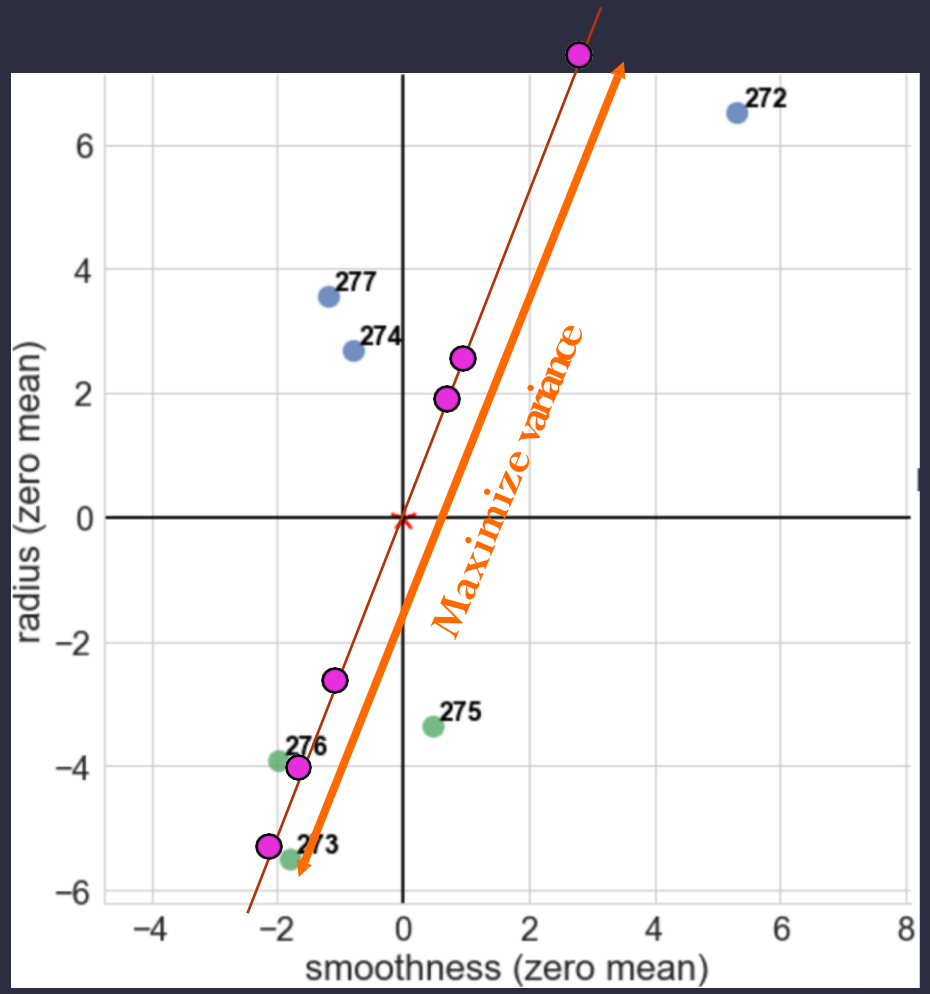
	smoothness (zero mean)	radius (zero mean)
smoothness (zero mean)	7.539518	8.868854
radius (zero mean)	8.868854	23.819936

Covariance matrix

	smoothness (zero mean)	radius (zero mean)
272	5.311833	6.508
273	-1.805167	-5.500
274	-0.790167	2.688
275	0.465833	-3.352
276	-1.990167	-3.912
277	-1.192167	3.568

! =

	smoothness (zero mean)	radius (zero mean)
smoothness (zero mean)	7.539518	8.868854
radius (zero mean)	8.868854	23.819936



Diagonalization of covariance matrix:

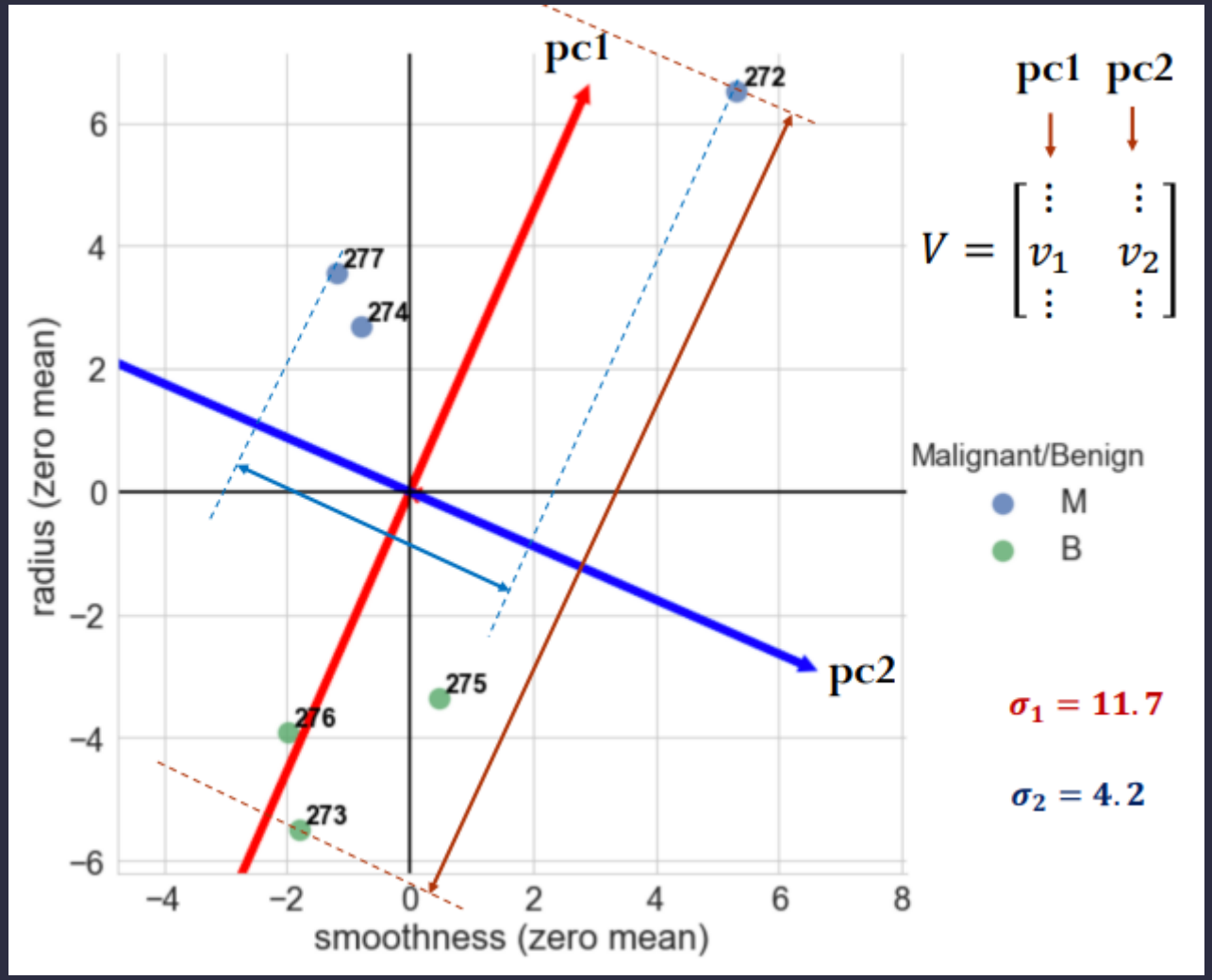
$$A^T A = XDX^T$$

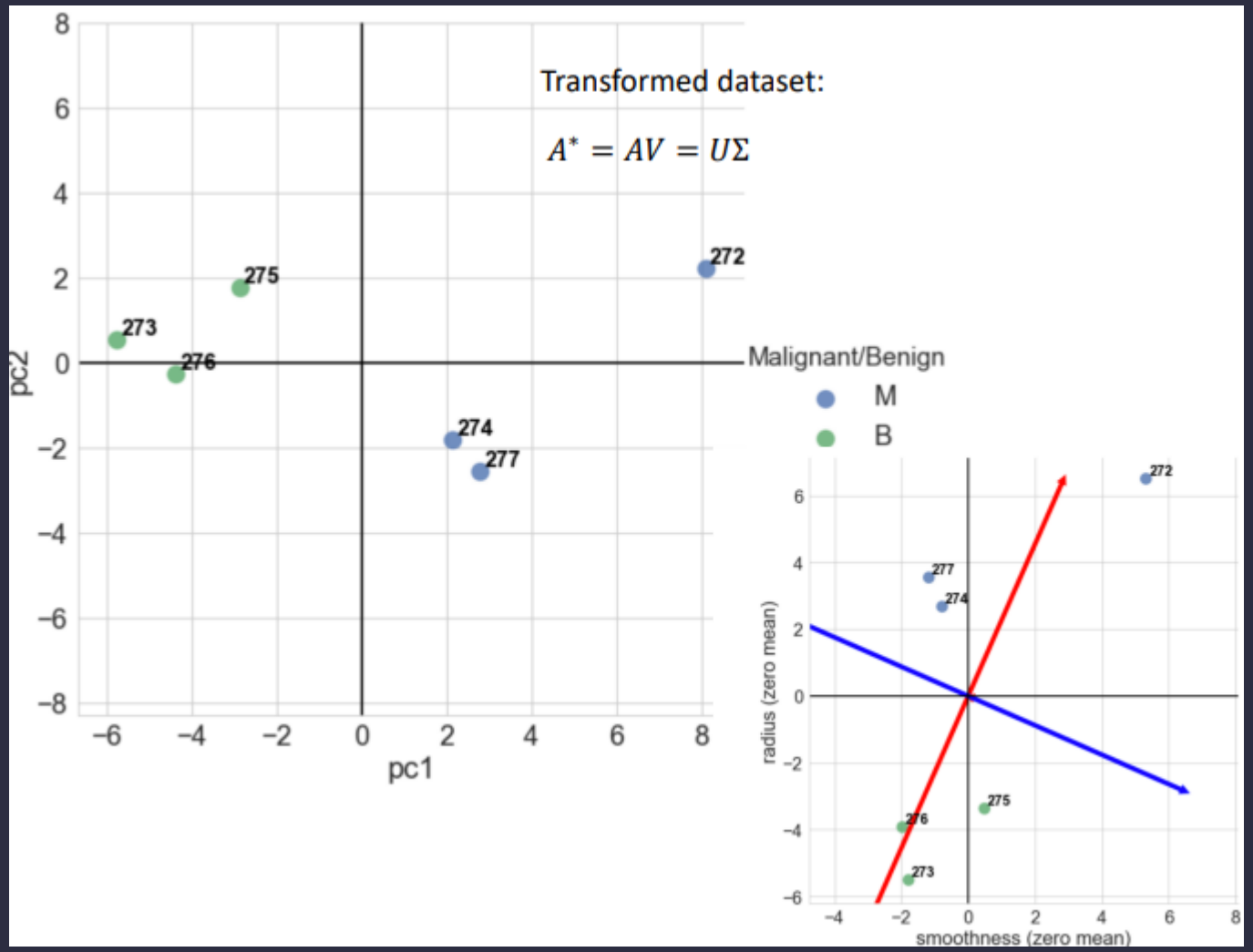
X: eigenvectors of $A^T A$
D: eigenvalues of $A^T A$

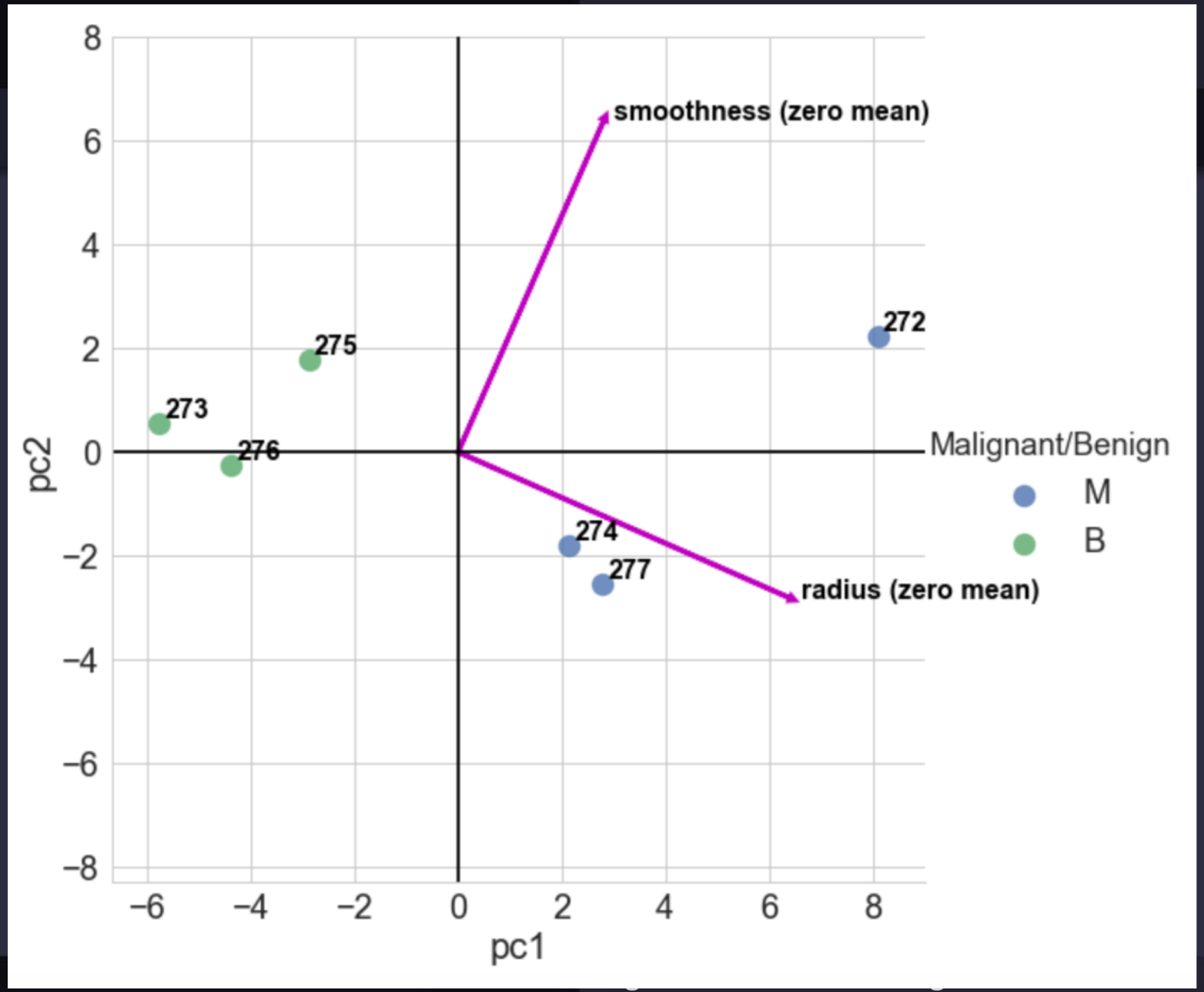
From SVD: $A = U\Sigma V^T$

Maximum variance:
largest singular value of Σ

Direction of maximum variance:
Corresponding column of V


pc1
pc2
272
277
274
275
276
273
radius (zero mean)
smoothness (zero mean)
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6
8
pc1 pc2
$V = \begin{bmatrix} \vdots & \vdots \\ v_1 & v_2 \\ \vdots & \vdots \end{bmatrix}$
Malignant/Benign
M
B
$\sigma_1 = 11.7$
$\sigma_2 = 4.2$

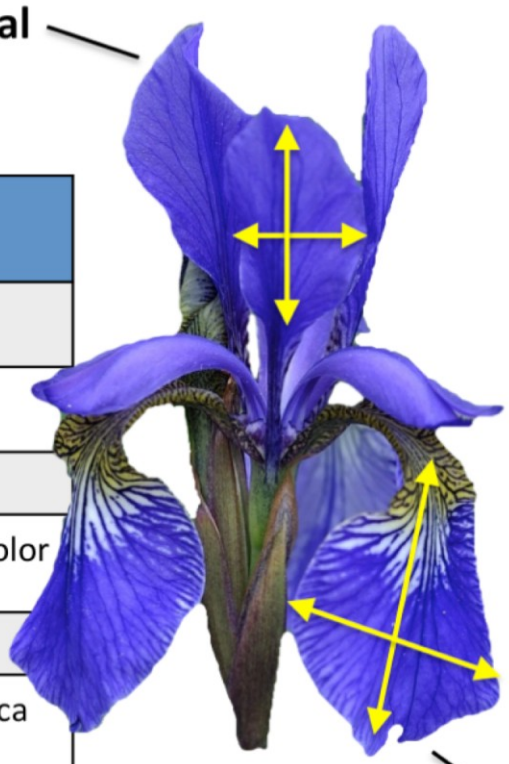

Transformed dataset:
$A^* = AV = U\Sigma$
pc2
pc1
272
275
273
276
274
277
Malignant/Benign
M
B
radius (zero mean)
smoothness (zero mean)


smoothness (zero mean)
radius (zero mean)
272
275
273
276
274
277
Malignant/Benign
M
B
pc1
pc2
8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8

Let’s add more features! Flower classification

Samples
(instances, observations)

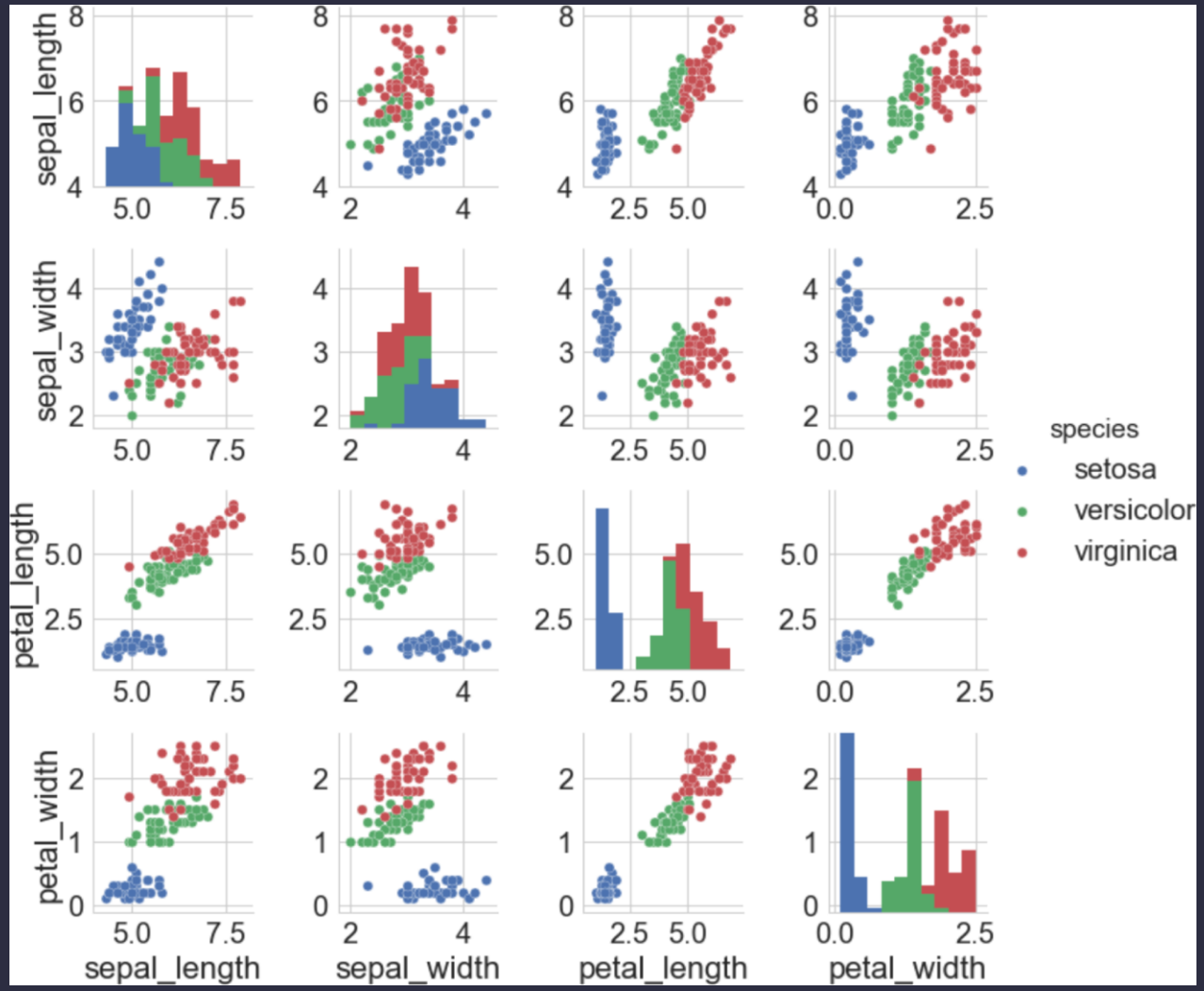
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica



Features
(attributes, measurements, dimensions)

Class labels
(targets)

Let’s add more features! Flower classification



Principal component analysis

How can we reduce the dimension of a dataset without missing important information?

Detect correlation between variables, if a strong correlation exists, then reducing the dimension of the dataset makes sense.

Overall idea: Find the directions of maximum variance in high-dimensional dataset (n dimension) and project it onto a subspace with smaller dimension (k dimension, with $k < n$), while retaining most of the information.

What is the adequate value for k?

Steps of PCA

1) Shift the dataset to zero mean: $A = A - A.mean()$

2) Compute SVD: $A = U\Sigma V^T$

3) Principal components: variances = singular values squared

4) Principal directions: columns of V

5) New dataset: $A^* = A\,V$

Note how the variances of the new dataset correspond to the singular values squared of the original dataset:

$$(A^*)^T A = V^T A^T A\,V = V^T (U\Sigma V^T)^T U\Sigma V^T V = \Sigma^T \Sigma$$

6) In general: $A^* = A\,V$

$m \times n$ $\quad$ $m \times n$ $\quad$ $n \times n$

7) But since we want to reduce the dimension of the dataset, we only use the first k columns of V

$A^* = A\,V$

$m \times k$ $\quad$ $m \times n$ $\quad$ $n \times k$

Iris dataset

Load and observe Dataset

Code Block

```
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris(as_frame=True)
df = iris.frame
df.head()

print(iris.keys())

print("feature_names:", iris.feature_names)
print("target_names :", iris.target_names)

print("data shape   :", iris.data.shape)    # (150, 4)
print("target shape :", iris.target.shape)  # (150,)

print("first 5 rows of data:\n", iris.data[:5])
print("first 5 targets      :", iris.target[:5])
```

Iris dataset

Load and observe Dataset

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
feature_names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
target_names : ['setosa' 'versicolor' 'virginica']
data shape   : (150, 4)
target shape : (150,)
first 5 rows of data:
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
first 5 targets      : 0    0
1    0
2    0
3    0
4    0
```

Iris dataset

Shift the dataset to zero mean:

Code Block

```
X = df.iloc[:, :4]                    # sadece sayısal 4
özellik
X_centered = X - X.mean(axis=0)       # 0-mean
Z = X_centered / X.std(ddof=0)        # std=1 (population
std; PCA oranlarını etkilemez)

print("After standardization (mean≈0, std≈1):")
print(pd.DataFrame({"mean": Z.mean(), "std":
Z.std(ddof=0)}).T.round(4), "\n")
```

```
After standardization (mean≈0, std≈1):
      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
mean               -0.0              -0.0               -0.0              -0.0
std                 1.0               1.0                1.0               1.0
```

In this new dataset Z each feature has mean zero and standard deviation 1.

This decision depends on the problem you are solving. If some variables have a large variance and some small, since PCA maximizes the variance, it will weight more the features with large variance. If you want your PCA to be independent of the variance, standardizing the features will do that.

Iris dataset

Singular Value Decomposition

Code Block

```
U, S, Vt = np.linalg.svd(Z, full_matrices=False)
V = Vt.T                                    # loadings
variances = S**2
explained_var_ratio = variances / variances.sum()

svd_report = pd.DataFrame({
    "SingularValue": S,
    "Variance(S^2)": variances,
    "ExplainedVarianceRatio": explained_var_ratio
}, index=[f"PC{i+1}" for i in range(len(S))])

print("SVD → variances & explained variance ratio:")
print(svd_report.round(6), "\n")
```

```
SVD → variances & explained variance ratio:
     SingularValue  Variance(S^2)  ExplainedVarianceRatio
PC1       20.923066     437.774672                0.729624
PC2       11.709166     137.104571                0.228508
PC3        4.691858      22.013531                0.036689
PC4        1.762732       3.107225                0.005179
```

Iris dataset

Reduced Features

Code Block

```
Zstar = Z.values @ V[:, :2]
Zstar_df = pd.DataFrame(Zstar, columns=["p0", "p1"])
Zstar_df["species"] = df["species"].values

print("Reduced 2D representation (first 5 rows):")
print(Zstar_df.head(), "\n")
```

```
Reduced 2D representation (first 5 rows):
         p0        p1 species
0 -2.264703 -0.480027  setosa
1 -2.080961  0.674134  setosa
2 -2.364229  0.341908  setosa
3 -2.299384  0.597395  setosa
4 -2.389842 -0.646835  setosa
```

Iris dataset

Features P0 & P1

Code Block

```
loadings = pd.DataFrame(V[:, :2], index=features,
columns=["p0", "p1"])
print("Feature loadings on p0 & p1:")
print(loadings.round(4), "\n")
```

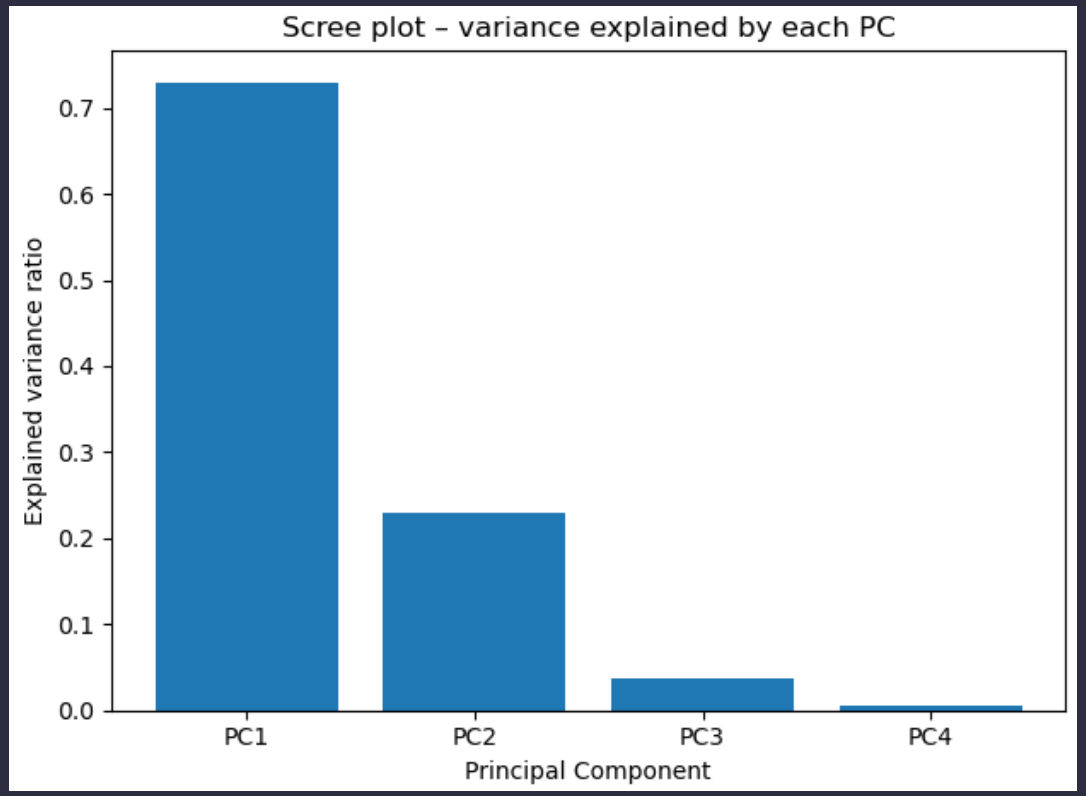
```
Feature loadings on p0 & p1:
                     p0      p1
sepal length (cm)  0.5211 -0.3774
sepal width (cm)  -0.2693 -0.9233
petal length (cm)  0.5804 -0.0245
petal width (cm)   0.5649 -0.0669
```

Iris dataset

Feature Plots

Code Block

```
plt.figure()
x = np.arange(1, len(explained_var_ratio) + 1)
plt.bar(x, explained_var_ratio)
plt.xticks(x, [f"PC{i}" for i in x])
plt.ylabel("Explained variance ratio")
plt.xlabel("Principal Component")
plt.title("Scree plot – variance explained by each PC")
plt.tight_layout()
```

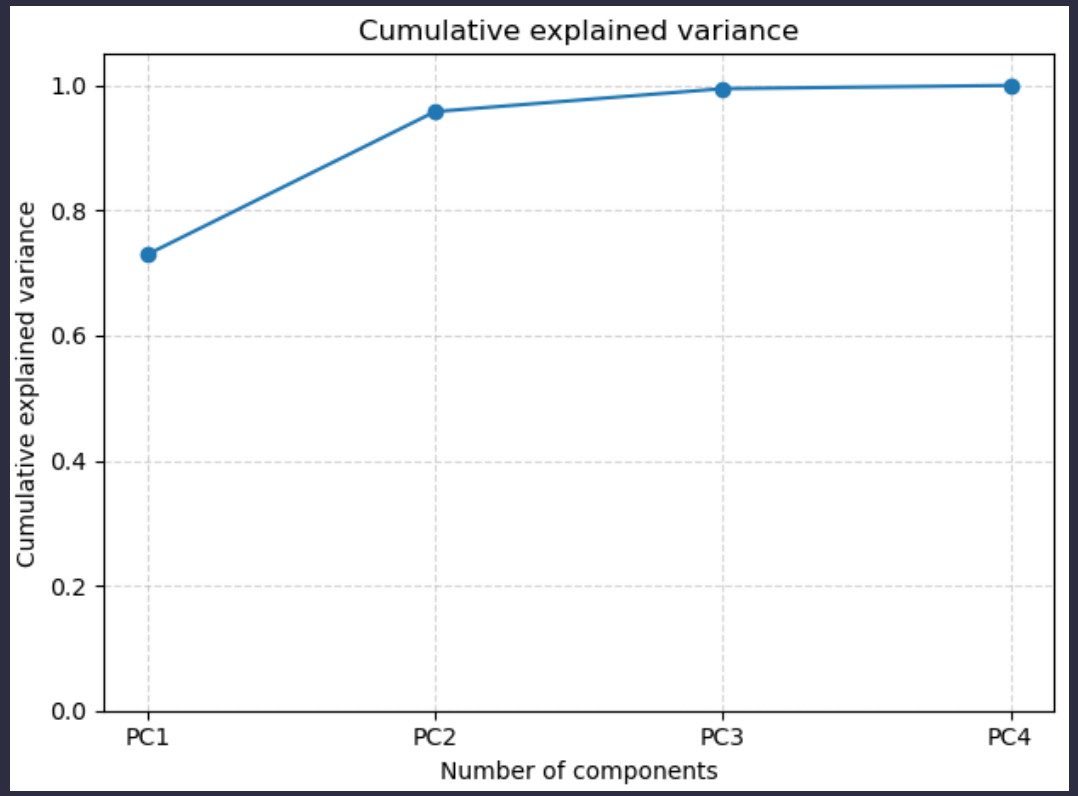


Iris dataset

FeCumulative Variance

Code Block

```
plt.figure()
cum = np.cumsum(explained_var_ratio)
plt.plot(x, cum, marker="o")
plt.xticks(x, [f"PC{i}" for i in x])
plt.ylim(0, 1.05)
plt.ylabel("Cumulative explained variance")
plt.xlabel("Number of components")
plt.title("Cumulative explained variance")
plt.grid(True, linestyle="--", alpha=0.5)
plt.tight_layout()
```

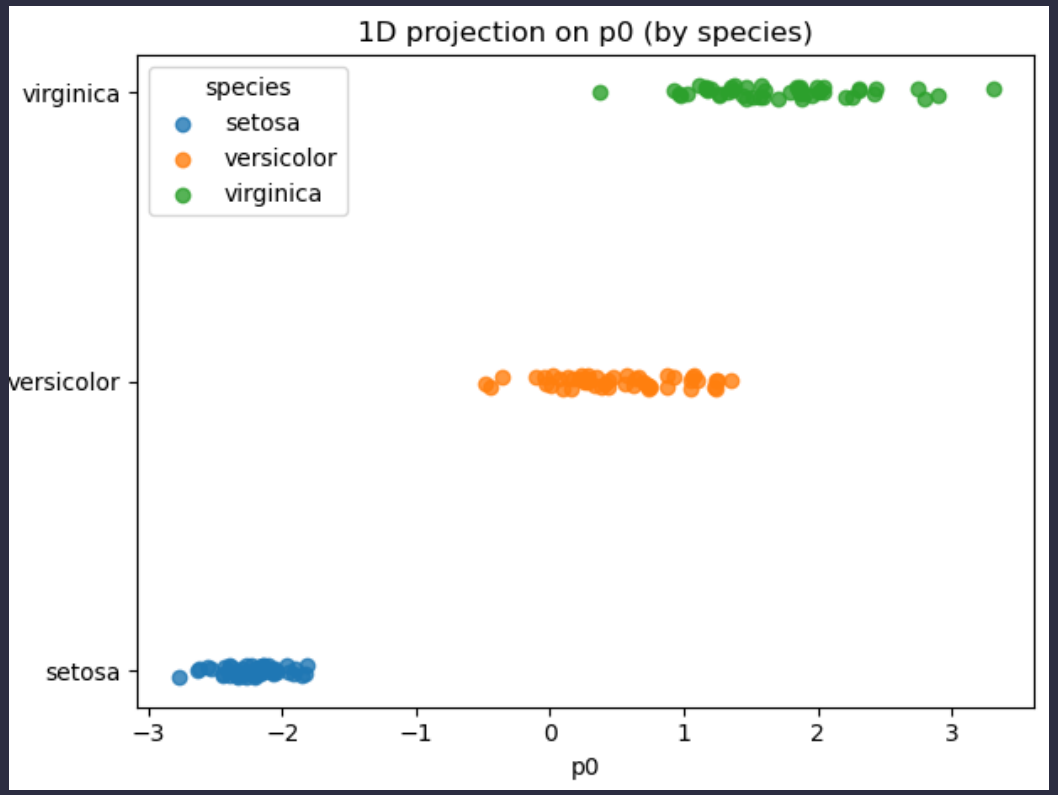


Iris dataset

P0 Feature Classes

Code Block

```
plt.figure()
species_order = ["setosa", "versicolor", "virginica"]
for i, sp in enumerate(species_order):
    mask = Zstar_df["species"] == sp
    y = np.full(mask.sum(), i) +
(np.random.rand(mask.sum()) - 0.5) * 0.05
    plt.scatter(Zstar_df.loc[mask, "p0"], y, label=sp,
alpha=0.8)
plt.yticks(range(len(species_order)), species_order)
plt.xlabel("p0")
plt.title("1D projection on p0 (by species)")
plt.legend(title="species")
plt.tight_layout()
```

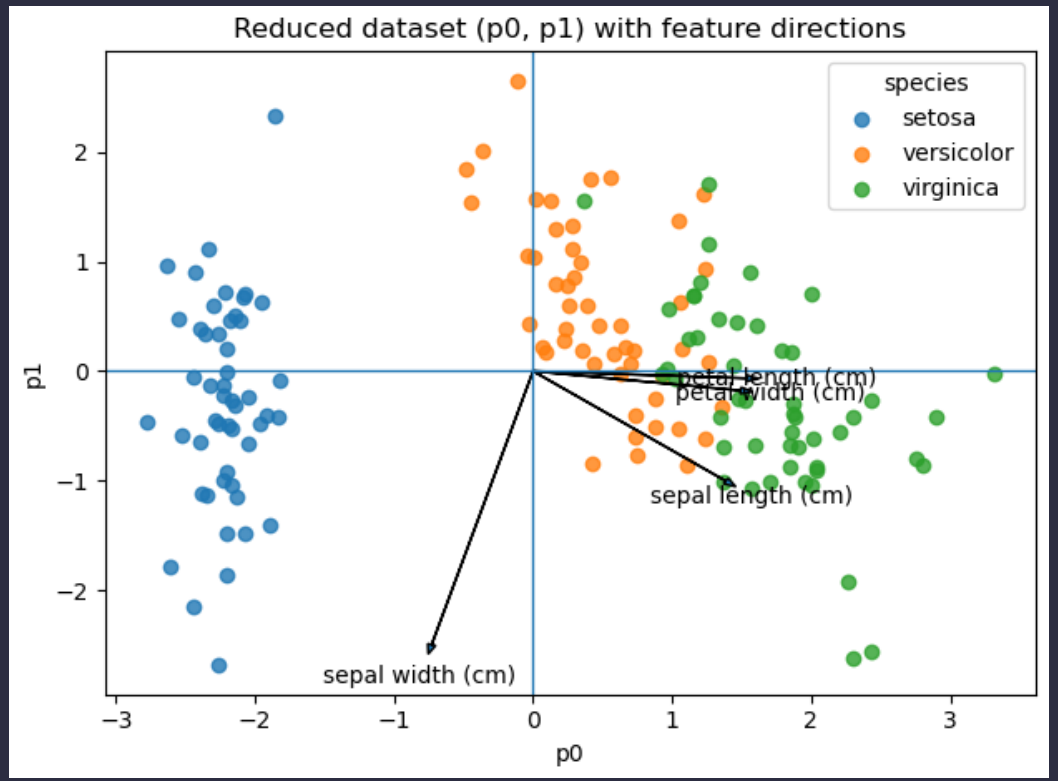


Iris dataset

P0 & P1 Feature Classes

Code Block

```
plt.figure()
for sp in species_order:
    m = Zstar_df["species"] == sp
    plt.scatter(Zstar_df.loc[m, "p0"], Zstar_df.loc[m, "p1"], label=sp,
alpha=0.8)

scale = 2.8
for i, feat in enumerate(features):
    x0, y0 = 0.0, 0.0
    x1, y1 = V[i, 0] * scale, V[i, 1] * scale
    plt.arrow(x0, y0, x1, y1, head_width=0.06, length_includes_head=True)
    plt.text(x1 * 1.08, y1 * 1.08, feat, ha="center", va="center")
plt.axhline(0, linewidth=1)
plt.axvline(0, linewidth=1)
plt.xlabel("p0")
plt.ylabel("p1")
plt.title("Reduced dataset (p0, p1) with feature directions")
plt.legend(title="species")
plt.tight_layout()

plt.show()
```

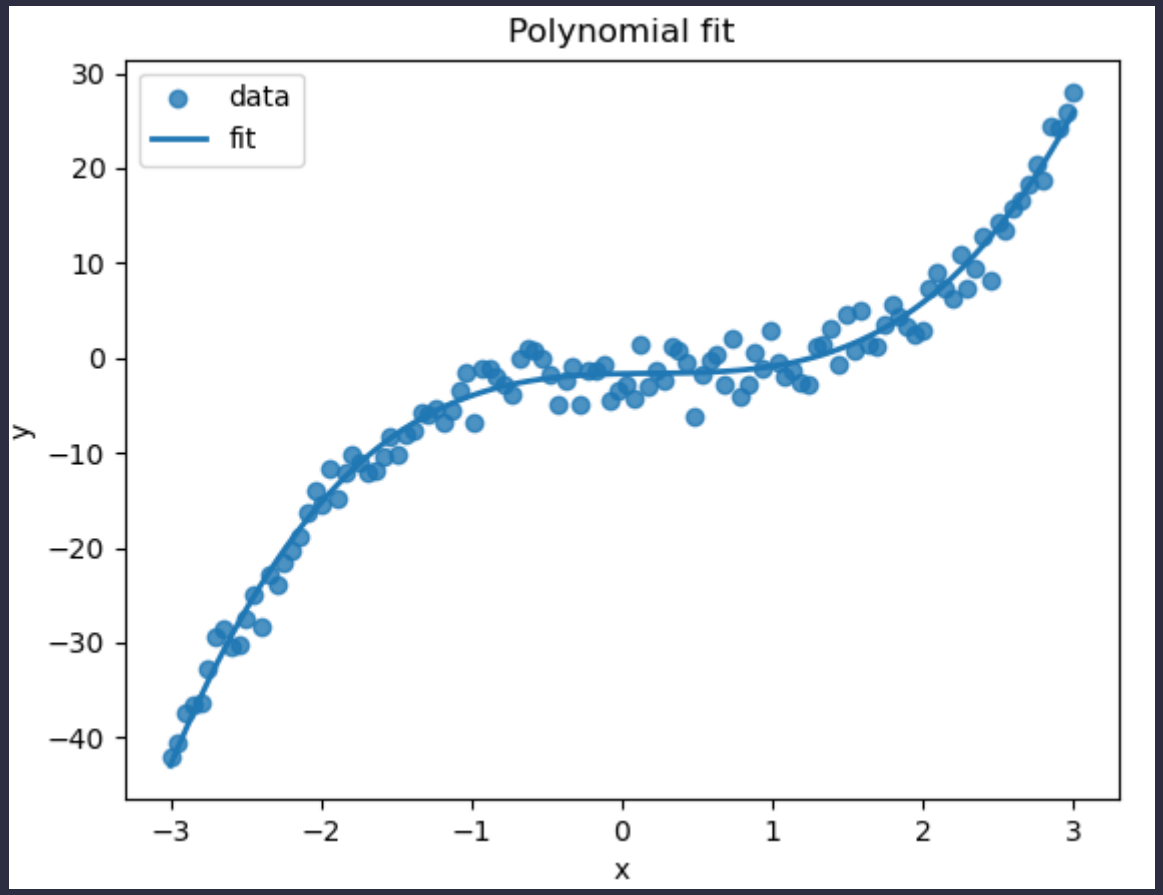


Regress Analysis

- Linear regression: $Y = \alpha + \beta X$
  - Two parameters , α and β specify the line and are to be estimated by using the data at hand.
  - using the least squares criterion to the known values of *Y1, Y2, …, X1, X2, ….*
- Multiple regression: $Y = b0 + b1\ X1 + b2\ X2$.
  - Many nonlinear functions can be transformed into the above.

Code Block

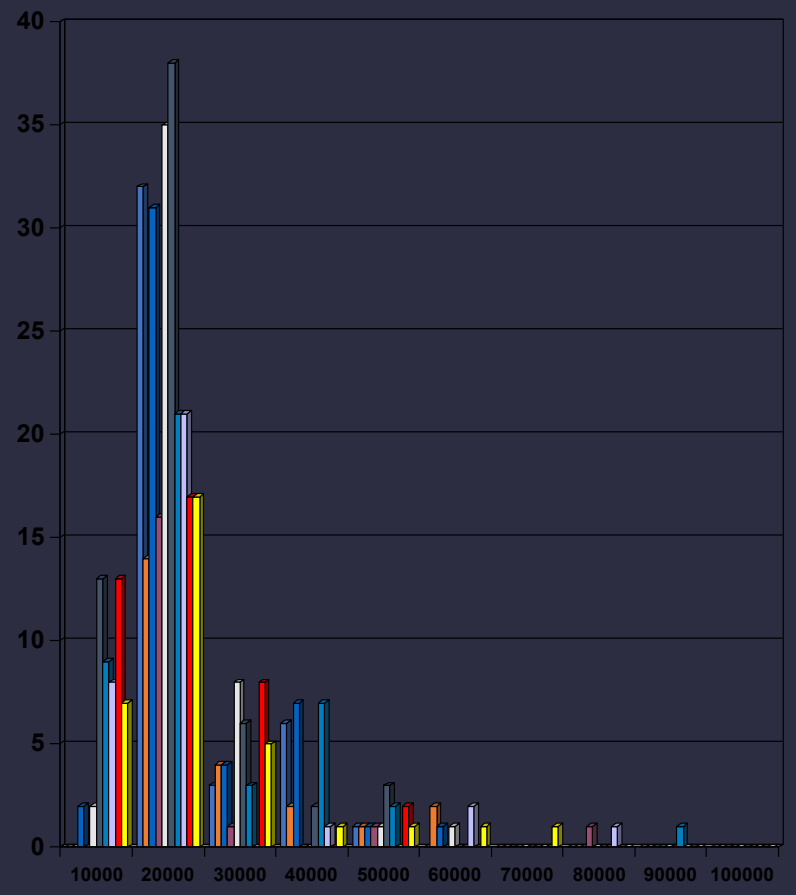
```
import numpy as np
import matplotlib.pyplot as plt

rng = np.random.default_rng(0)
x = np.linspace(-3, 3, 120)
y_true = 1.2*x**3 - 0.7*x**2 + 0.5*x - 2
y = y_true + rng.normal(0, 2.0, size=len(x))

deg = 3
coeffs = np.polyfit(x, y, deg)      # descending powers
y_hat = np.polyval(coeffs, x)

ss_res = np.sum((y - y_hat)**2)
ss_tot = np.sum((y - np.mean(y))**2)
r2 = 1 - ss_res/ss_tot

def fmt_eq(c):
    d = len(c) - 1
    parts = []
    for i, a in enumerate(c):
        p = d - i
        if p == 0: parts.append(f"{a:+.3g}")
        elif p == 1: parts.append(f"{a:+.3g}·x")
        else: parts.append(f"{a:+.3g}·x^{p}")
    s = " ".join(parts)
    return ("ŷ(x) = " + s[1:]) if s.startswith("+") else "ŷ(x) = " + s

print("Coefficients (descending):", coeffs)
print("Equation:", fmt_eq(coeffs))
print(f"R^2: {r2:.5f}")

plt.scatter(x, y, alpha=0.8, label="data")
plt.plot(x, y_hat, linewidth=2, label="fit")
plt.title("Polynomial fit")
plt.xlabel("x"); plt.ylabel("y"); plt.legend(); plt.show()
```




```
Coefficients (descending): [ 1.24177935 -0.75844609  0.31349136 -1.65928557]
Equation: ŷ(x) = 1.24·x^3 -0.758·x^2 +0.313·x -1.66
```

Histograms

- A popular data reduction technique
- Divide data into buckets and store average (sum) for each bucket
- Can be constructed optimally in one dimension using dynamic programming
- Related to quantization problems.



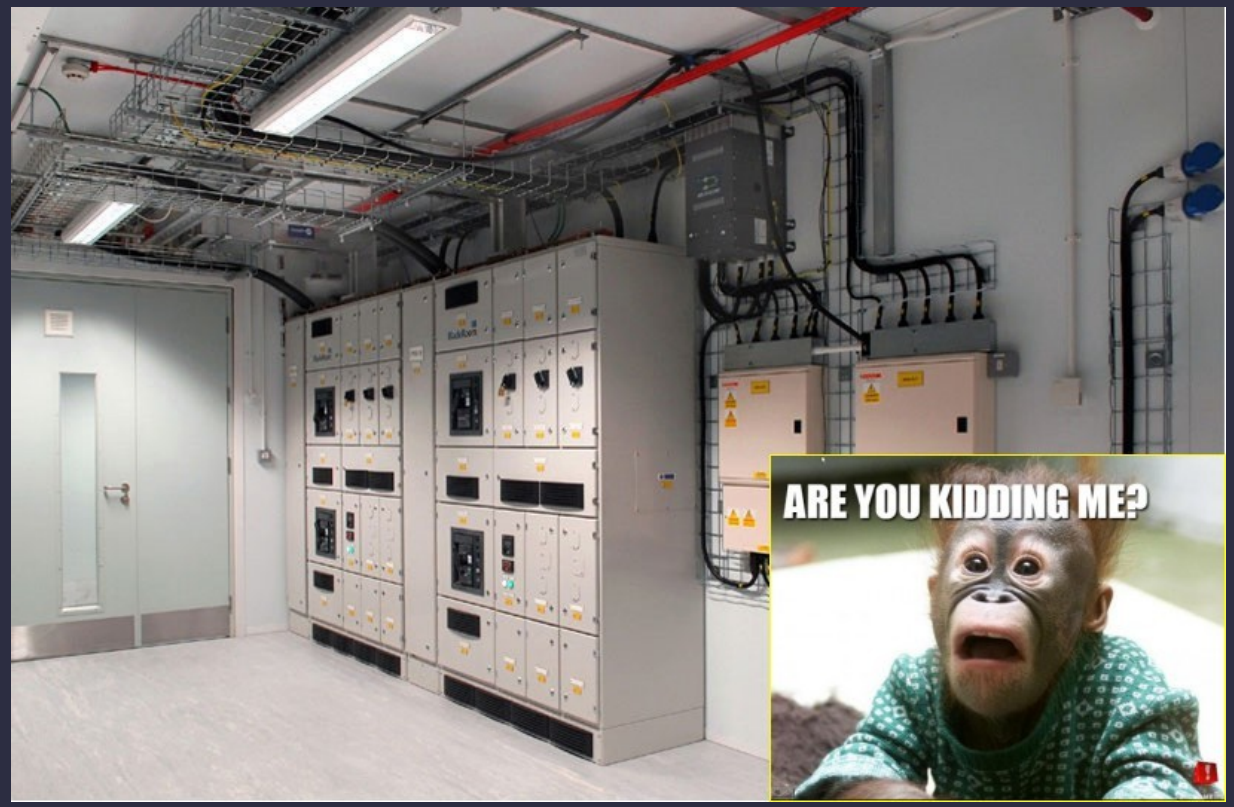
The Escalator Problem

Energy efficient escalators:

- ON when there are pedestrians
- STAND-BY when there is no pedestrian for several seconds
- How much saving?

That’s Easy!

- Go to the meter room
- Measure it!!!



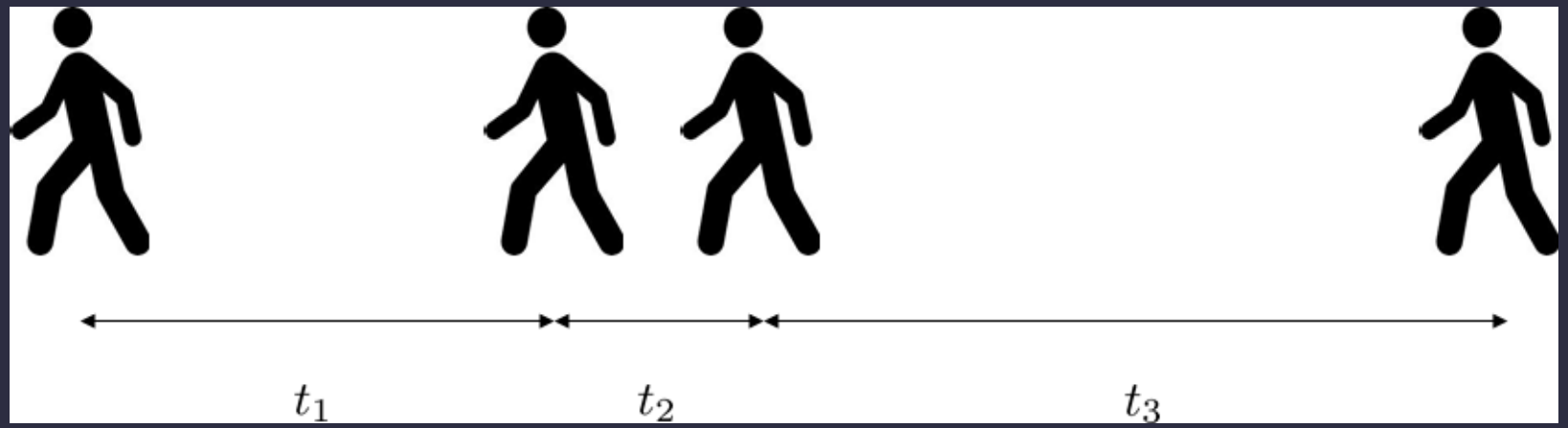
But what if you have not yet built the escalator?

Let’s collect data

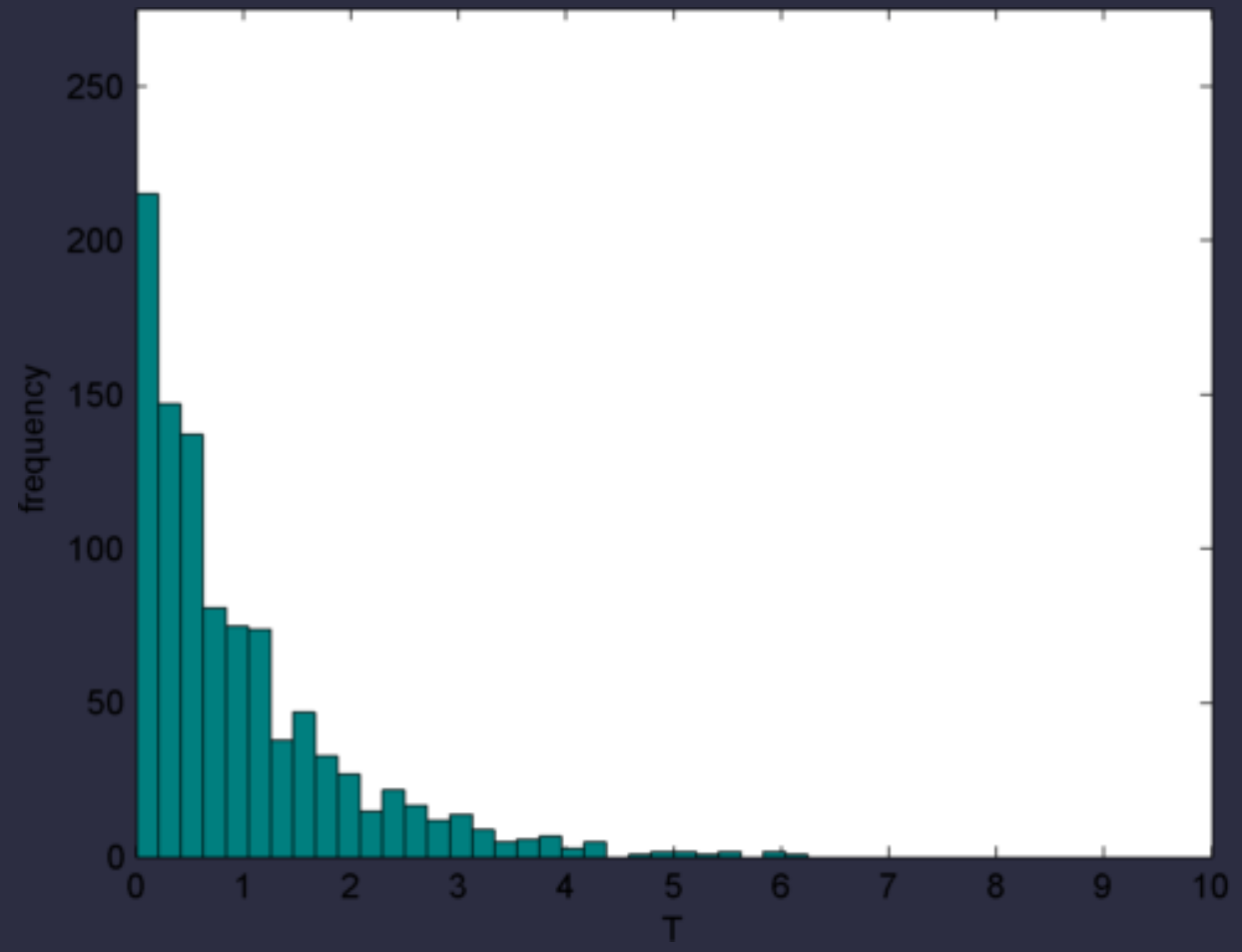
Inter-arrival Time

Let T be the inter-arrival time.

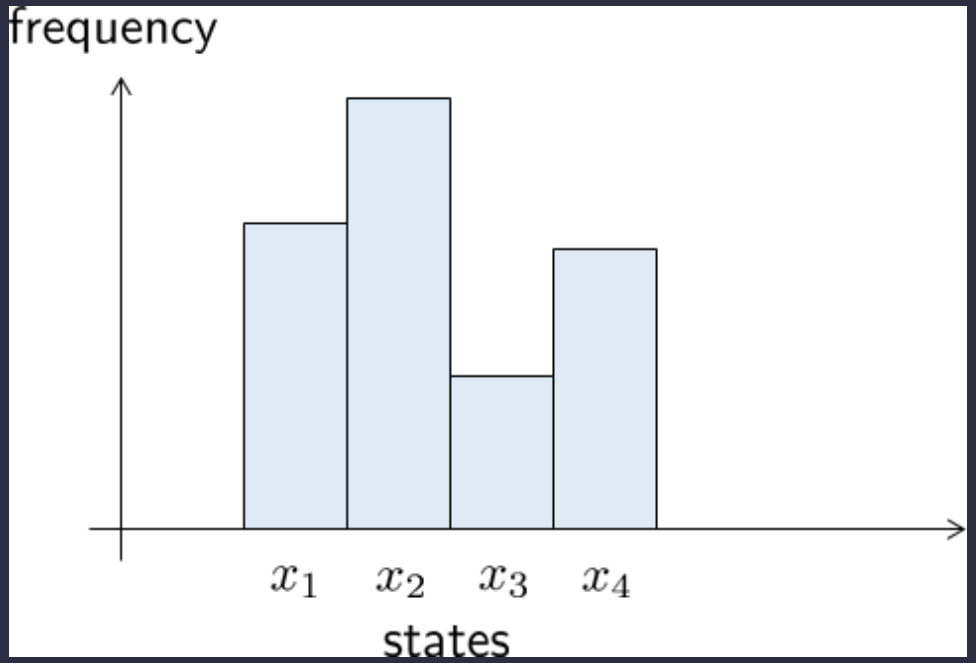
Possible values of T: Call them $t_1, t_2, t_3, \ldots,$



How does the histogram of T look like?



What can be told from a histogram?



- Set of all possible state: $x_1, x_2, \ldots, x_m$.
- Empirical **frequency** of each state: $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_m$.

Important!

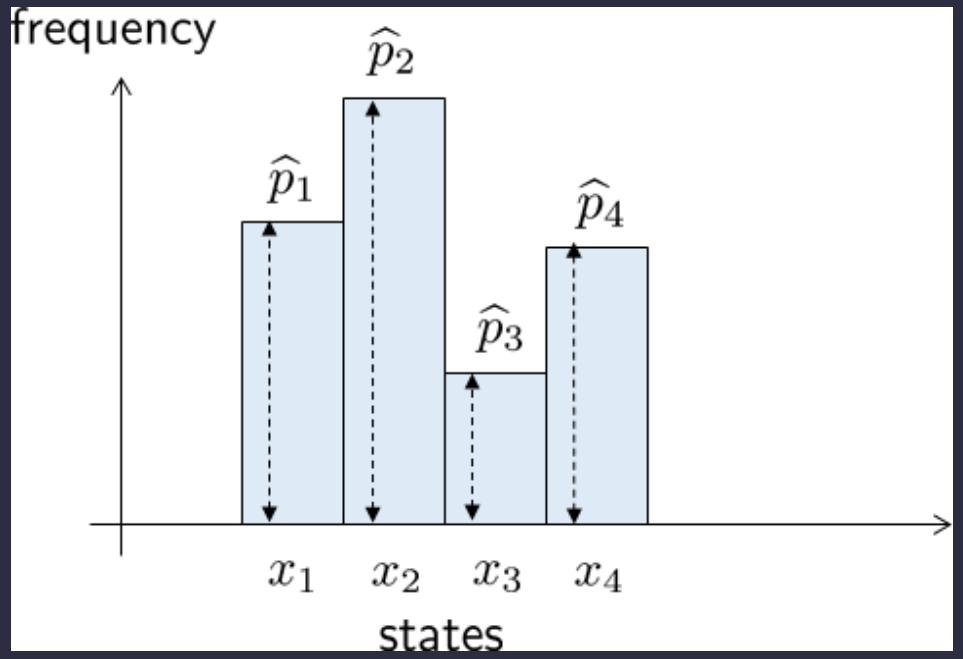
$$\hat{p}_1 + \hat{p}_2 + \ldots + \hat{p}_m = 1.$$

What can be told from a histogram?

Sample Mean:

$$\overline{X} = \sum_{i=1}^{m} \hat{p}_i x_i$$

- "Average" of computed from the histogram

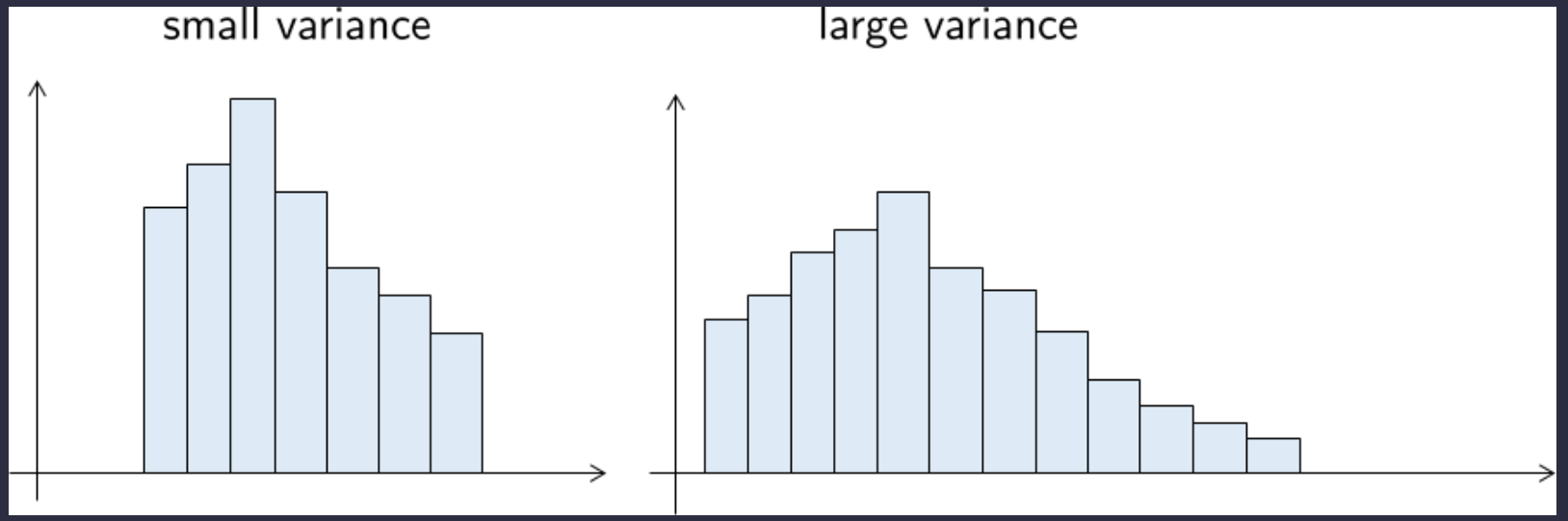


What can be told from a histogram?

Sample Variance:

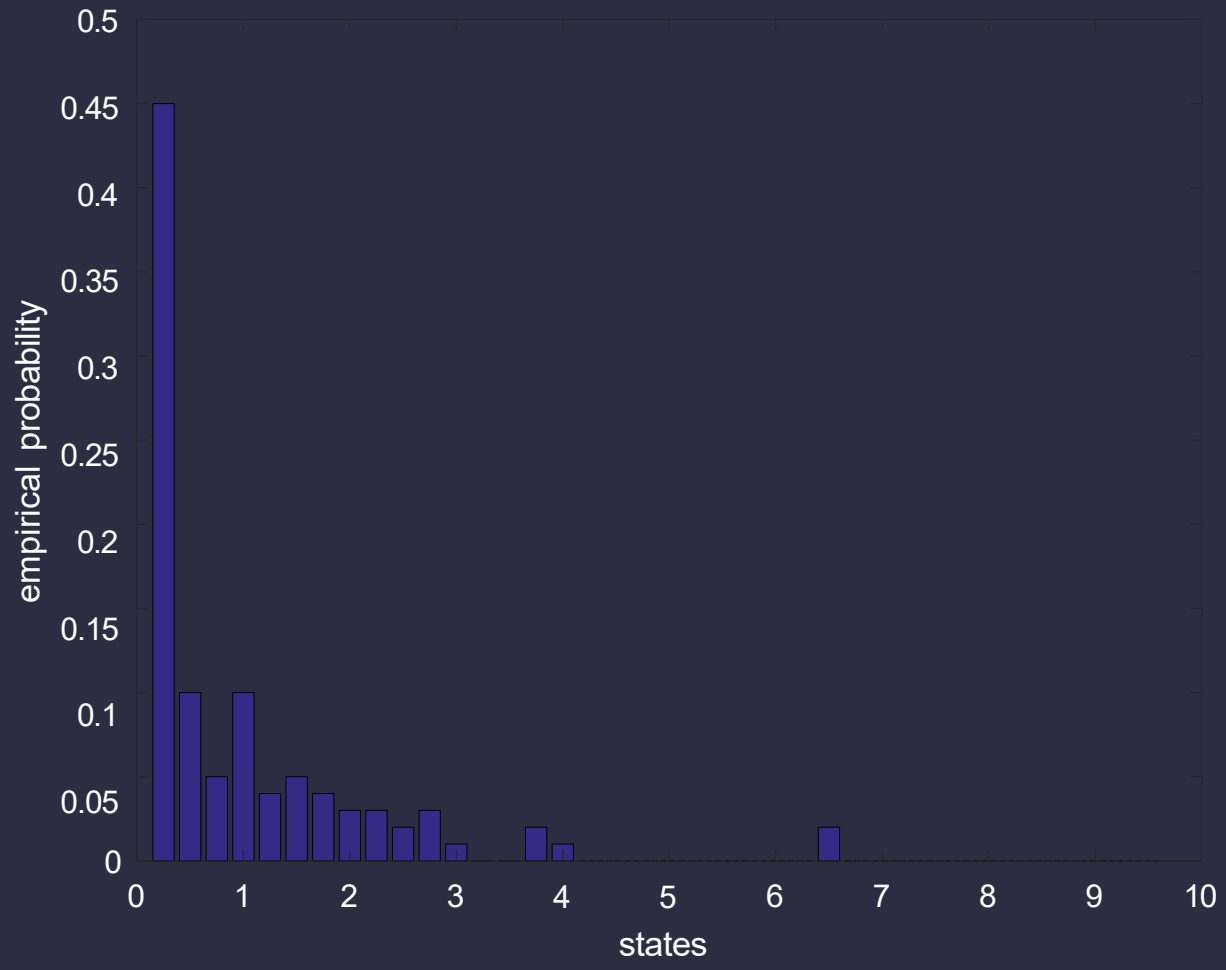
$$S^2 = \sum_{i=1}^{m} p_i (x_i - \bar{X})^2$$

- Measures the deviation
- Large S^2 means that the histogram is wide-spread
- S is the sample standard deviation



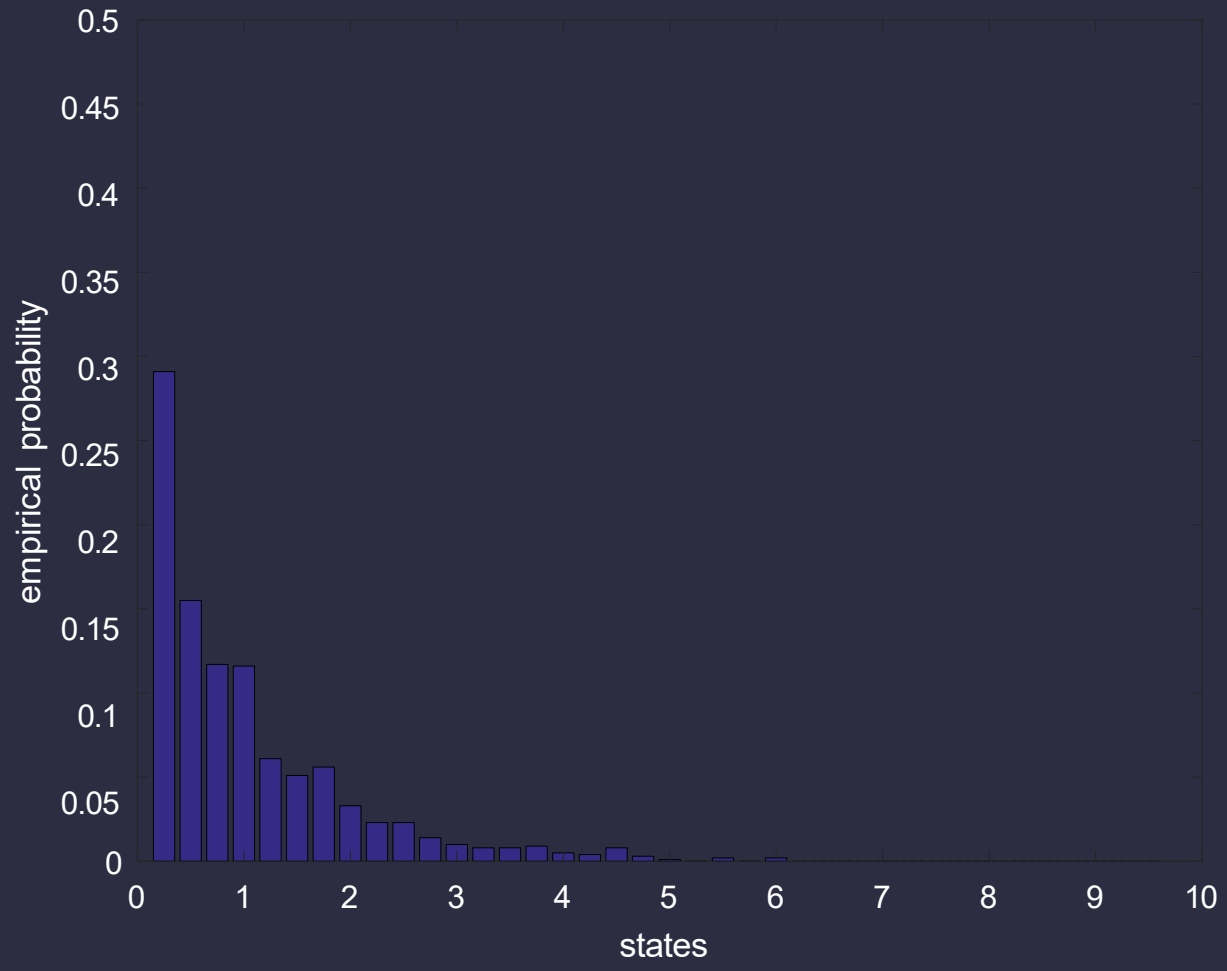
Histogram Grows

What if we have 100 measurements?



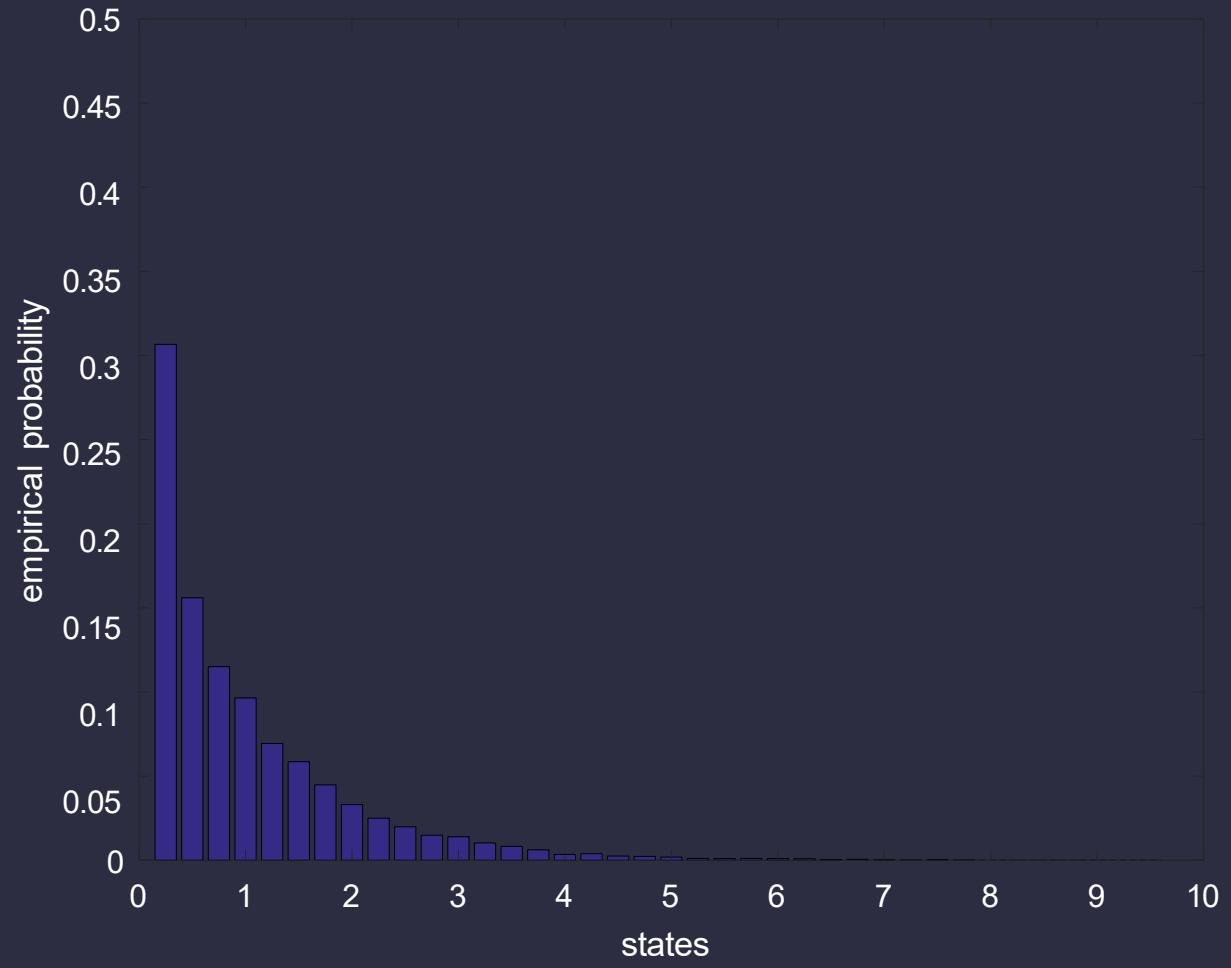
Histogram Grows

What if we have 1000 measurements?



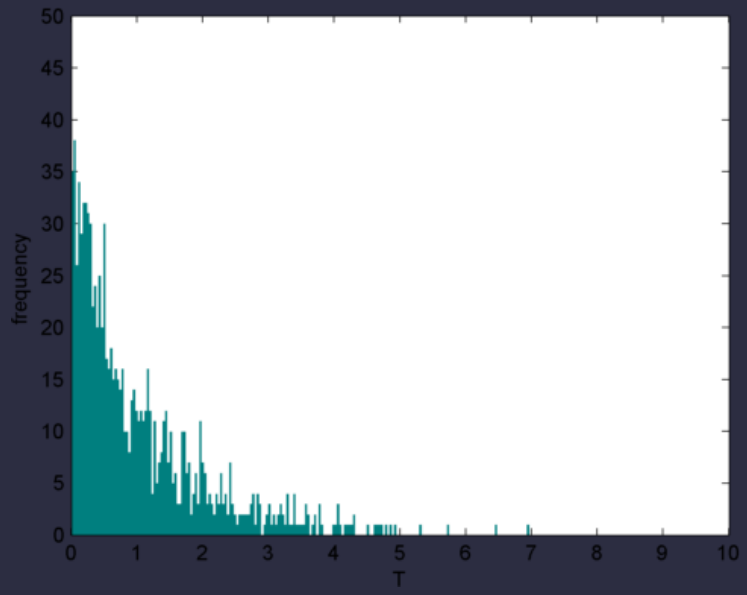
Histogram Grows

What if we have 10000 measurements?

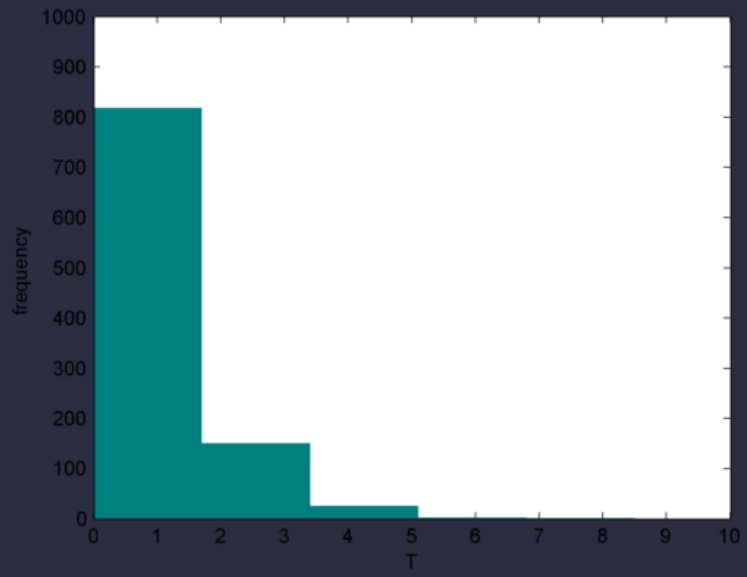


Bin-width of Histogram

Bad choice of bin-width:



200 bins



5 bins

- Too many bins: Not enough data!
- Too few bins: Not descriptive!

Optimal Bin-width

Here is a method to estimate the bin-width. The method is called **Cross-Validation.**

Notations

- n: number of data points
- m: number of bins
- h: bin-width: n/m. (Can round off to nearest integer.)
- $\hat{p}_j$: frequency of the j-th bin.

Cross-validation Score:

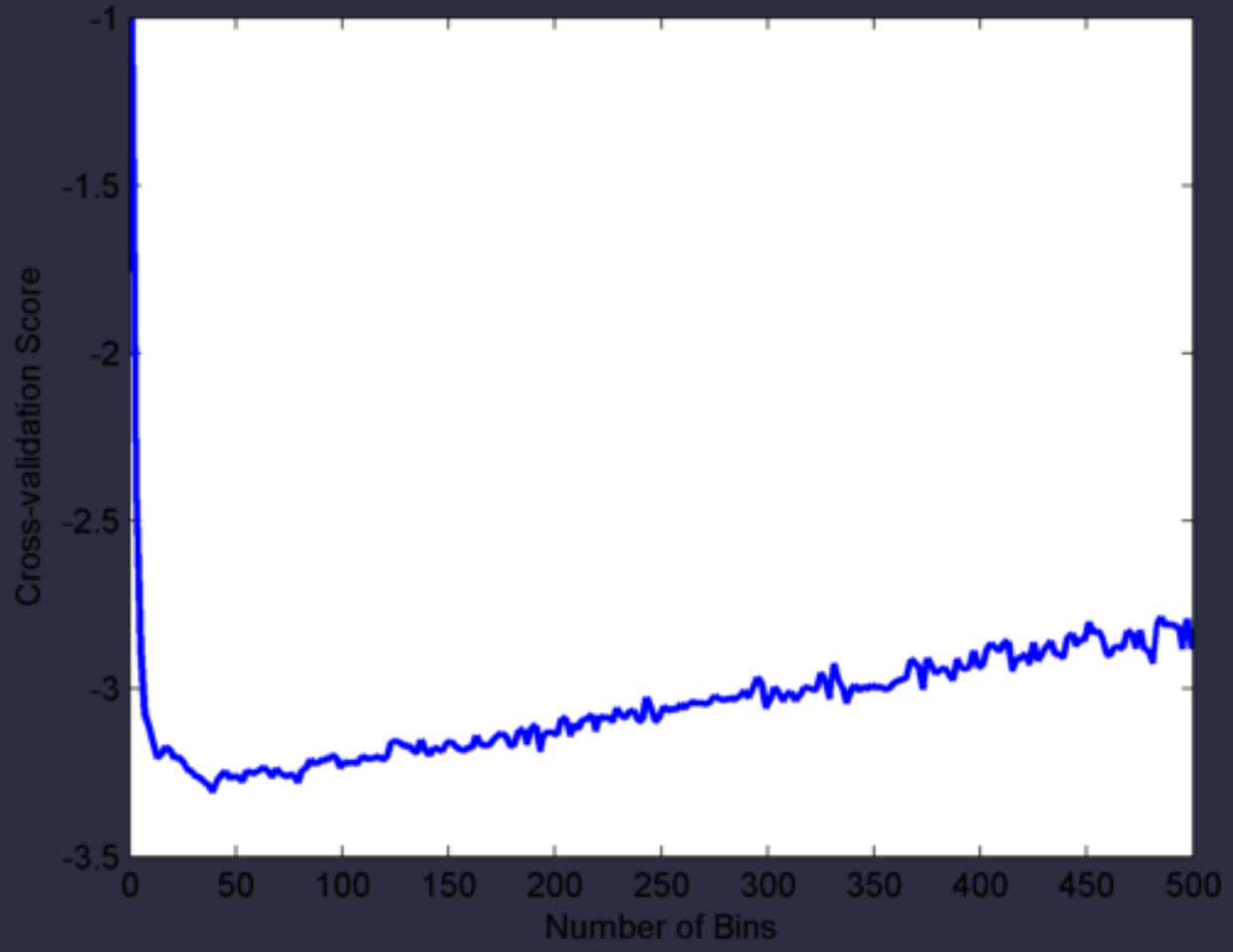
$$J(h) = \frac{2}{(n-1)h} - \frac{n+1}{(n-1)h}\left(\hat{p}_1^2 + \hat{p}_2^2 + \ldots + \hat{p}_m^2\right).$$

Optimal Bin-width

Procedure:

- Pick the number of bins m.
- Since n is fixed, we can compute $h = n/m$.
- Build a histogram of m bins.
- The heights of the histogram bars are $\hat{p}_j$.
- Calculate the Cross-Validation Score $J(h)$.
- If $J(h)$ is high, try another m until $J(h)$ is low enough.

Optimal Bin-width



Regress Analysis

Code Block

```
import numpy as np
import matplotlib.pyplot as plt

def cv_score_histogram(x, h, data_range=None):
    x = np.asarray(x, dtype=float).ravel()
    n = x.size
    if n < 2 or h <= 0:
        return np.inf

    if data_range is None:
        xmin, xmax = float(x.min()), float(x.max())
    else:
        xmin, xmax = map(float, data_range)
m = int(np.ceil((xmax - xmin) / h))
    if m < 1:
        return np.inf
    edges = xmin + np.arange(m + 1) * h

    # Kenar sonu taşmalarına karşı sağlamlık: xmax'i kapsa( t )
    if edges[-1] < xmax:
        edges = np.r_[edges, edges[-1] + h]
        m += 1

    # Bin countları ve oransal frekanslar p_j
    counts, _ = np.histogram(x, bins=edges)
    p = counts / n

    # J(h) = 2/((n-1)h) - ((n+1)/(n-1))*(1/h)*sum p_j^2
    J = (2.0 / ((n - 1) * h)) - ((n + 1.0) / (n - 1.0)) * (np.sum(p ** 2) /
h)
    return J
```

Code Block

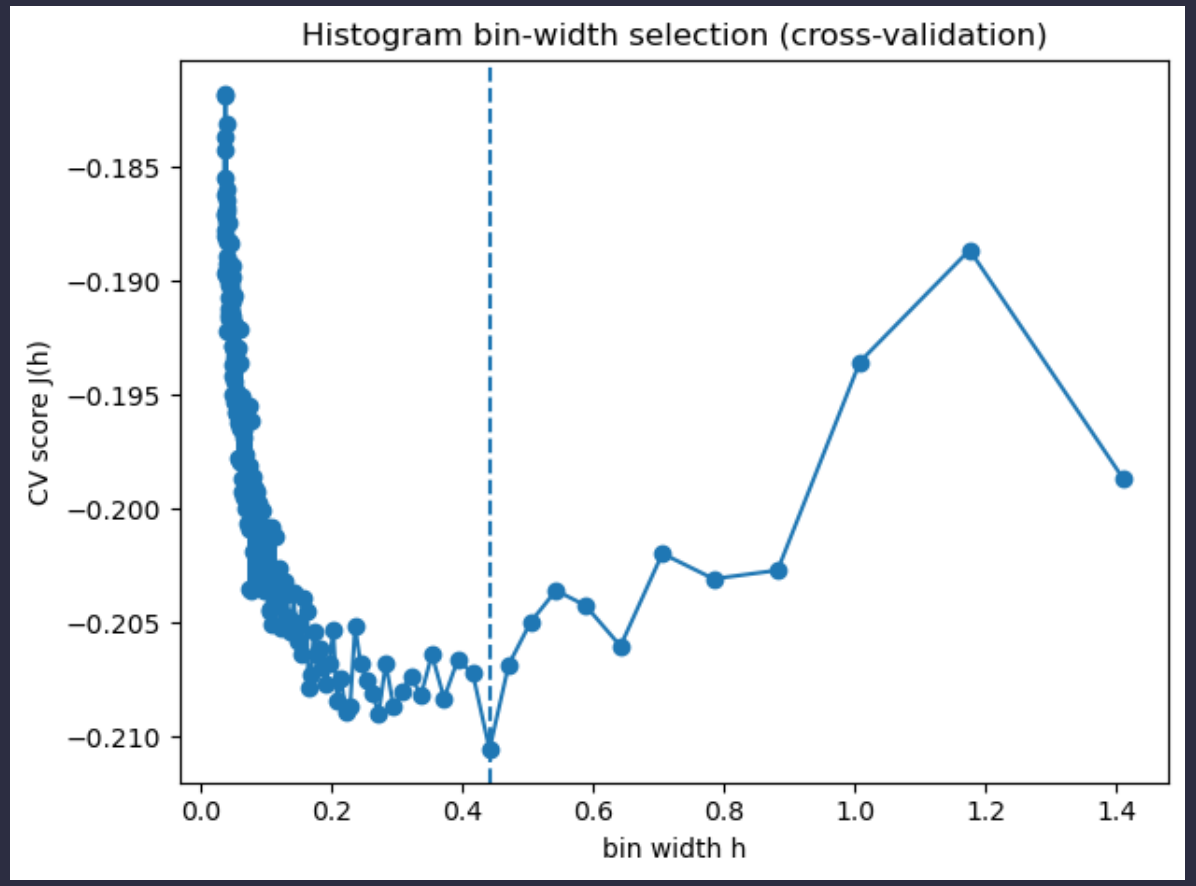
```
def optimal_binwidth_cv(x, m_min=5, m_max=None, data_range=None):    x =
np.asarray(x, dtype=float).ravel()
    n = x.size
    if n < 2:
        raise ValueError("Data length must be >= 2")

    if data_range is None:
        xmin, xmax = float(x.min()), float(x.max())
    else:
        xmin, xmax = map(float, data_range)

    rng = xmax - xmin
    if rng <= 0:
        raise ValueError("Data range must be positive.")

    if m_max is None:
        # makul bir üst sınır: min(200, n//2)
        m_max = max(m_min + 1, min(200, n // 2))

    ms = np.arange(m_min, m_max + 1, dtype=int)
    hs = rng / ms

    scores = np.array([cv_score_histogram(x, h, (xmin, xmax)) for h in
hs])
    idx = int(np.argmin(scores))
    h_opt, m_opt, J_opt = float(hs[idx]), int(ms[idx]),
float(scores[idx])

    return h_opt, m_opt, J_opt, hs, scores
```

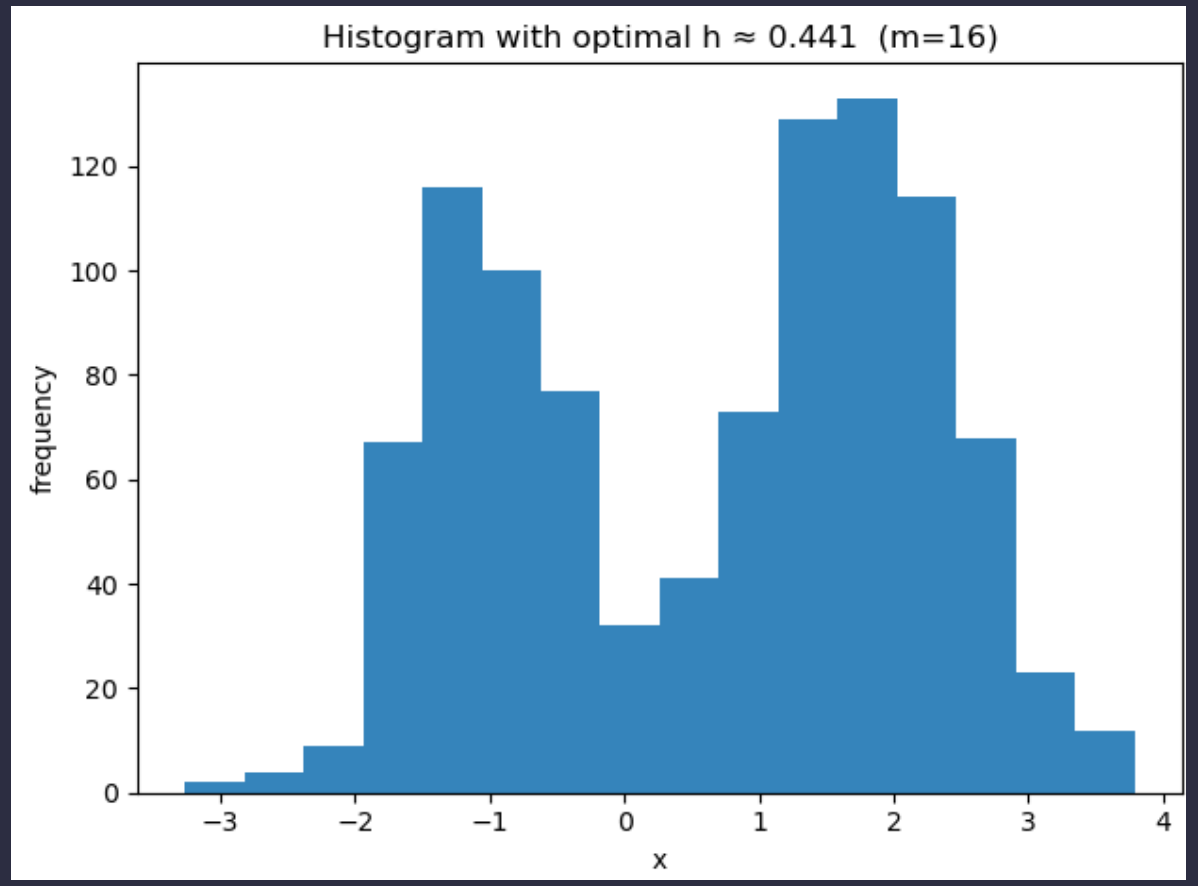
Regress Analysis

Code Block

```
if __name__ == "__main__":
    rng = np.random.default_rng(0) x = np.r_[rng.normal(-1.0, 0.6, 400), rng.normal(1.8, 0.8, 600)]

    h_opt, m_opt, J_opt, hs, scores = optimal_binwidth_cv(x)

    print(f"Optimal bin width h* = {h_opt:.4g}  (m* = {m_opt} bins),  J(h*) = {J_opt:.6g}")

    # Skor eğrisi
    plt.figure()
    plt.plot(hs, scores, marker="o")
    plt.axvline(h_opt, linestyle="--")
    plt.xlabel("bin width h")
    plt.ylabel("CV score J(h)")
    plt.title("Histogram bin-width selection (cross-validation)")
    plt.tight_layout()

    # Seçilen bin genişliğiyle histogram
    xmin, xmax = float(x.min()), float(x.max())
    m = int(np.ceil((xmax - xmin) / h_opt))
    edges = xmin + np.arange(m + 1) * h_opt
    if edges[-1] < xmax:
        edges = np.r_[edges, edges[-1] + h_opt]

    plt.figure()
    plt.hist(x, bins=edges, density=False, alpha=0.9)
    plt.xlabel("x")
    plt.ylabel("frequency")
    plt.title(f"Histogram with optimal h ≈ {h_opt:.3f}  (m={m})")
    plt.tight_layout()
    plt.show()
```

Regress Analysis

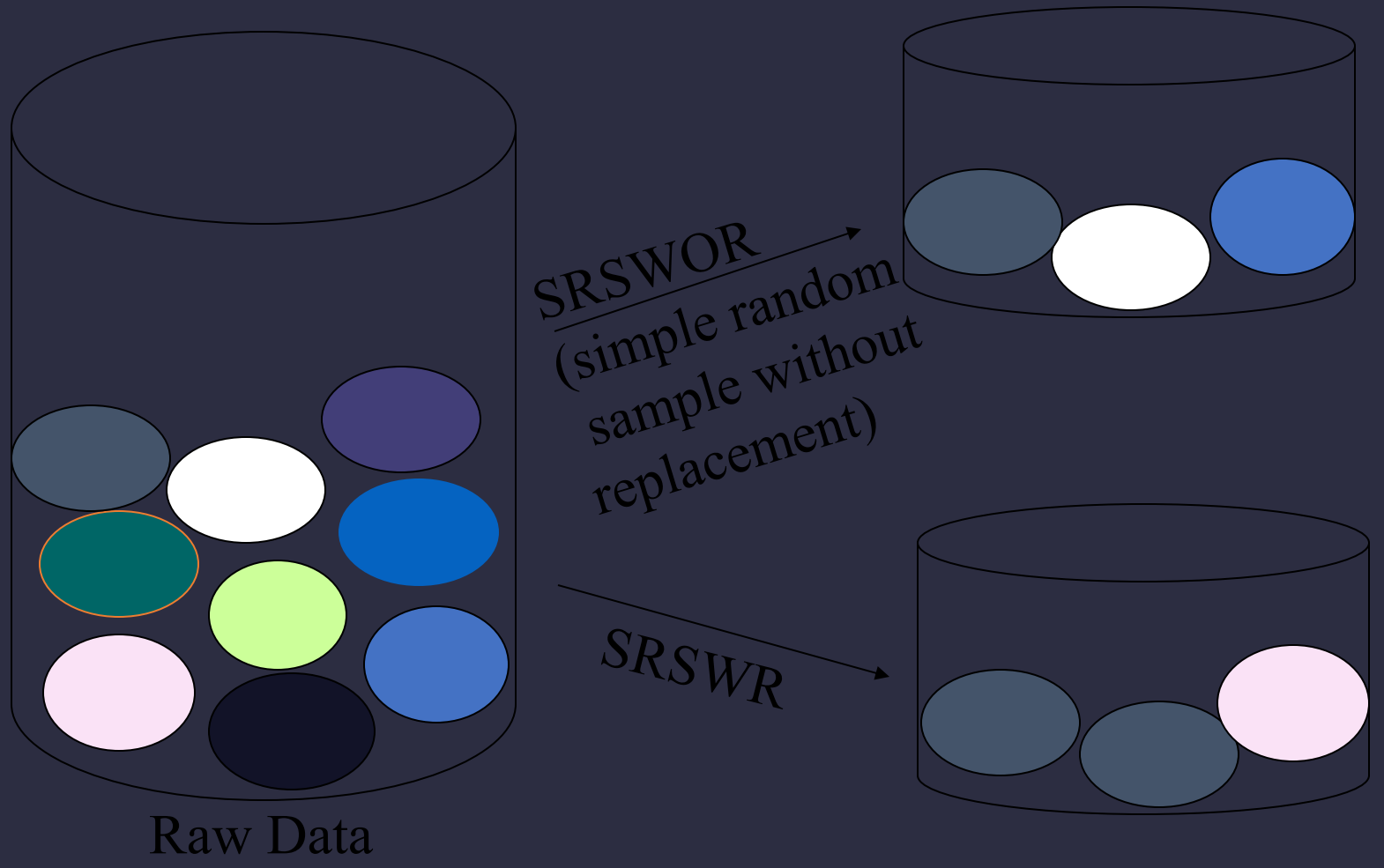




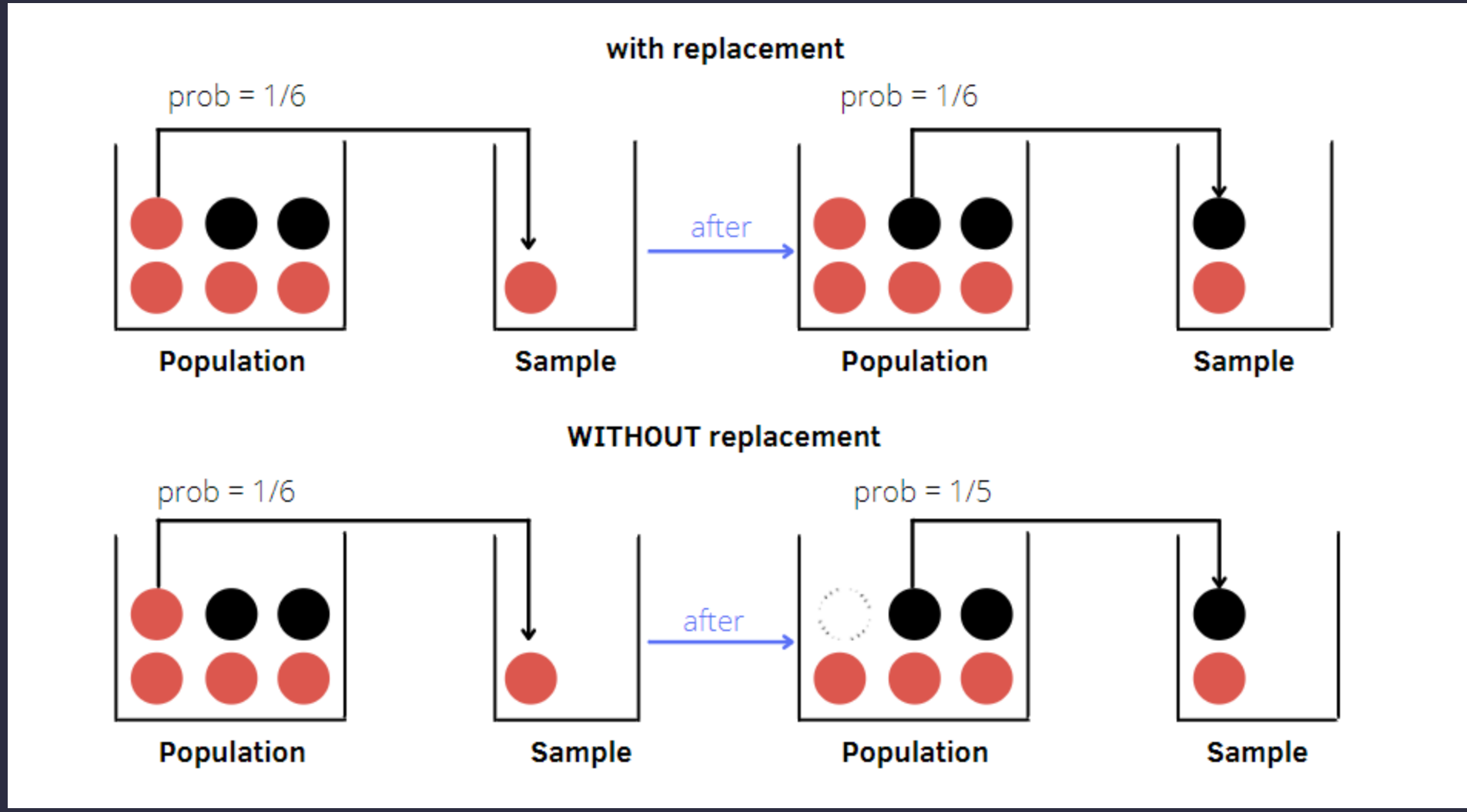
Sampling

- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data
- Sampling may not reduce database I/Os (page at a time).

Sampling

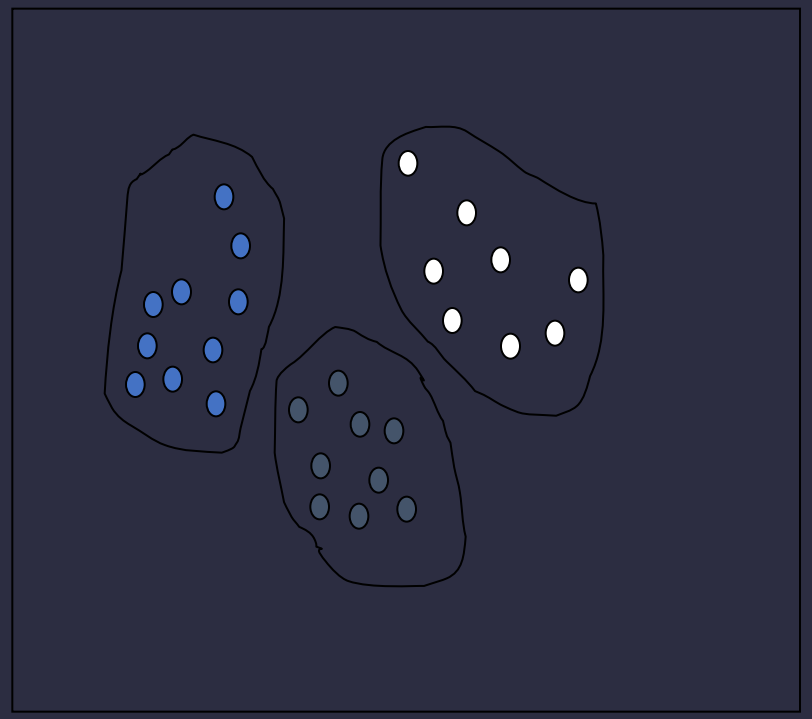

SRSWOR
(simple random sample without replacement)
SRSWR
Raw Data

Sampling

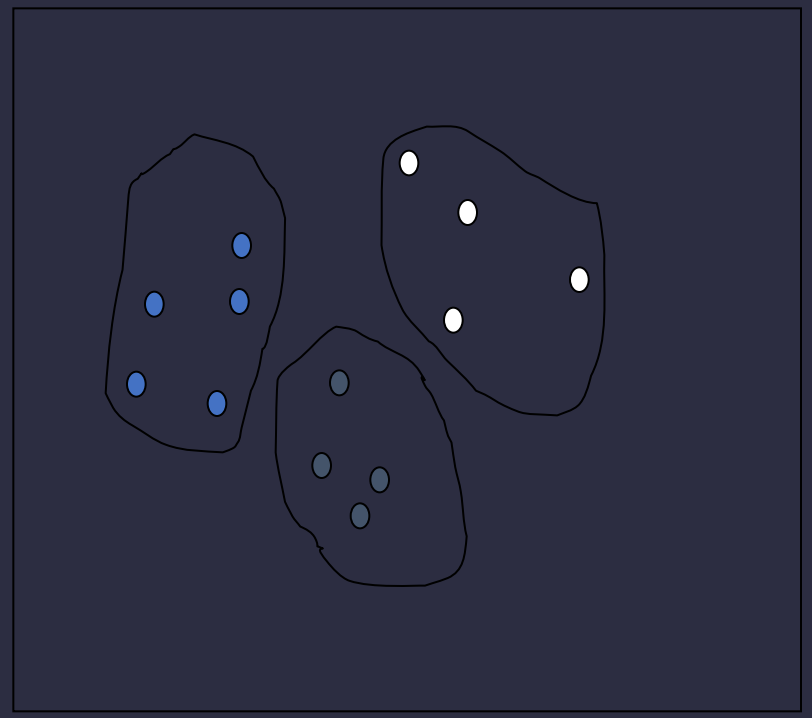

with replacement
prob = 1/6
prob = 1/6
after
Population
Sample
Population
Sample
WITHOUT replacement
prob = 1/6
prob = 1/5
after
Population
Sample
Population
Sample

Sampling

Raw Data



Cluster/Stratified Sample



Weka Tool

- What is Weka?
- Why use Weka (and why not)
- Weka versions and downloading the software
- Weka ARFF file format
- Starting Weka and the 5 Weka Interfaces

What is Weka?

- Weka is a data mining suite developed at University of Waikato
- Weka stands for Waikato Environment for Knowledge Analysis
- Weka includes everything necessary to generate and apply data mining models
  - Covers all major data mining tasks
  - Includes tools to preprocess and visualize data

WEKA is also a bird

Why Weka?

- There are many options on what data mining suite or toolkit one can use
- Weka has the following advantages:
  - Easy to learn
  - Free and open-source
  - Easy to download and runs on many platforms
  - Does not require programming knowledge
    - This is important since a few students may not have much programming experience

Weka Disadvantages

- Weka is not as flexible as other tools that are programming based
  - If you are programming anyway you can combine capabilities more flexibly and write code to do things not supported by Weka
  - Tools based on Python or R have large ecosystems

WEKA: the software

- Main features:
  - Comprehensive set of data pre-processing tools, learning algorithms and evaluation methods
  - Multiple interfaces that do not require programming
  - Experimenter can compare learning algorithms

Downloading Weka

- Weka runs on all major platforms
- It is free and easy to download
- Download Weka from here:
  - https://waikato.github.io/weka-wiki/downloading_weka/
- There is plenty of documentation
  - https://waikato.github.io/weka-wiki/documentation/

Weka ARFF File Format

- ARFF= Attribute Relation File Format
- Weka ARFF files include two main parts:
  - Specification of the features
  - The actual data
- Files are "flat files" usually comma separated
- Other tools use a separate file for each part
  - C4.5 decision tree tool, which was once very popular, had a "names" and "data" file
  - I find the single file format odd, since specification is short but actual data can be millions of lines

Weka ARFF Example

@relation heart-disease-simplified

This just defines dataset name

@attribute age numeric
@attribute sex { female, male}

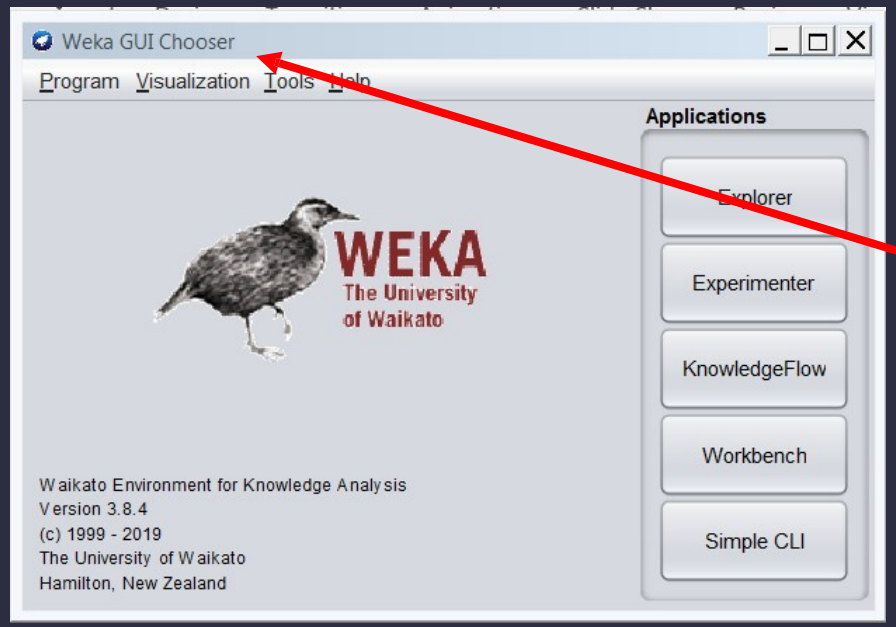
Categorical feature must list values

@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes}
@attribute class { present, not_present}

@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...

Starting Weka

- Run Weka (with console)
  - Different version if a new stable version released
  - May want to create a shortcut on your desktop



This is the GUI Chooser Window

The Weka Interfaces

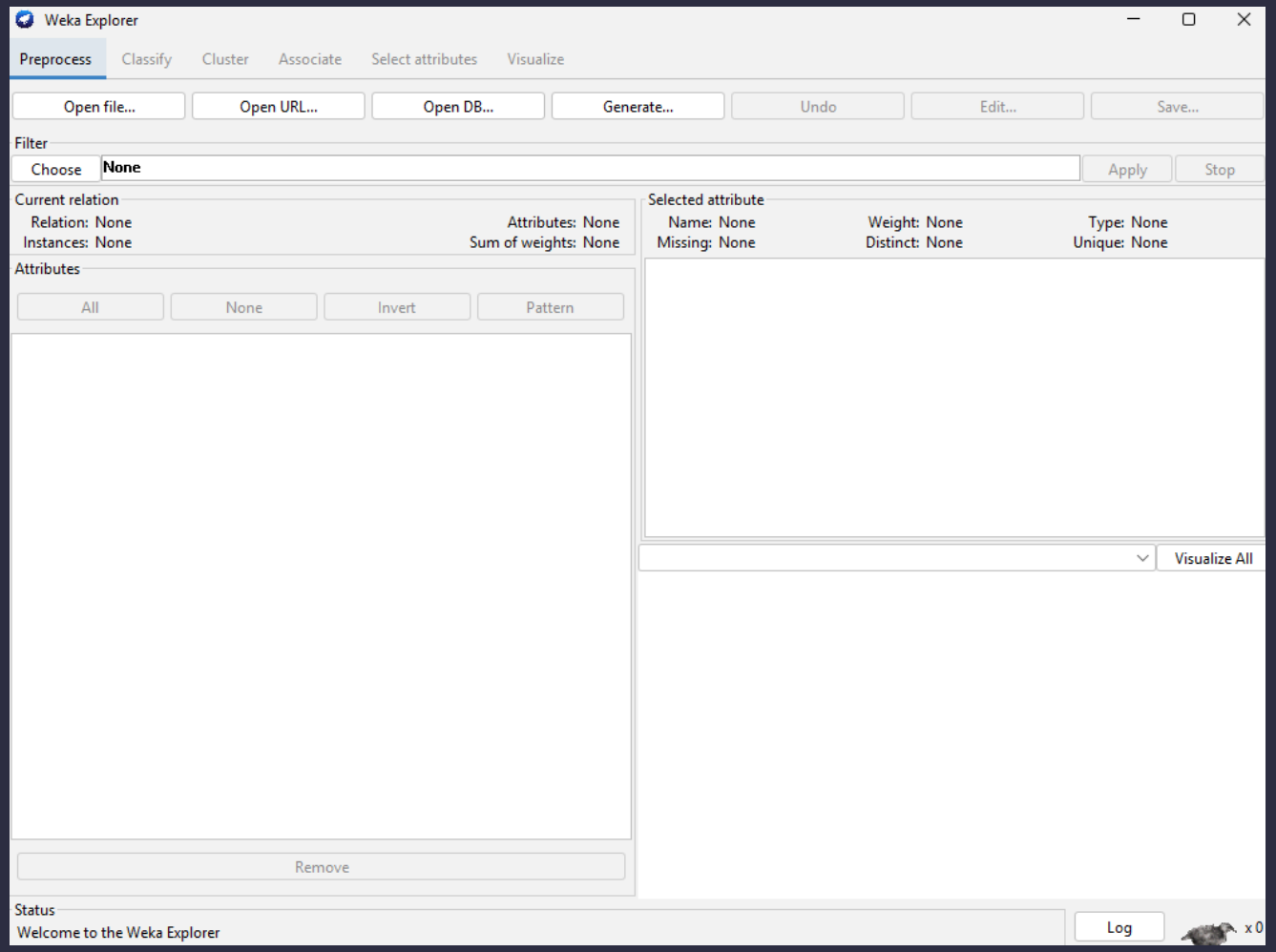
- Explorer:
  - We cover in detail
  - Enables you to apply multiple actions but does not explicitly model them
- Experimenter
  - Run many experiments in controlled manner and compare results
- KnowledgeFlow:
  - Like Explorer but each step is represented as a node in a graph, so flow explicitly represented
- Workbench
  - Combines all GUI interfaces into one
- Simple CLI (command line interface)
  - No GUI so can use shell scripts to control experiments
  - Similar to using Python where each command is a function

Preprocessing

- How to import data
- Visualizing feature value distribution and relationships to class values
- Preprocessing filters
  - Example using “Discretize” filter
- It is strongly recommended that you follow along on your own installation of Weka

Start the Explorer

Click on "Explorer " Button in the GUI Chooser Window

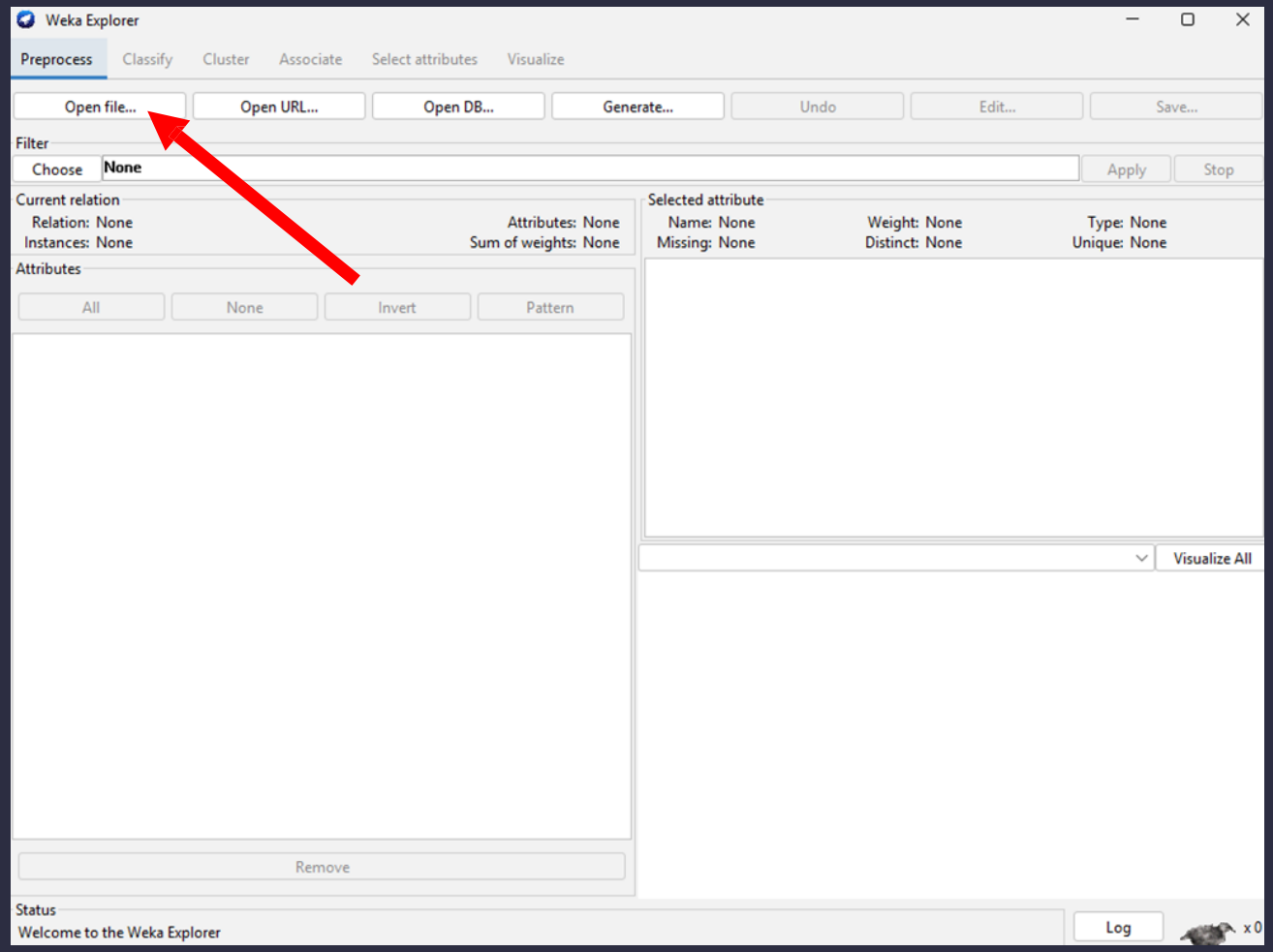


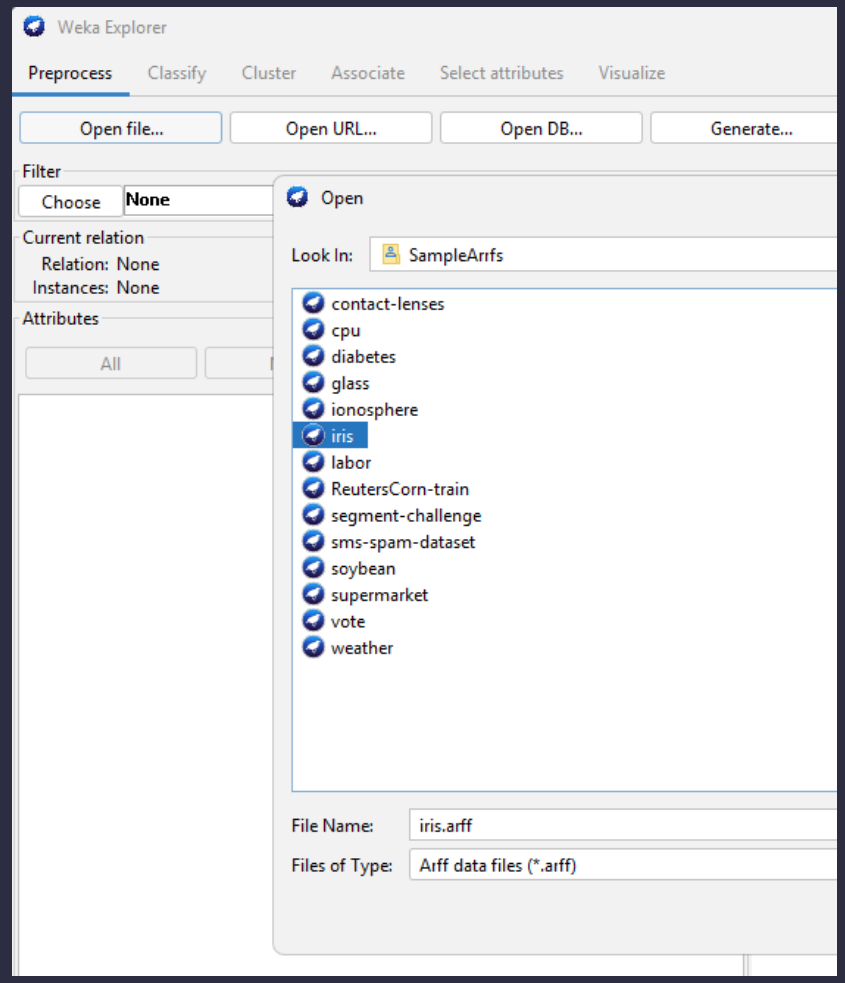
Importing Data

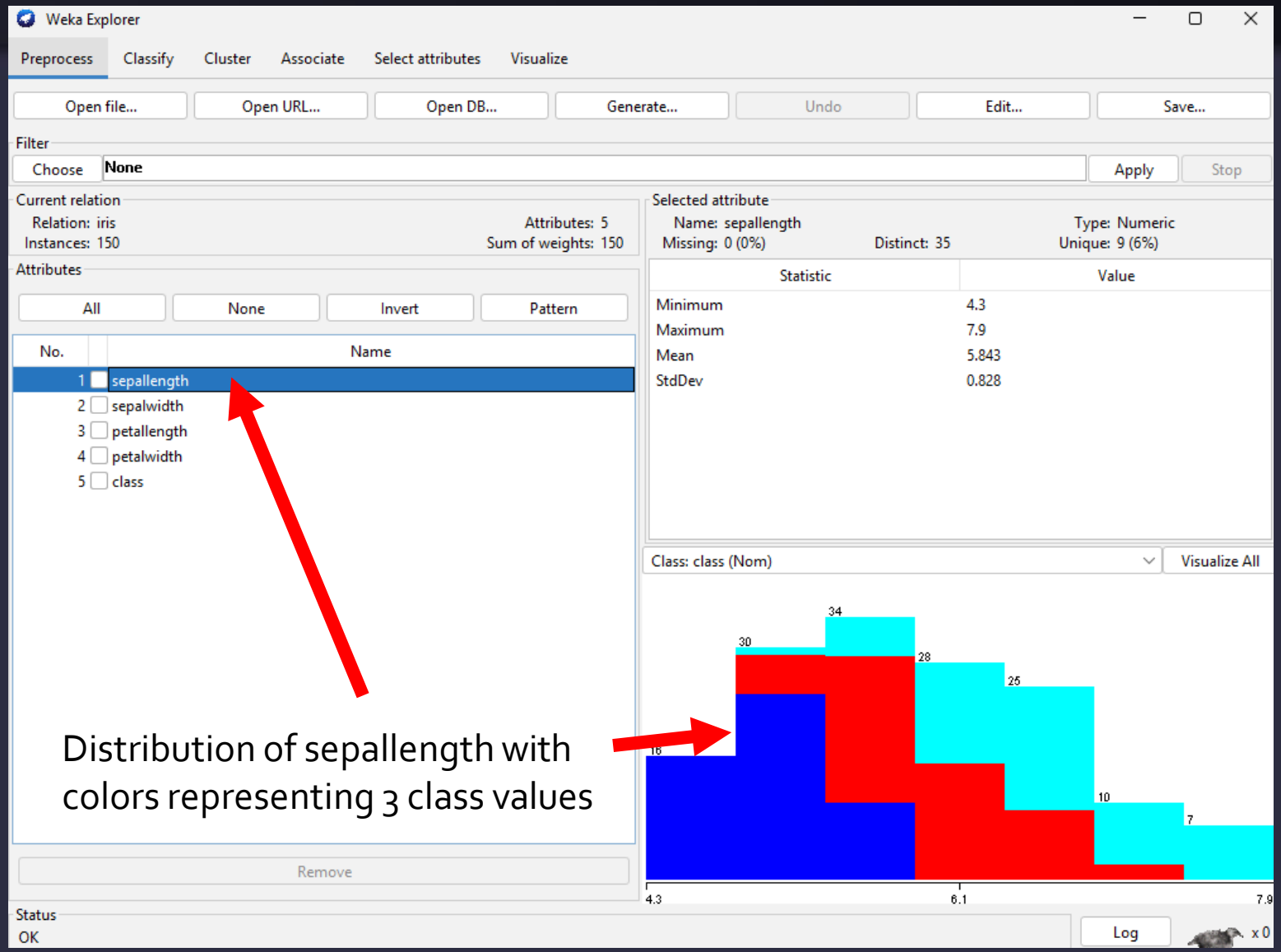
- Data can be imported from various formats:
  - ARFF, CSV, C4.5
  - You will most often import data in csv if not already prepared for WEKA
- Weka steps
  - Make sure Explorer "PreProcess" tab is selected
  - Click "Open File…" or "Open URL…"
  - Set the extension appropriately (default is .arff) and navigate to the file and select it
- If first line has feature names they are used
  - If no feature names then generic names are generated
  - Add feature names if missing or models won't be interpretable

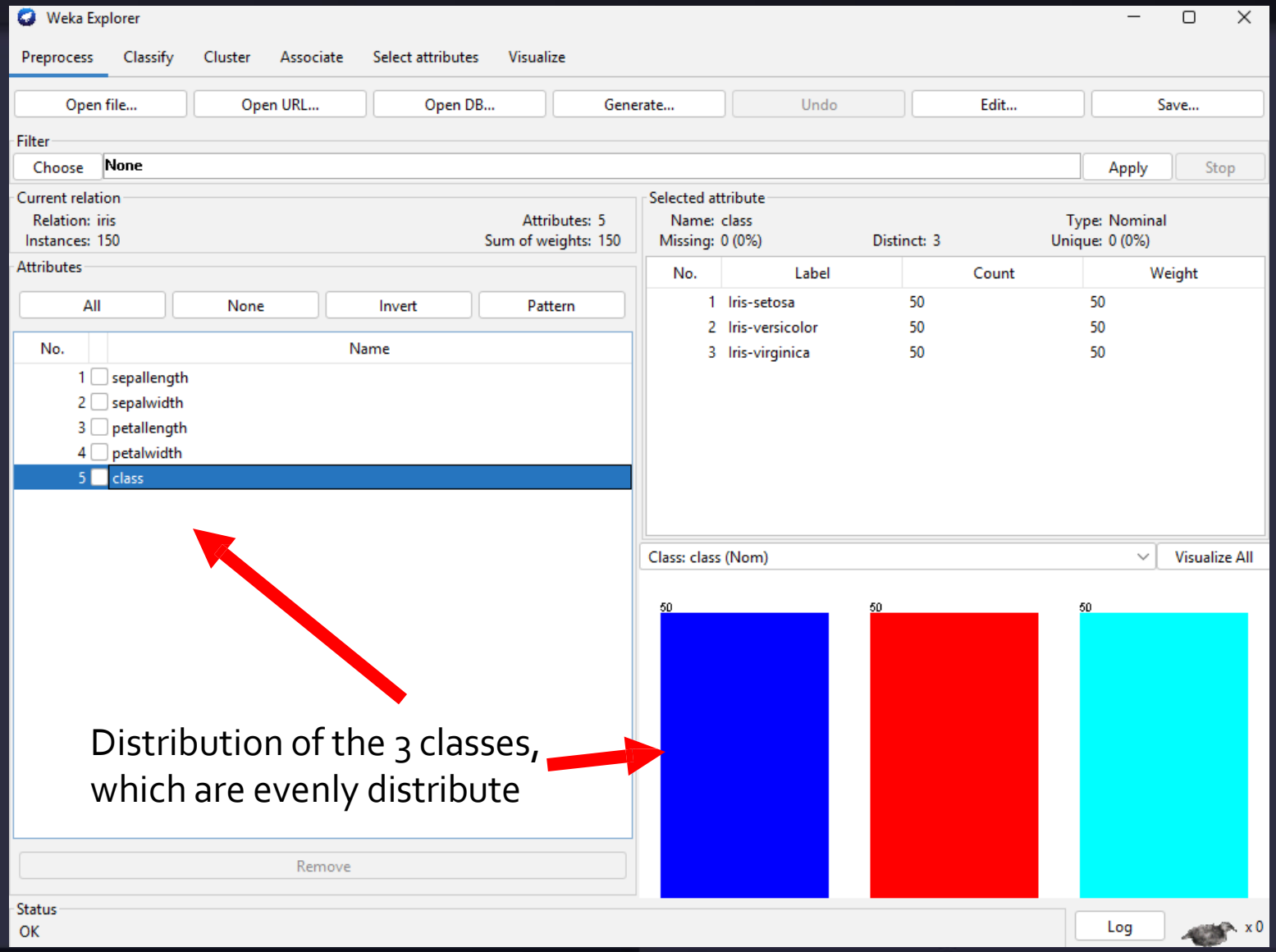
Reading in the Iris Dataset

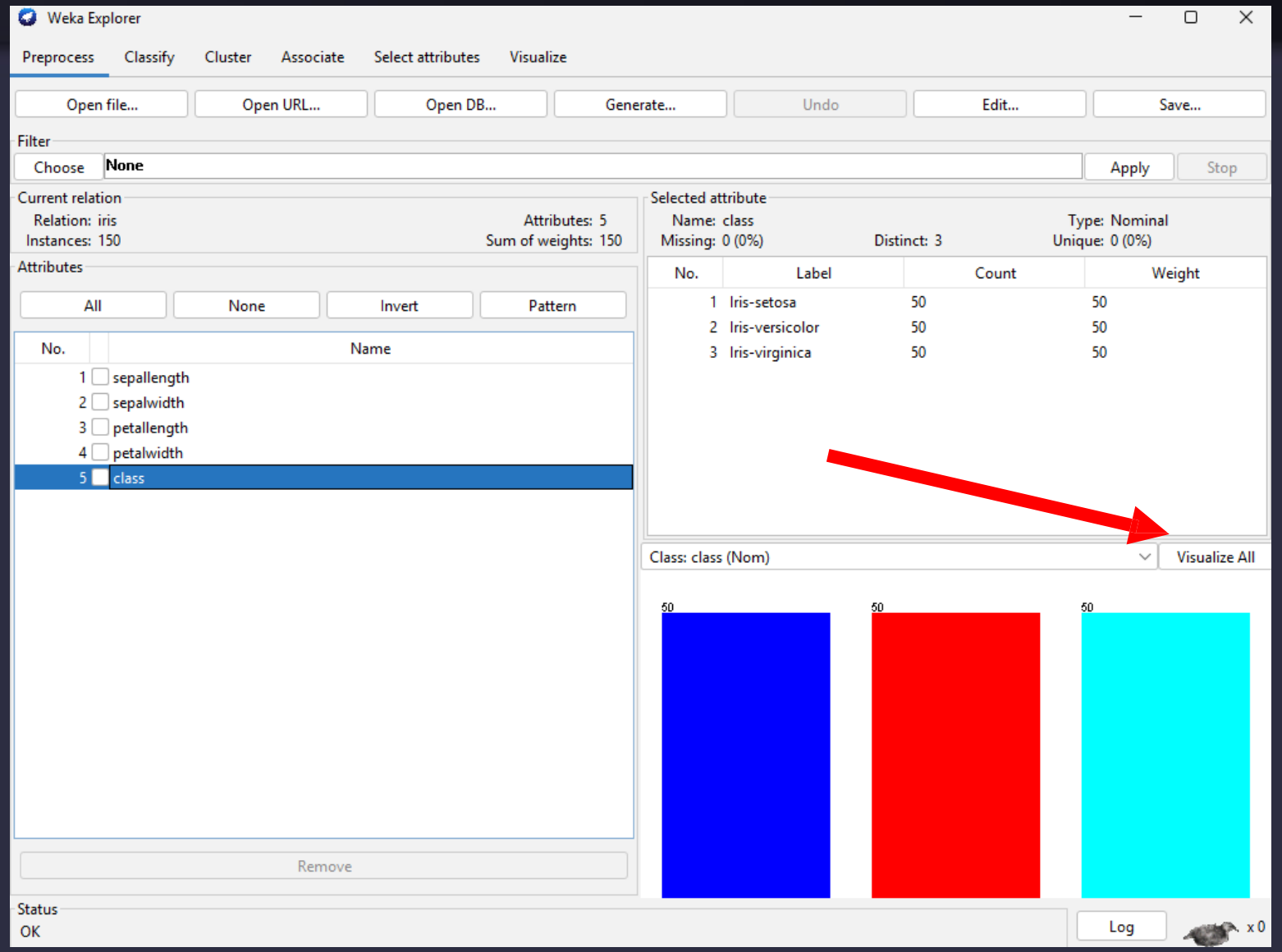
- We start with the well-known Iris data set
  - Download Iris Dataset
  - http://www.levent.tc/files/courses/big_data/SampleArrfs.rar
- Open it using "Open File" or "Open URL"
- Weka immediately shows stats about the features and how they are distributed

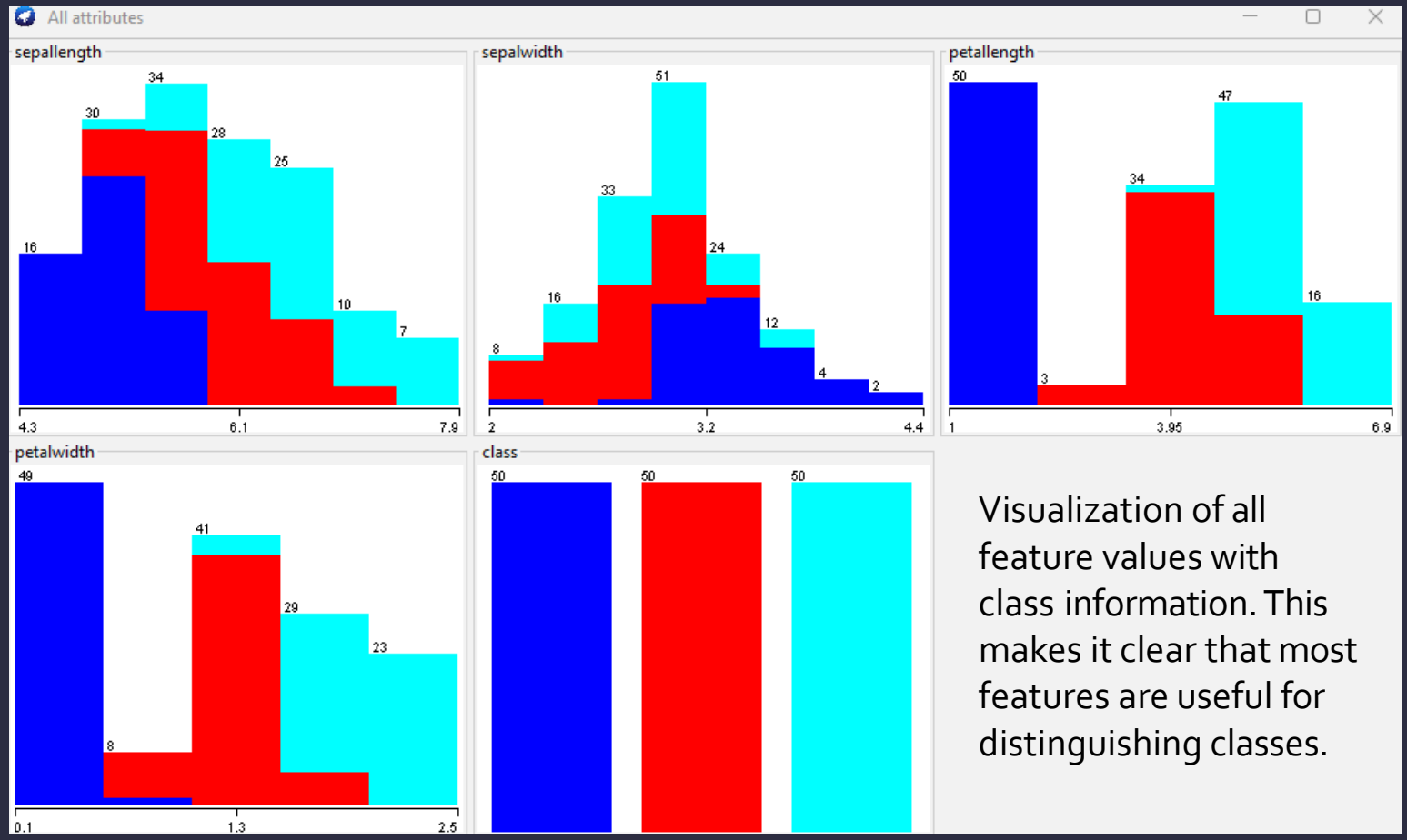

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: None
Instances: None
Attributes: None
Sum of weights: None
Selected attribute
Name: None
Missing: None
Weight: None
Distinct: None
Type: None
Unique: None
Attributes
All
None
Invert
Pattern
Visualize All
Remove
Status
Welcome to the Weka Explorer
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Filter
Choose
None
Current relation
Relation: None
Instances: None
Attributes
All
Open
Look In:
SampleArrfs
contact-lenses
cpu
diabetes
glass
ionosphere
iris
labor
ReutersCorn-train
segment-challenge
sms-spam-dataset
soybean
supermarket
vote
weather
File Name:
iris.arff
Files of Type:
Arff data files (*.arff)


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: sepallength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Class: class (Nom)
Visualize All
34
30
28
25
16
10
7
4.3
6.1
7.9
Status
OK
Log
x 0
Distribution of sepallength with colors representing 3 class values


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: class
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No.
Label
Count
Weight
1 Iris-setosa 50 50
2 Iris-versicolor 50 50
3 Iris-virginica 50 50
Class: class (Nom)
Visualize All
50
50
50
Distribution of the 3 classes,
which are evenly distribute
Status
OK
Log
x 0

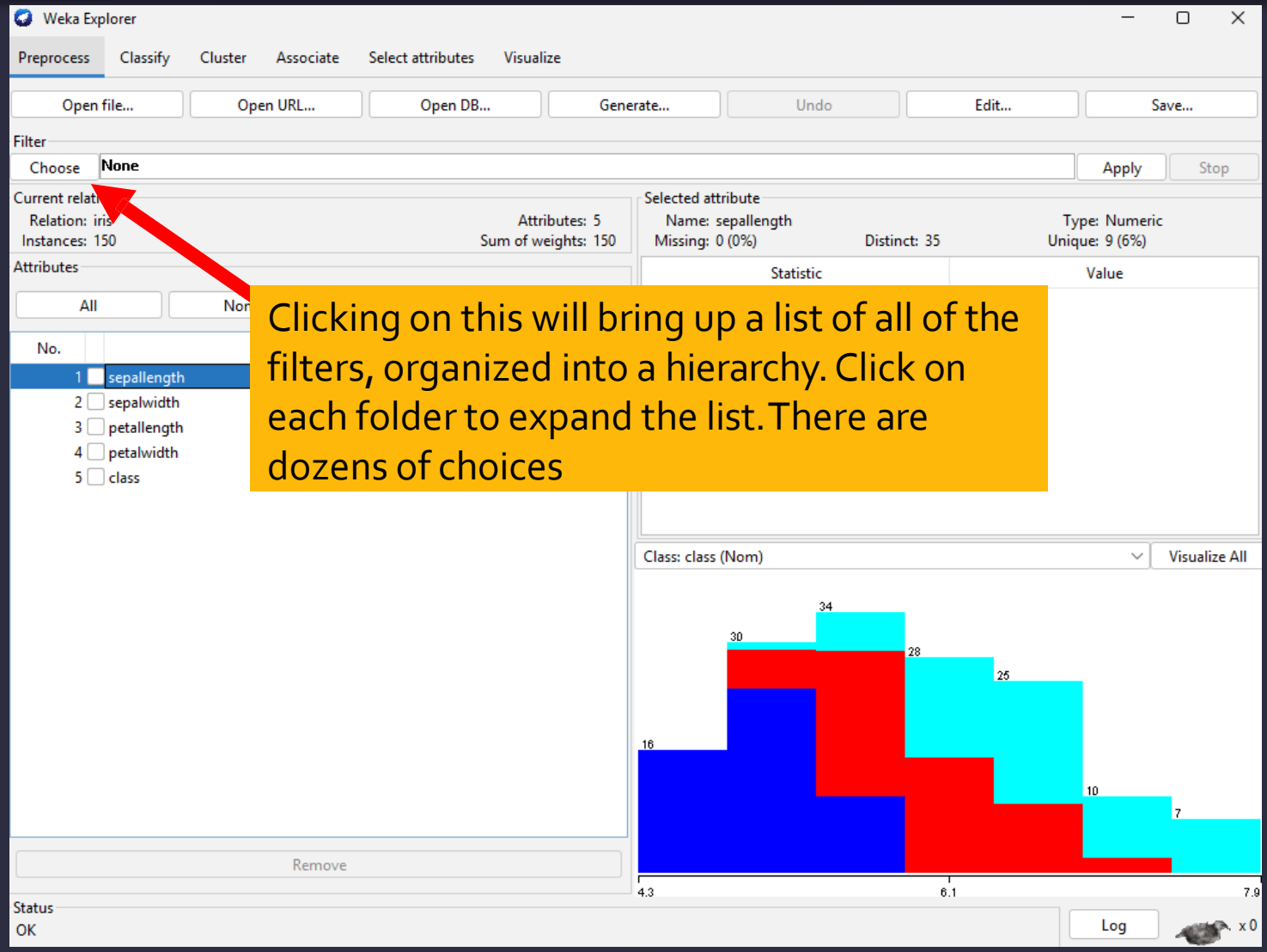

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Selected attribute
Name: class
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No.
Label
Count
Weight
1 Iris-setosa 50 50
2 Iris-versicolor 50 50
3 Iris-virginica 50 50
Class: class (Nom)
Visualize All
50
50
50
Status
OK
Log
x 0



Visualization of all feature values with class information. This makes it clear that most features are useful for distinguishing classes.

Preprocessing Filters

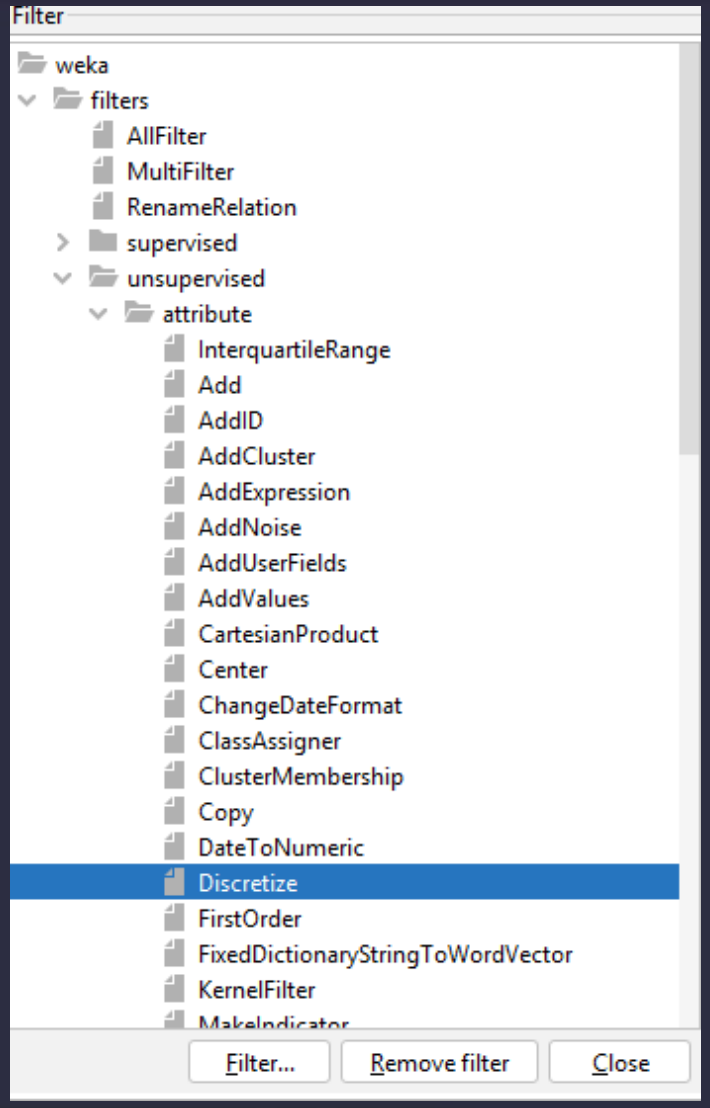
- Pre-processing tools in called “filters”
- Many preprocessing filters available:
  - Discretization
  - Normalization
  - Resampling
  - Feature selection
  - Feature transformation
  - Many more
- Know what available so can use it when needed
  - We will focus on Discretization

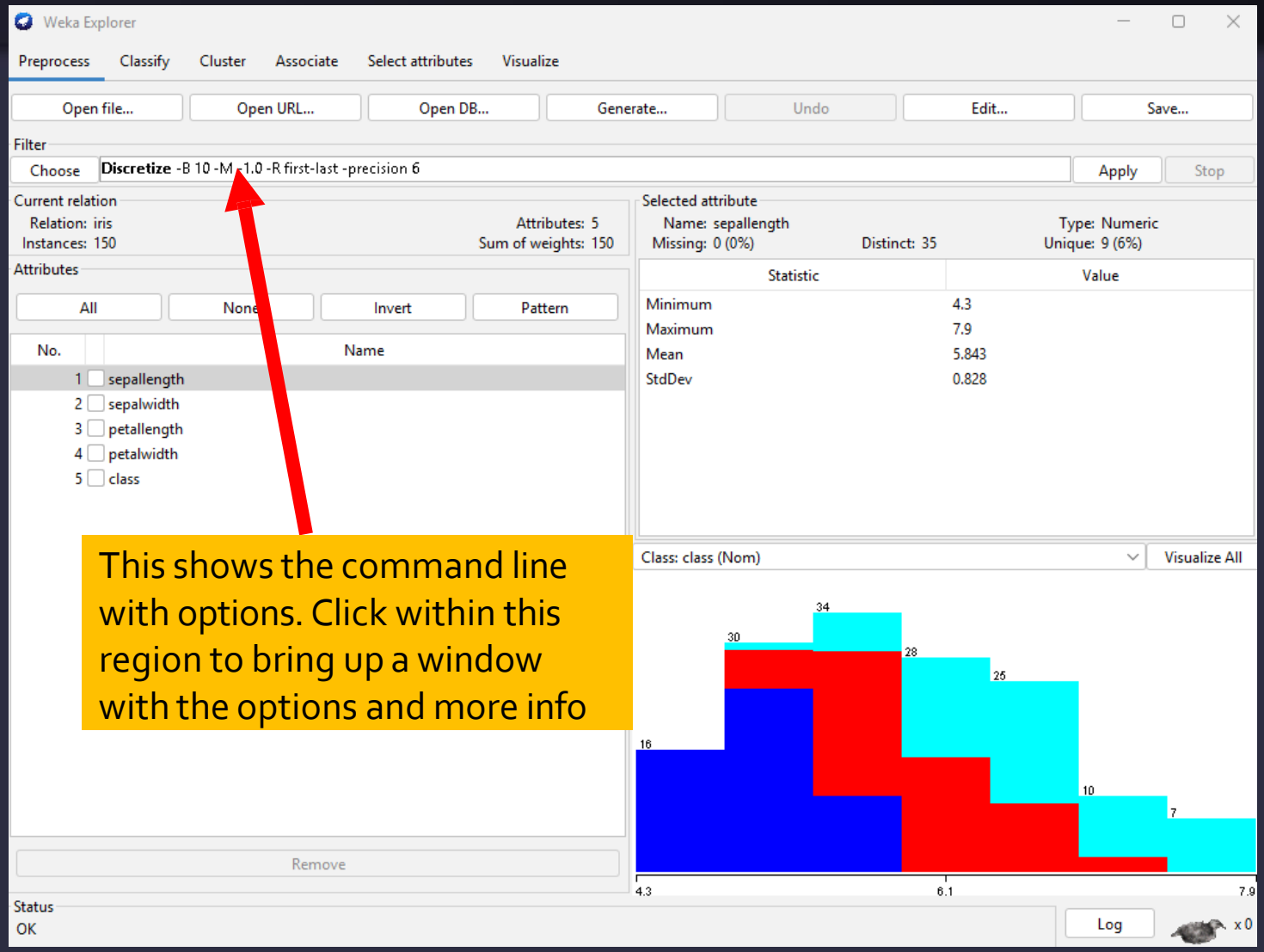


Clicking on this will bring up a list of all of the filters, organized into a hierarchy. Click on each folder to expand the list. There are dozens of choices

Discretization

- Discretization: numerical feature → categorical
  - In preprocess tab select "Choose", expand "unsupervised" then "attribute"
    - Select "Discretize" filter
    - Note discretize applies to attributes not instances
    - The following few slides are from an earlier version and may look just a little bit different

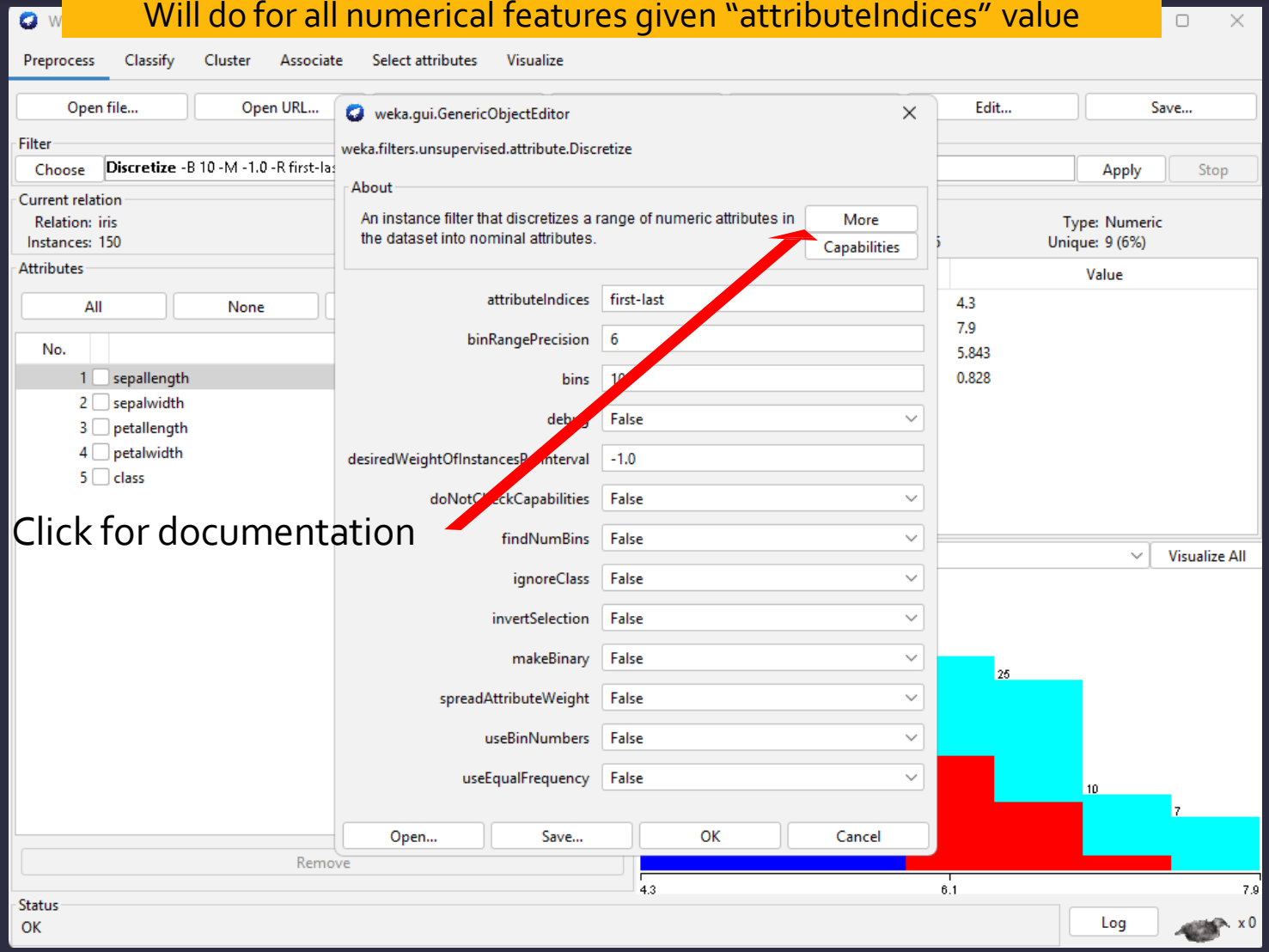

Filter
weka
filters
AllFilter
MultiFilter
RenameRelation
supervised
unsupervised
attribute
InterquartileRange
Add
AddID
AddCluster
AddExpression
AddNoise
AddUserFields
AddValues
CartesianProduct
Center
ChangeDateFormat
ClassAssigner
ClusterMembership
Copy
DateToNumeric
Discretize
FirstOrder
FixedDictionaryStringToWordVector
KernelFilter
Filter...
Remove filter
Close



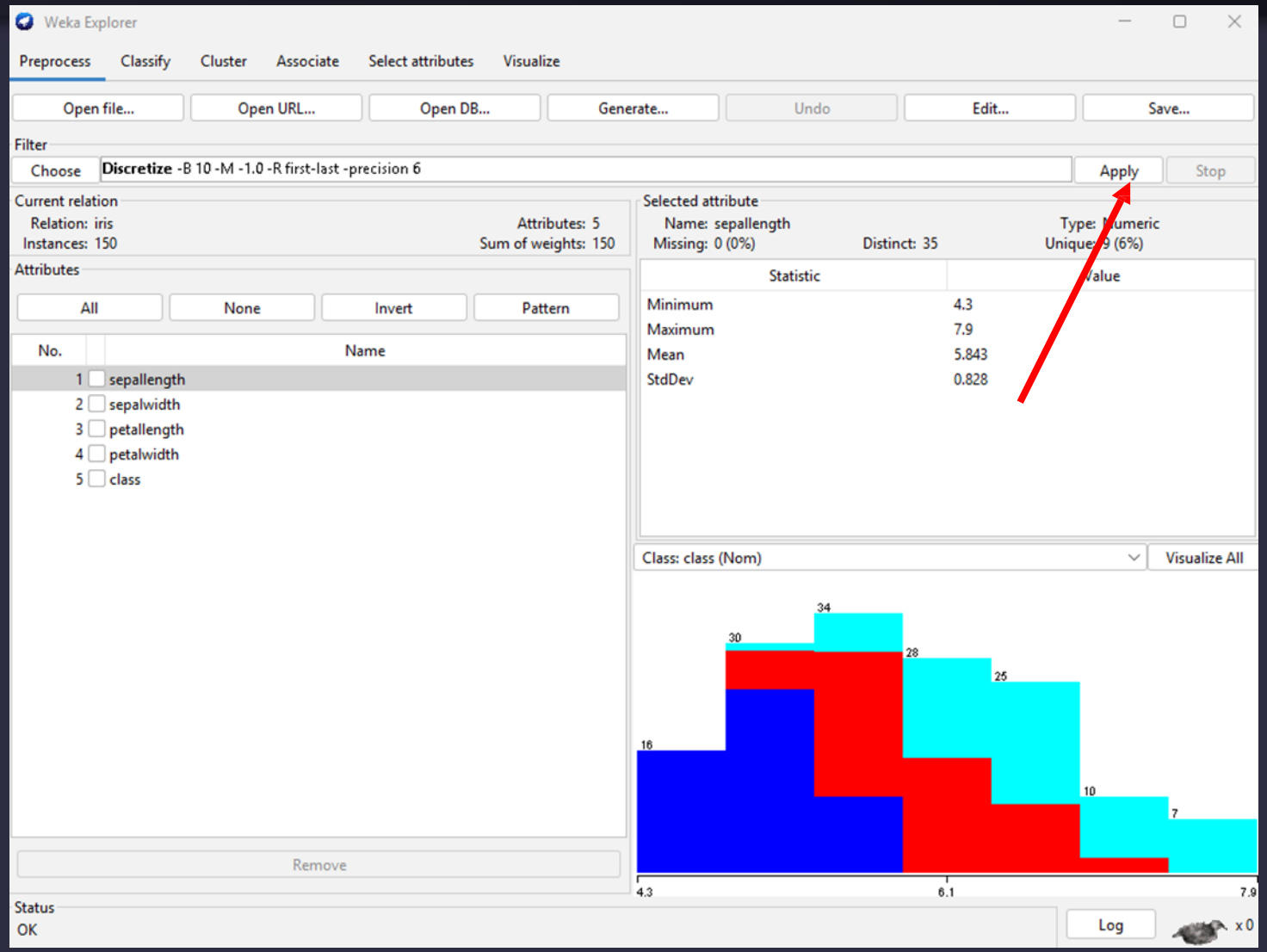
This shows the command line with options. Click within this region to bring up a window with the options and more info

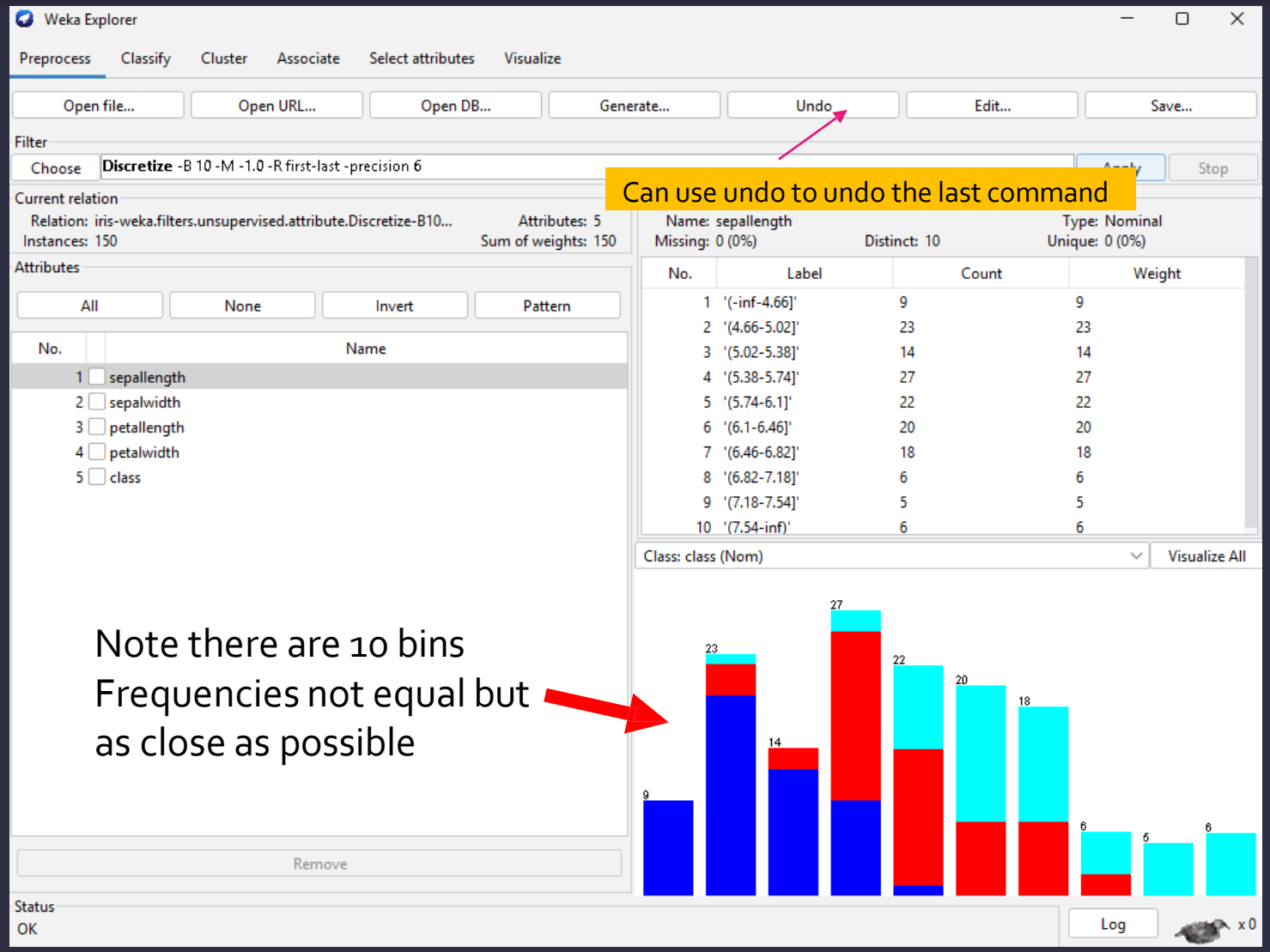
Create 10 equal frequency bins

Will do for all numerical features given "attributeIndices" value



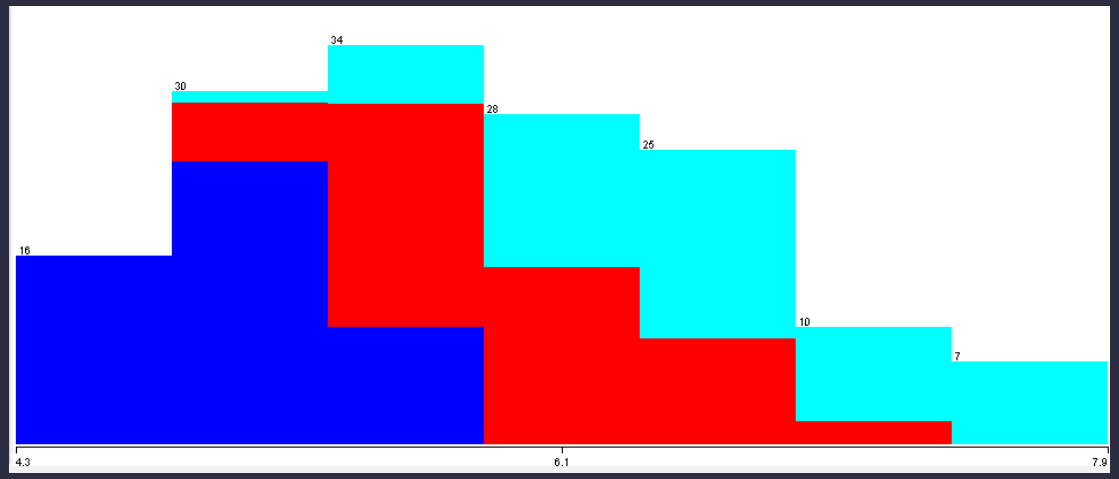
Click for documentation

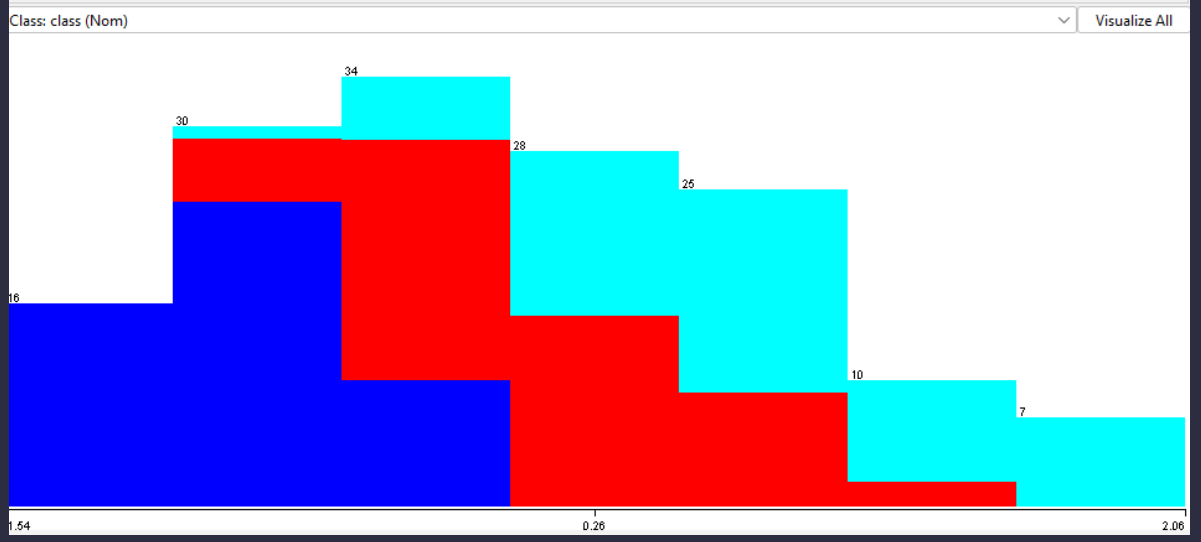

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
Discretize -B 10 -M -1.0 -R first-last -precision 6
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Selected attribute
Name: sepallength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Class: class (Nom)
Visualize All
34
30
28
25
16
10
7
4.3
6.1
7.9
Remove
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
Discretize -B 10 -M -1.0 -R first-last -precision 6
Apply
Stop
Can use undo to undo the last command
Current relation
Relation: iris-weka.filters.unsupervised.attribute.Discretize-B10...
Attributes: 5
Instances: 150
Sum of weights: 150
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Name: sepallength
Type: Nominal
Missing: 0 (0%)
Distinct: 10
Unique: 0 (0%)
No.
Label
Count
Weight
1 '(-inf-4.66]' 9 9
2 '(4.66-5.02]' 23 23
3 '(5.02-5.38]' 14 14
4 '(5.38-5.74]' 27 27
5 '(5.74-6.1]' 22 22
6 '(6.1-6.46]' 20 20
7 '(6.46-6.82]' 18 18
8 '(6.82-7.18]' 6 6
9 '(7.18-7.54]' 5 5
10 '(7.54-inf)' 6 6
Class: class (Nom)
Visualize All
9
23
14
27
22
20
18
6
5
6
Note there are 10 bins
Frequencies not equal but
as close as possible
Status
OK
Log
x 0

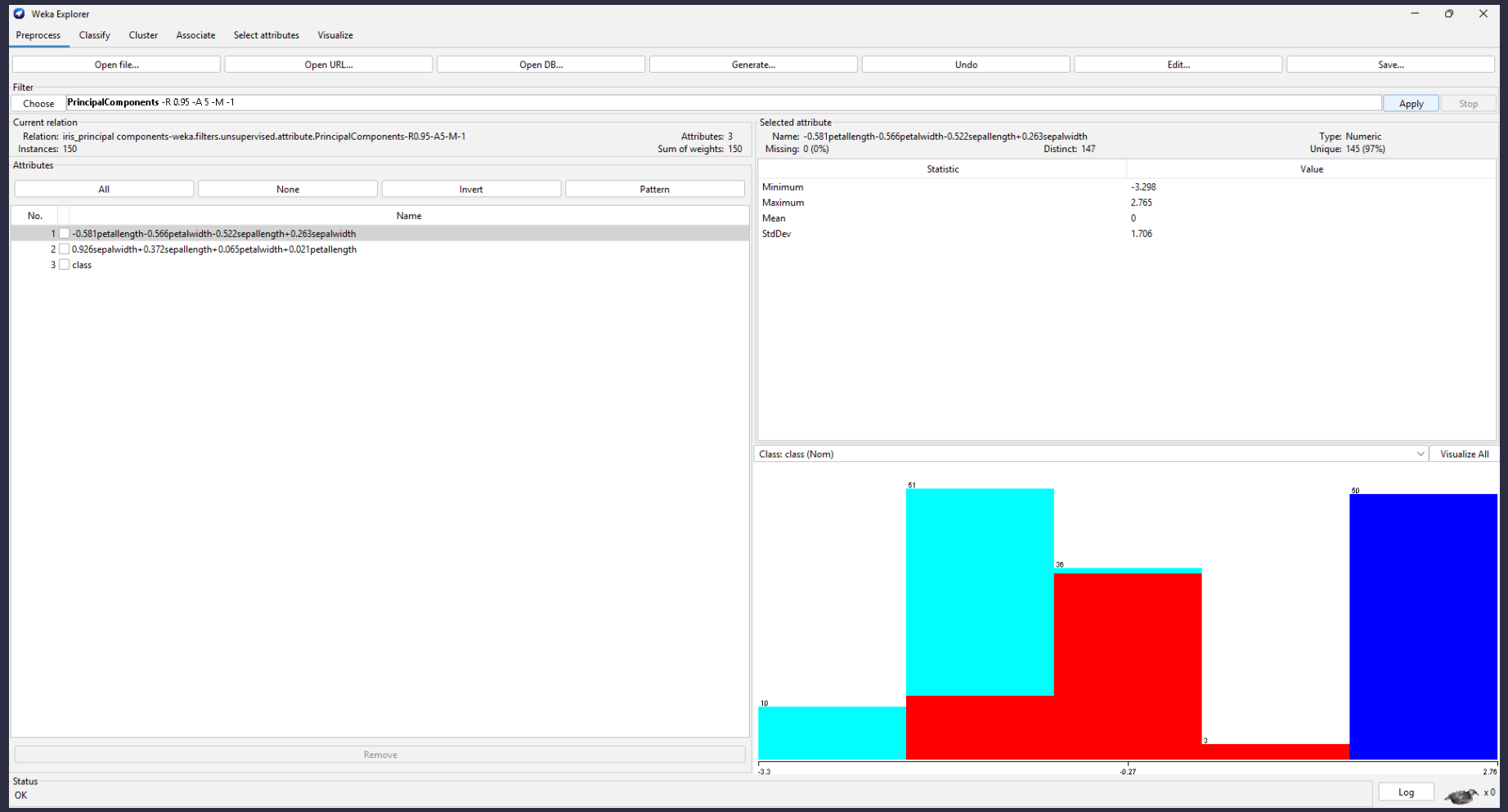
Center (Normalize) (Standardize) – Transform numeric attributes to have zero mean (or into a given numeric range) (or to have zero mean and unit variance)

Statistic	Value
Minimum	-1.543
Maximum	2.057
Mean	-0
StdDev	0.828





PrincipalComponents – Perform a principal components analysis/transformation of the data

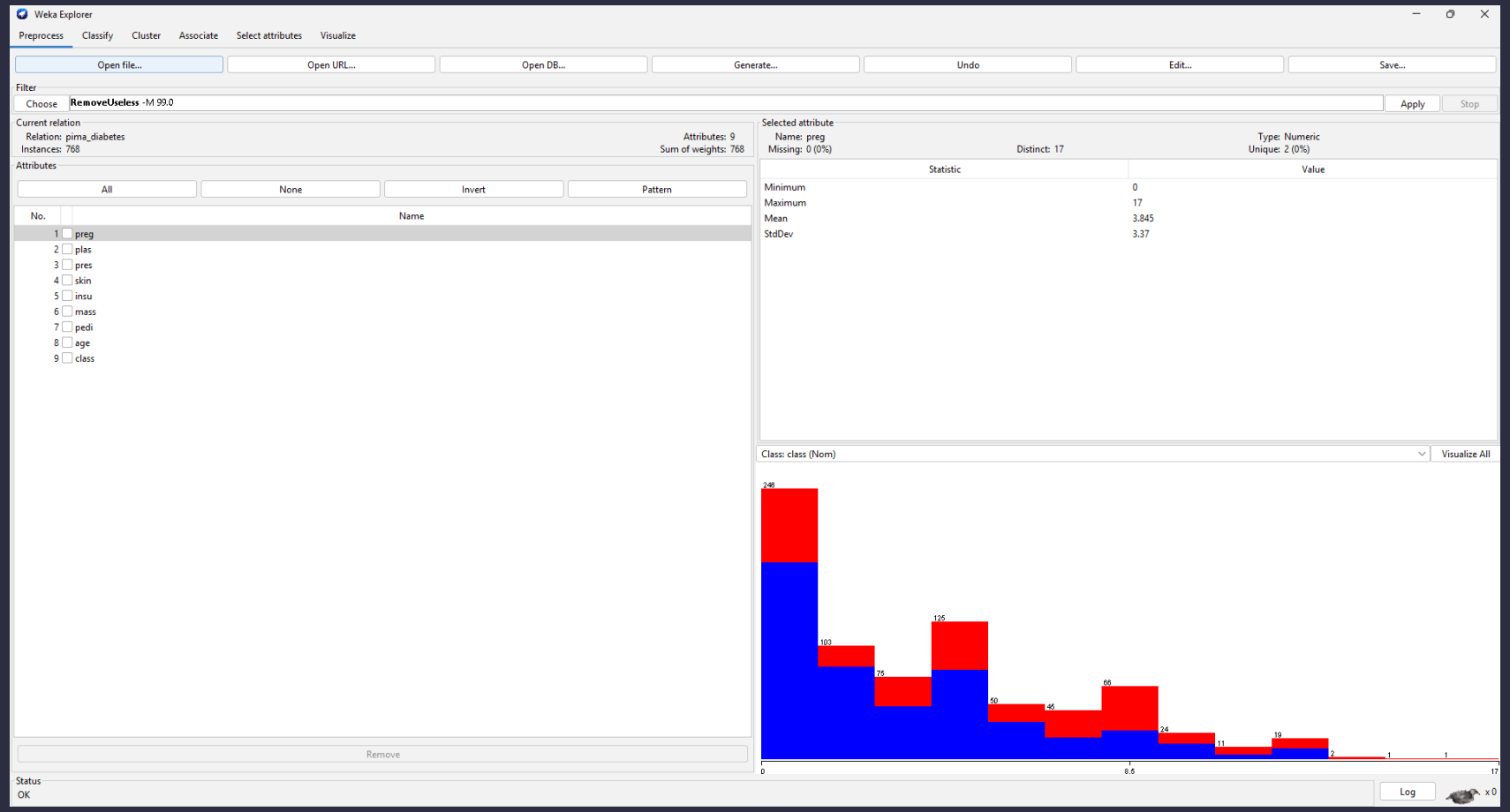


Handle Missing Values

- Data is rarely clean and often you can have corrupt or missing values.
- It is important to identify, mark and handle missing data when developing machine learning models in order to get the very best performance.
- How to handle missing values in your machine learning data using Weka.
  - How to mark missing values in your dataset.
  - How to remove data with missing values from your dataset.
  - How to impute missing values.

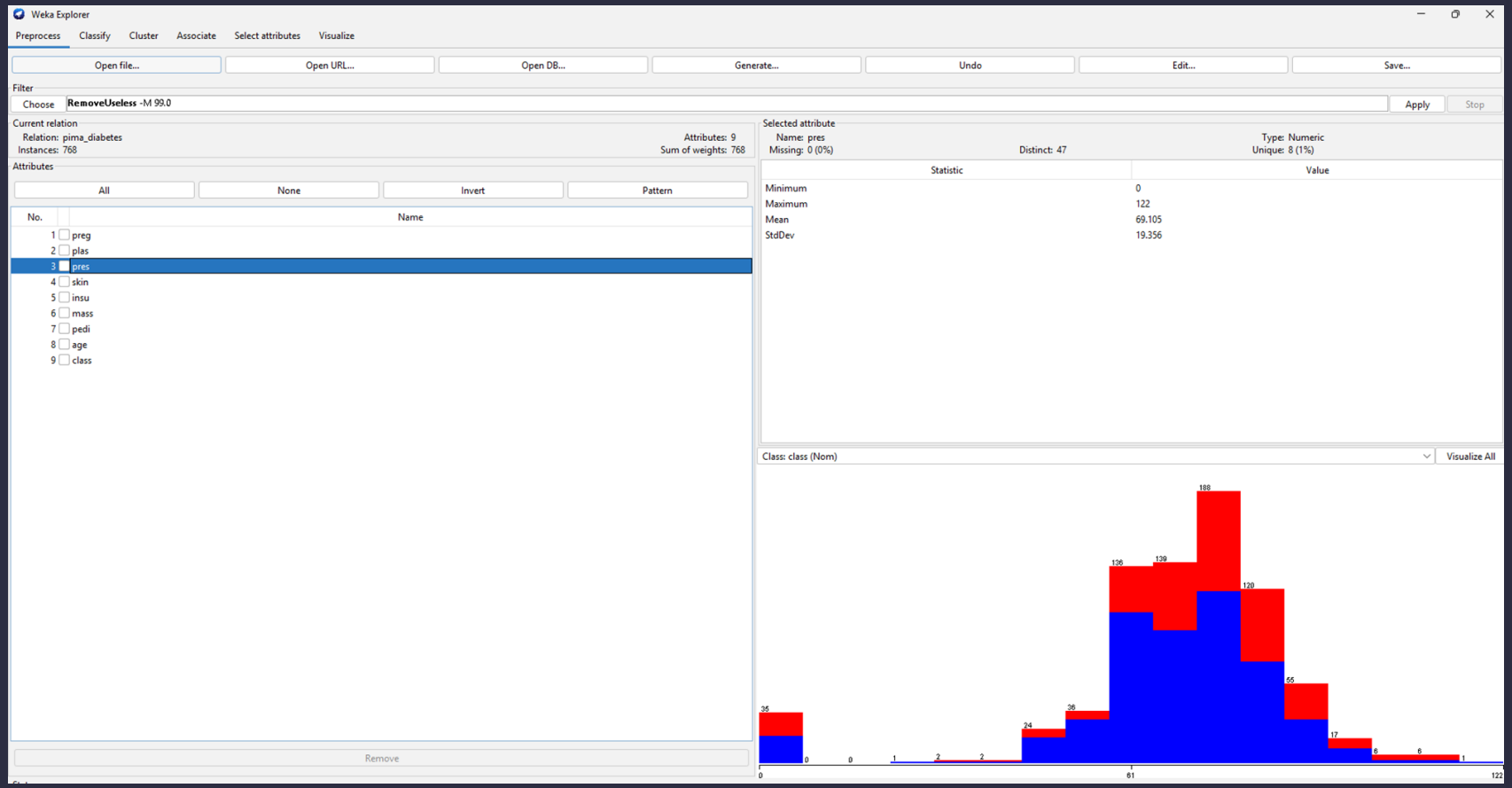
Handle Missing Values

Load Diabetes arff to Weka



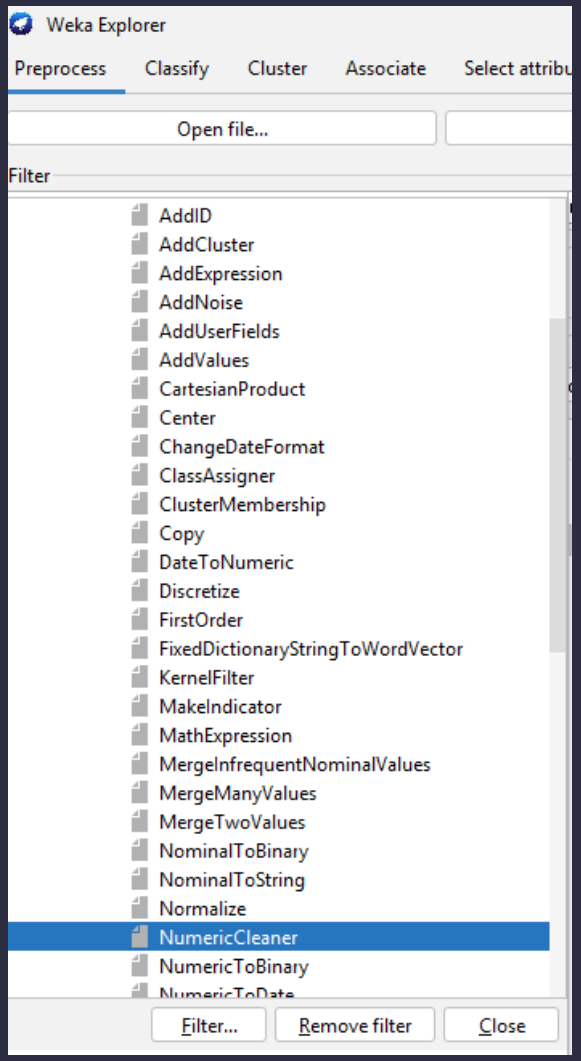
Handle Missing Values

Some attributes such as blood pressure (pres) and Body Mass Index (mass) have values of zero, which are impossible. These are examples of corrupt or missing data that must be marked manually.



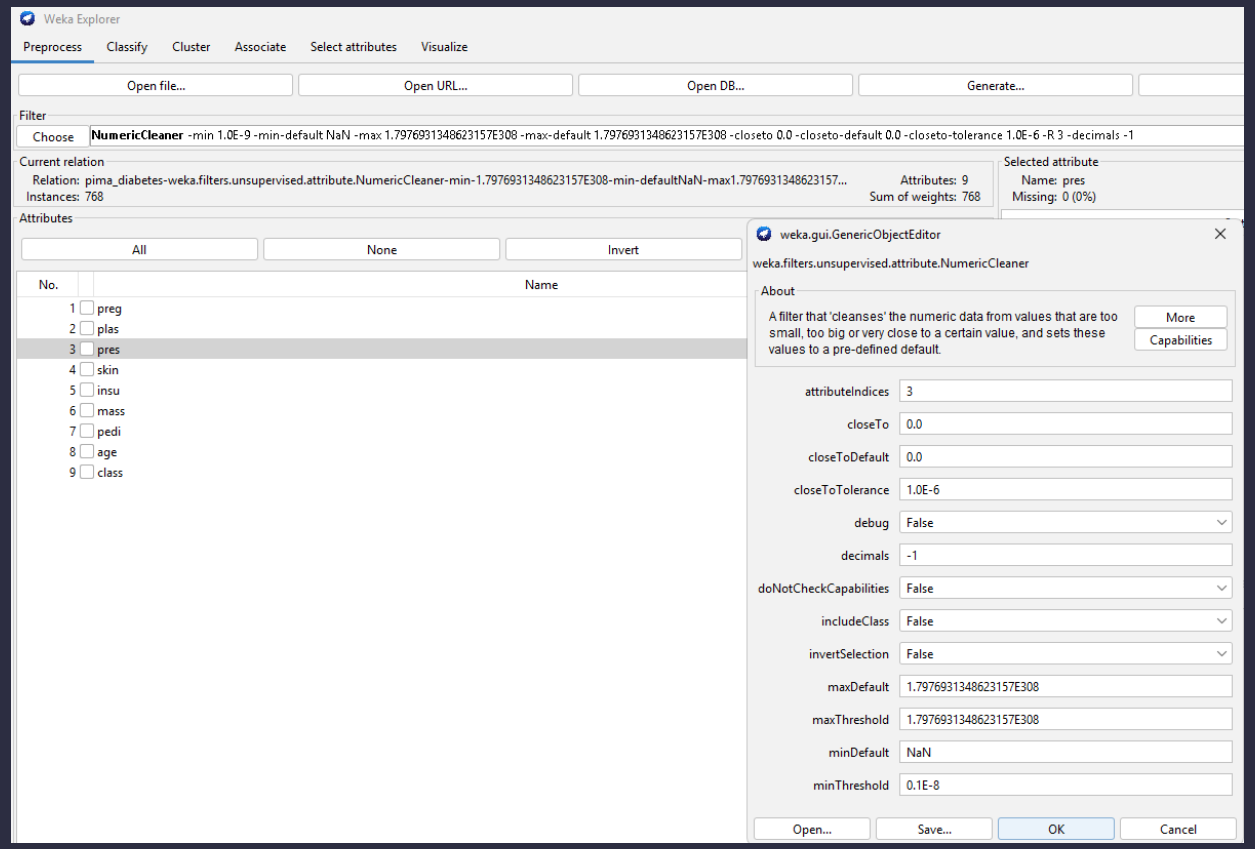
Handle Missing Values

Mark missing values in Weka using the NumericalCleaner filter.



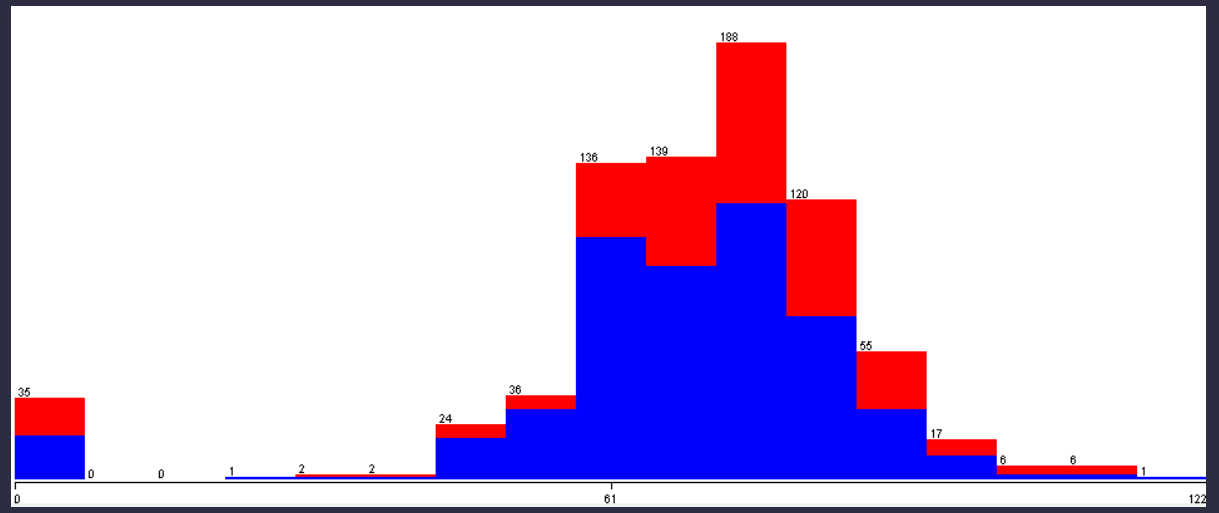
Handle Missing Values

- Set the attributeIndicies to 3, the index of the Pressure
- Set minThreshold to 0.1E-8 (close to zero), which is the minimum value allowed for the attribute.
- Set minDefault to NaN, which is unknown and will replace values below the threshold

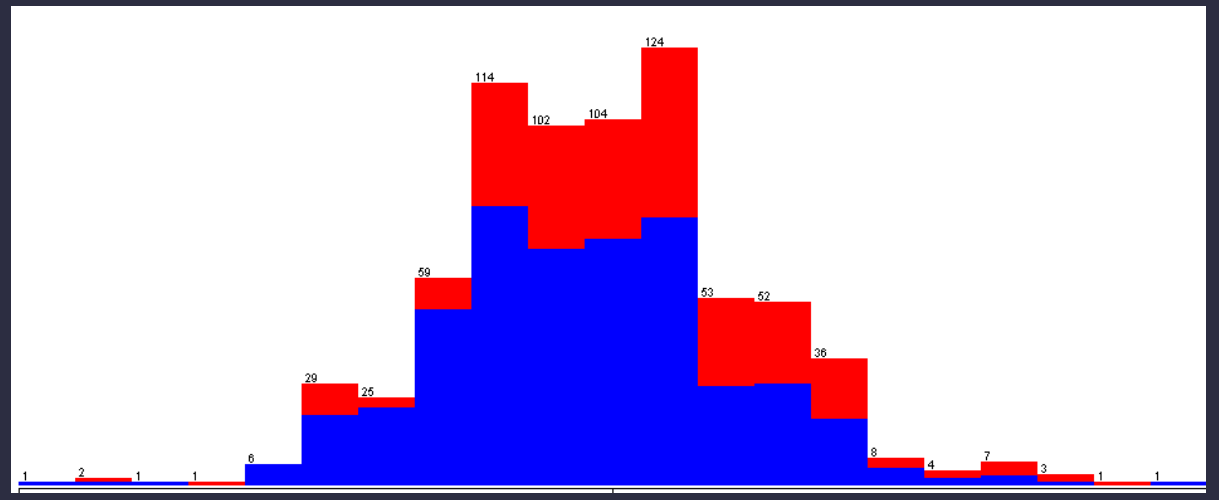


Handle Missing Values

Before



After

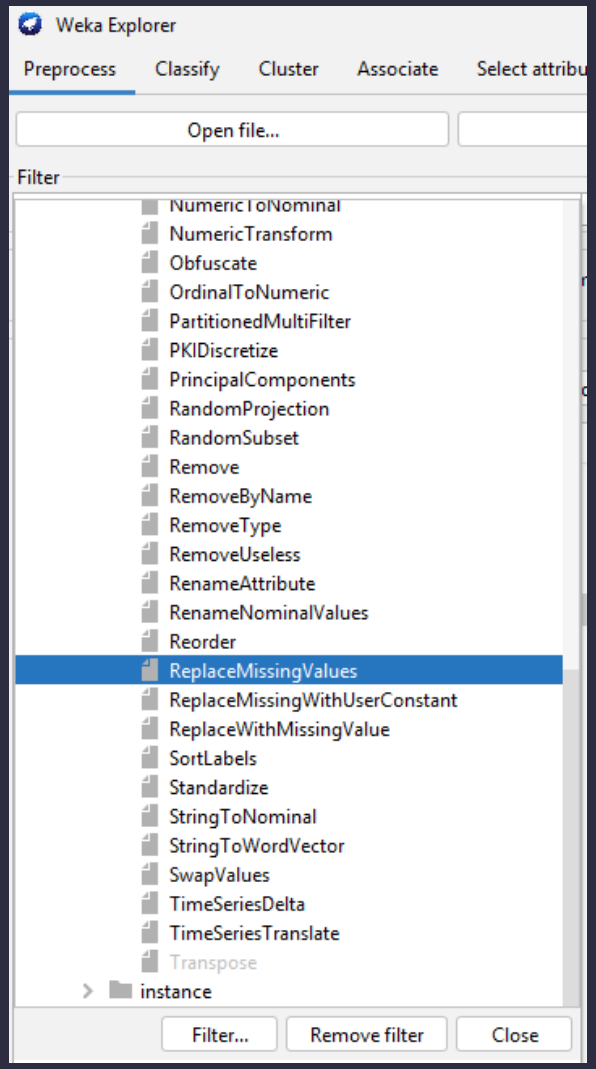


Impute Missing Values

It is common to impute missing values with the mean of the numerical distribution. You can do this easily in Weka using the ReplaceMissingValues filter.

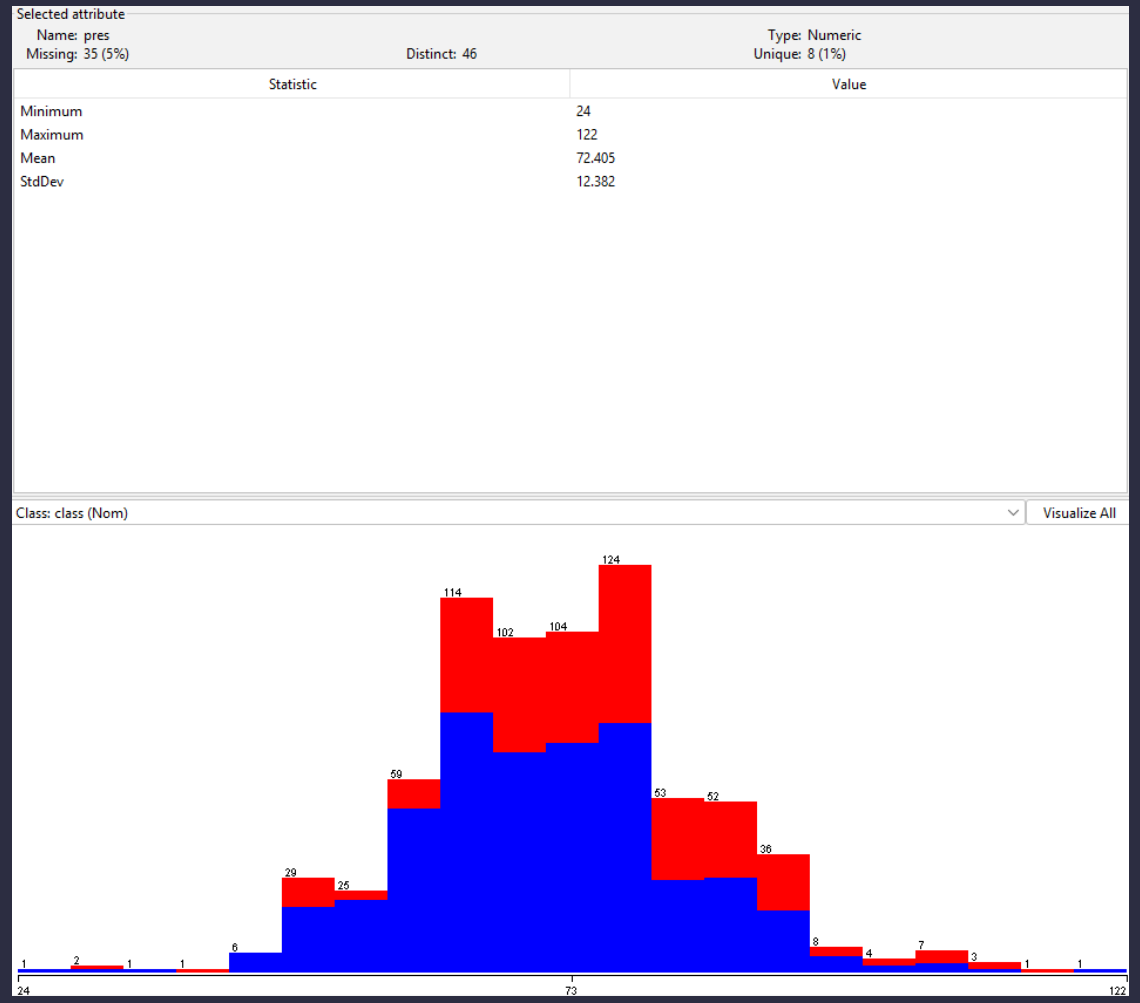
Click the “Choose” button for the Filter and select ReplaceMissingValues, it us under unsupervized.attribute.ReplaceMissingValues.

Missings will set to the mean value of the distribution.



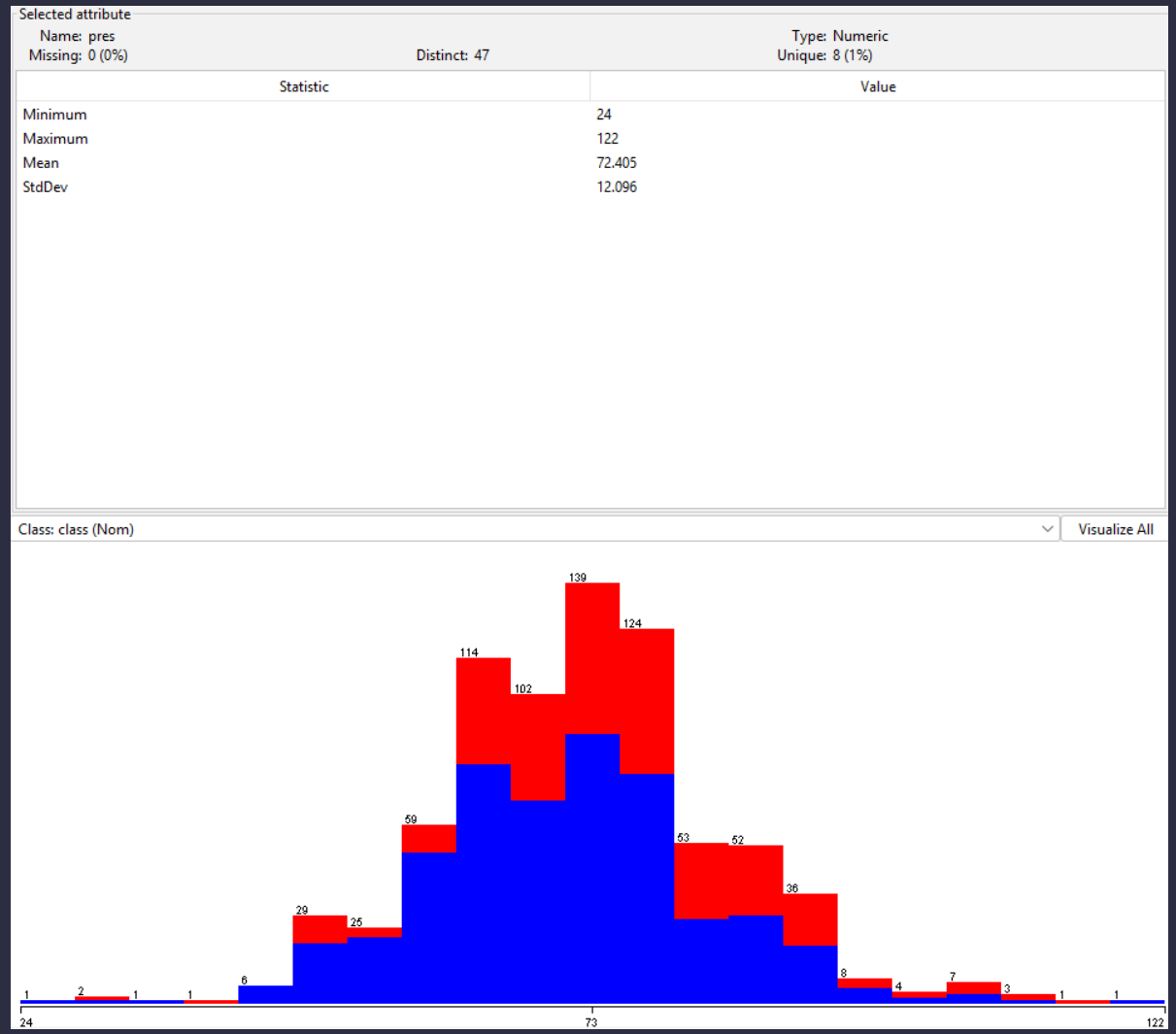
Impute Missing Values

Before



Impute Missing Values

After



Thesis Topic

- Review and apply all Weka Tutorial
- Research Filters
- Find a Dataset, better related with your thesis topic (if you dont have, define)
- Apply preprocessing algorithms on it
  - Missing value
  - Components of nonconforming values
  - Use Weka and Develop a Python Script

Prepair a slide (5-6 Pages) and present next week, you will have 5 minutes

Thesis Topic

Data sources

- Google Dataset Search: https://datasetsearch.research.google.com/
- OpenML: https://www.openml.org/
- Hugging Face: https://huggingface.co/datasets
- Sağlık Bakanlığı: https://acikveri.saglik.gov.tr/
- TUIK: https://data.tuik.gov.tr
- Harvard: https://dataverse.harvard.edu/

Thesis Topic

Why you might not use a dataset directly in Weka

- Format & Encoding
  - Not ARFF/CSV, wrong delimiter, missing header, mixed decimal symbols (, vs .), bad quoting/escaping, non-UTF-8.
- Schema & Target Issues
  - No class/label column, wrong target selected, multilabel structure (Weka expects a single nominal class unless using specific packages).
- Type Mismatch
  - Text left as string (should be converted to nominal or vectorized), dates not parsed as Date, high-cardinality categoricals treated as numeric (or vice-versa).

You might need to do some preprocessing with a programming language or manually