National Technical
University of Ukraine
"Igor Sikorsky
Kyiv Polytechnic Institute"

Institute of
Physics and
Technology

Intellectual Data Analysis

Practice 1: Data Visualization and Preprocessing

Dr. Nataliya K. Sakhnenko

Data Analysis Steps

Data Collection & Assembly

Data Preprocessing

Data Exploration & Visualization

Model Building

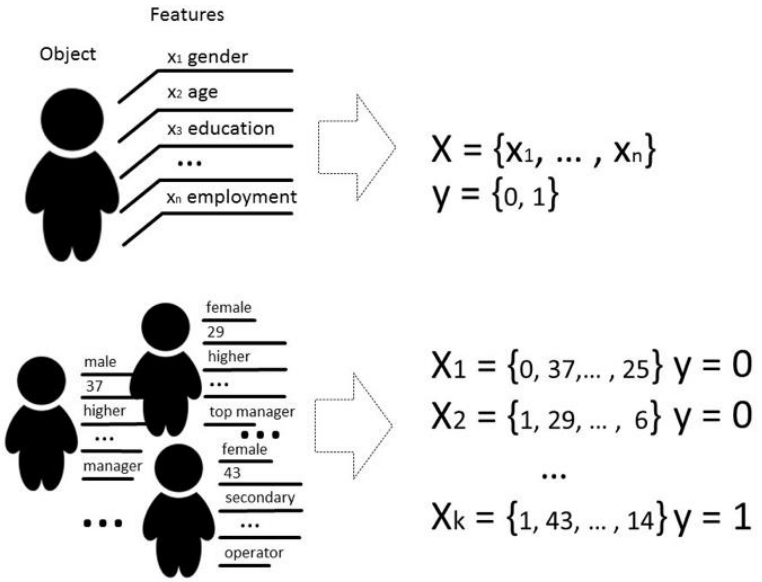
Model Evaluation

Recap (Lc1):Tabular data

Feature is an individual measurable property or characteristic of a phenomenon being observed.

- Let X be object
- f: X -> D
- feature vector $\left(f_1(x), f_2(x), \dots, f_n(x)\right)$
- $\begin{pmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots & \dots \\ f_1(x_m) & f_2(x_m) & \dots & f_n(x_m) \end{pmatrix}$ is feature set

- X_{mxn}, m is number of examples(objects), n is number of features

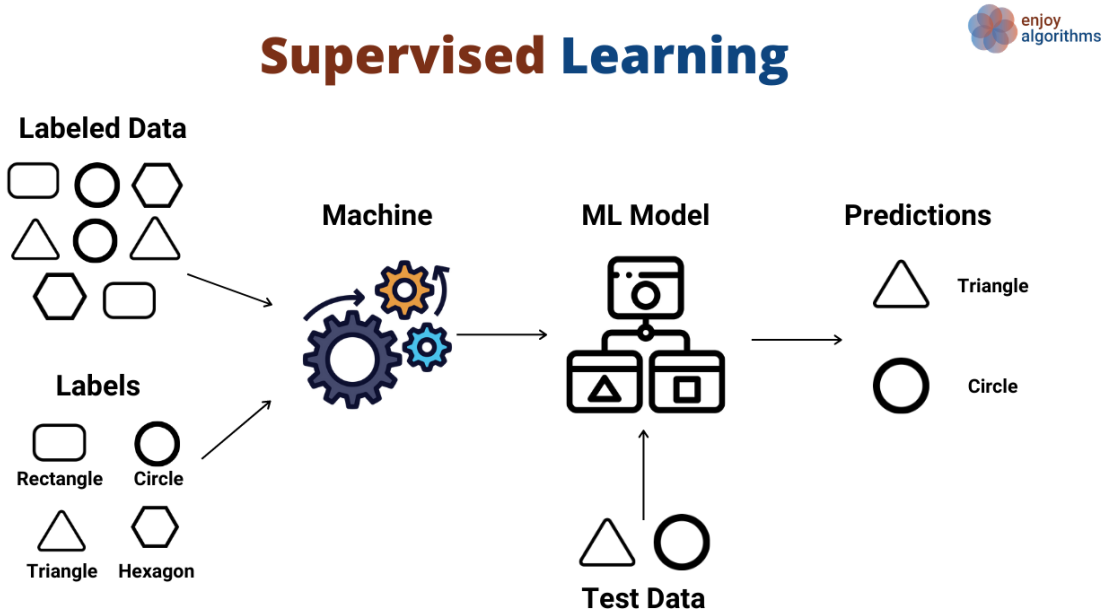


Tabular Data:

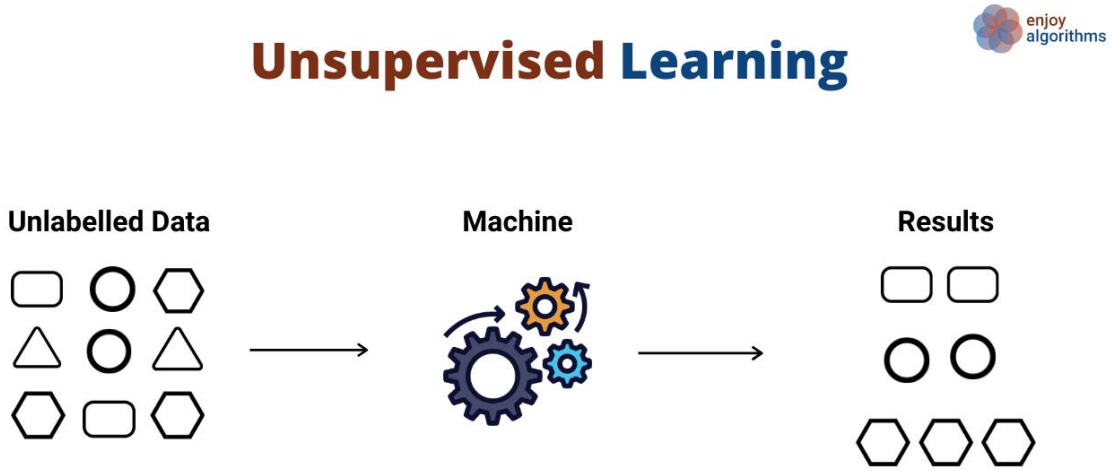
- Rows = samples / observations
- Columns = features / variables
- Each row represents one instance / data point
- Each column stores a specific attribute / feature

Recap (Lc1): Supervised and unsupervised learning

- In supervised learning the training data includes the desired solutions, called labels



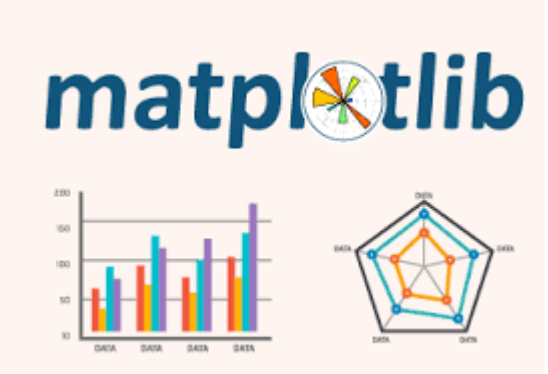
- In unsupervised learning the training data is unlabeled



https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning

Data Visualization and Preprocessing Tools

- **pandas** – data loading, cleaning, transformation
- **NumPy** – numerical computations, arrays & matrices
- **matplotlib** – basic plots (line, bar, scatter, histograms)
- **seaborn** – statistical plots, heatmaps, pairplots
- **scikit-learn (sklearn.preprocessing)** – scaling, normalization, encoding, missing values

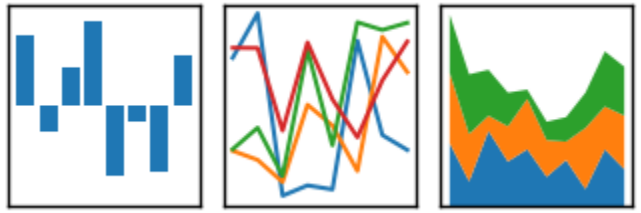


Pandas lib

pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



`pandas.read_csv()`

Read a comma-separated values (csv) file into DataFrame.

	id	iv2	rt
count	120.000000	120.00000	120.000000
mean	9.500000	2.00000	877.587425
std	5.790459	0.81992	309.293048
min	0.000000	1.00000	283.240752
25%	4.750000	1.00000	582.630955
50%	9.500000	2.00000	902.719888
75%	14.250000	3.00000	1114.050194
max	19.000000	3.00000	1472.688933

`pandas.DataFrame.head()`

Return the first *n* rows

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

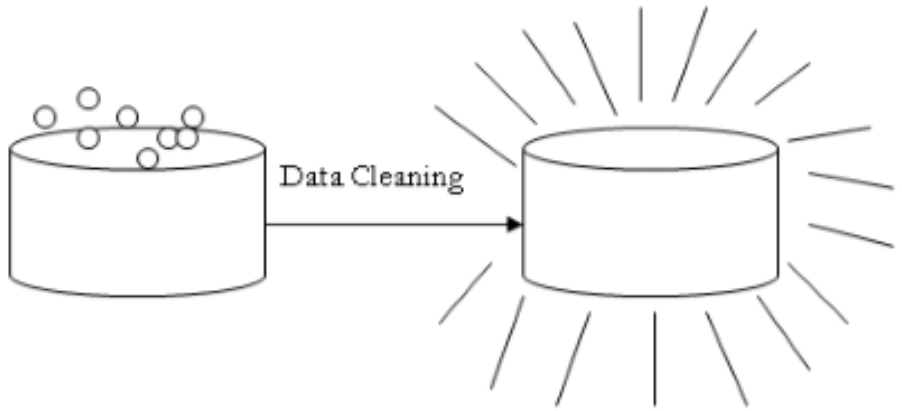
`pandas.DataFrame.describe()`

Generate descriptive statistics

Tabular data

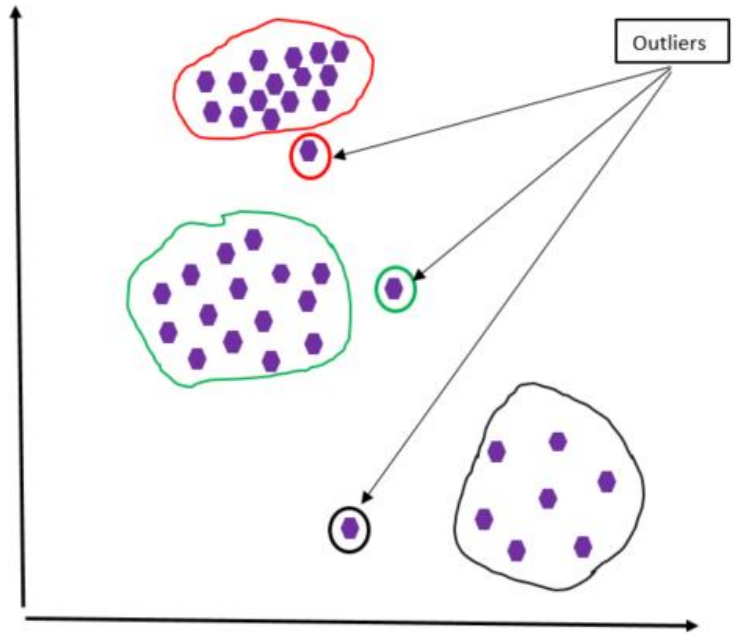
Common Issues in Tabular Data:

- ✓ **Incomplete** – missing or null values in some fields
- ✓ **Noisy** – random errors, typos, or extreme outliers
- ✓ **Inconsistent** – conflicting or duplicated information (e.g., mismatched dates, names, or formats)
- ✓ **Imbalanced / skewed data** – one group is much more frequent than others (e.g., 90% of survey answers are “Yes”, only 10% are “No”)
- ✓ **Irrelevant or redundant features** – columns that don’t add useful information or repeat existing ones

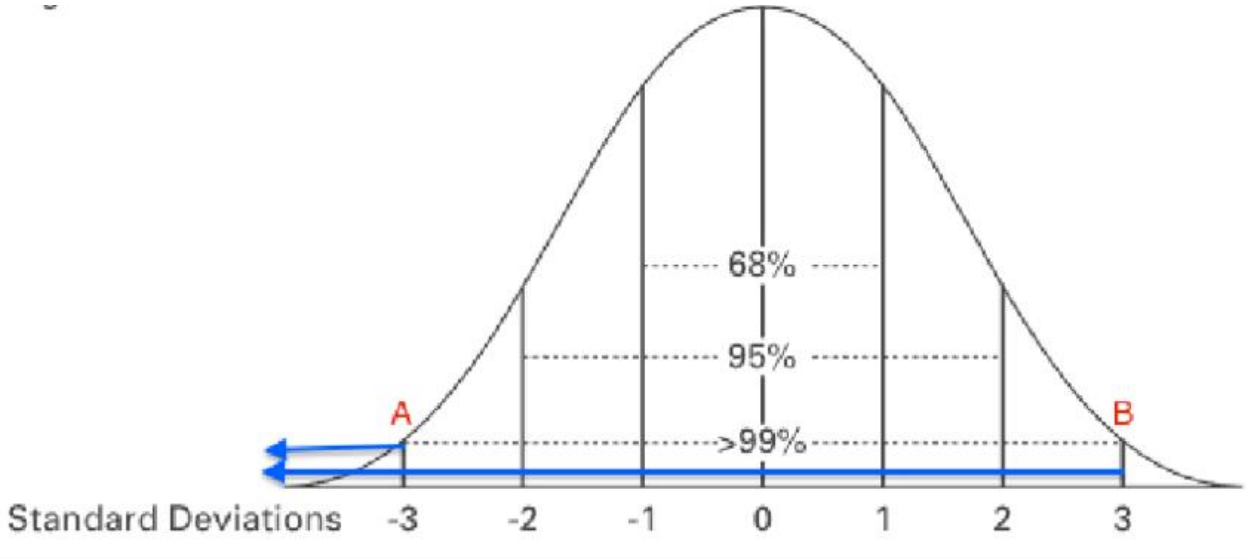


Data cleaning

Outliers detection



Three sigma rule



Normally data with $|x-\mu|>3\sigma$ are considered as outliers

Data cleaning

Missing data

- may be deleted
- may be filled by:
  - ✓ the attribute mean
  - ✓ the attribute mean for all samples belonging to the same class
  - ✓ or other

`pandas.DataFrame.dropna()`

Remove missing values.

pandas.DataFrame.fillna()
Fill missing values

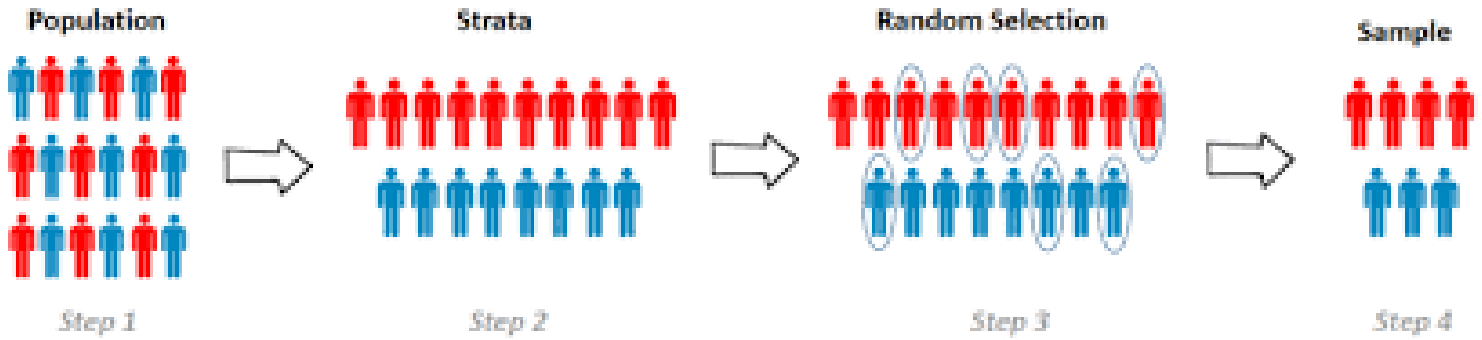
	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

df.fillna(0)

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	0.0
1	9	0.0	9.0	0	7.0
2	19	17.0	0.0	9	0.0

How to Handle Imbalanced Data

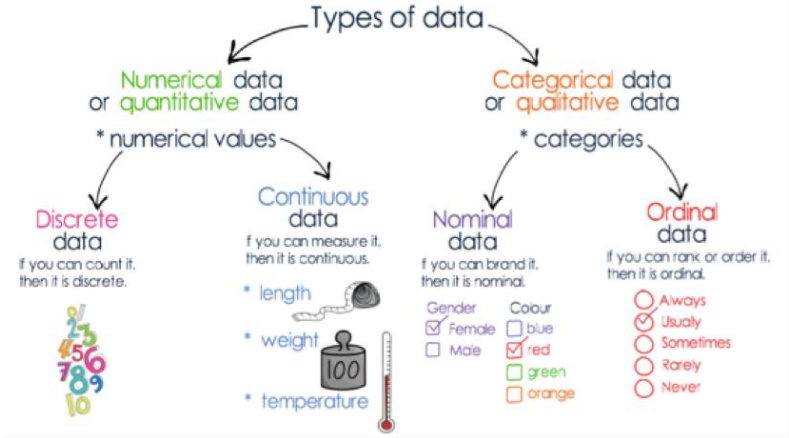
- **Collect more data** – if possible, gather additional samples for rare cases
- **Oversampling** – synthetically generate more examples of the minority group
- **Undersampling** – reduce the number of majority group examples
- **Class weights** – give more importance to underrepresented groups in model training
- **Stratified sampling** – keep the same proportion of groups in train/test splits



Strata — groups of data formed according to a specific attribute (e.g., gender, city, age group).

Data Types in Tabular Data

- Numeric: integers, floats – measurable values for calculations
- Categorical: discrete groups like gender, color, country
- Ordinal: ordered categories, e.g., education level
- Textual/String: free-form text data
- Date/Time: timestamps, can be split into year, month, weekday, etc.

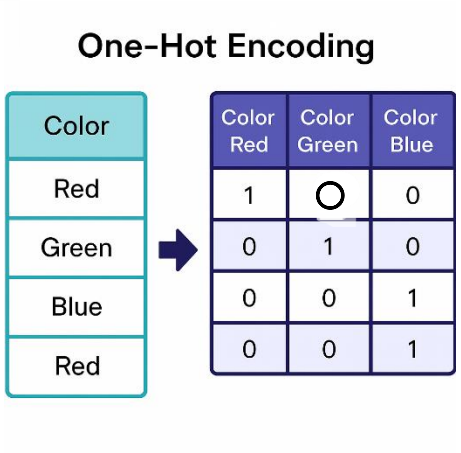


Numeric	Categorical	Ordinal	Text	Date/Time
Age	Gender	Rating	Name	Date
29	Female	4★	Smith	2023-01-15

sklearn.preprocessing

Encoding Categorical Data:

- **OneHotEncoder** – for nominal (unordered) categories
- **OrdinalEncoder** – for ordinal (ordered) categories
- **LabelEncoder** – for the target variable (y), sometimes for binary features




```
import pandas as pd
from sklearn.preprocessing import
OneHotEncoder
data = pd.DataFrame({"color": ["red",
"green", "blue","red"]})
encoder = OneHotEncoder(sparse_output=False)
encoded =
encoder.fit_transform(data[["color"]])
```

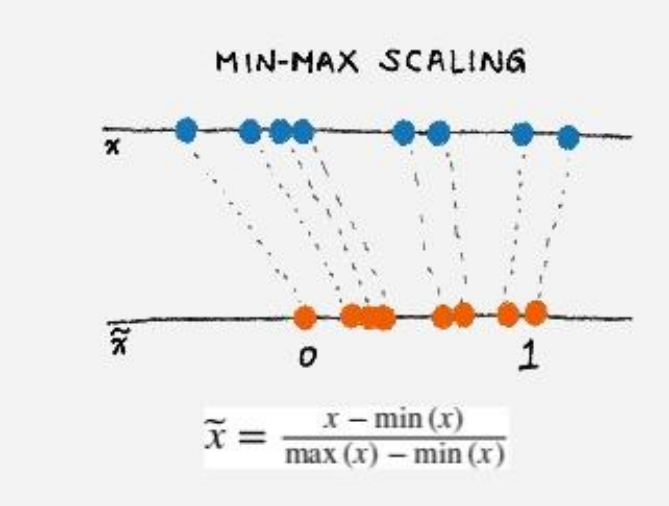
```
[[0. 0. 1.]
[0. 1. 0.]
[1. 0. 0.]
[0. 0. 1.]]
```

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder
data = pd.DataFrame({"size": ["small", "medium", "large", "medium"]})
encoder = OrdinalEncoder(categories=[["small", "medium", "large"]])
encoded = encoder.fit_transform(data[["size"]])
print(encoded)
```

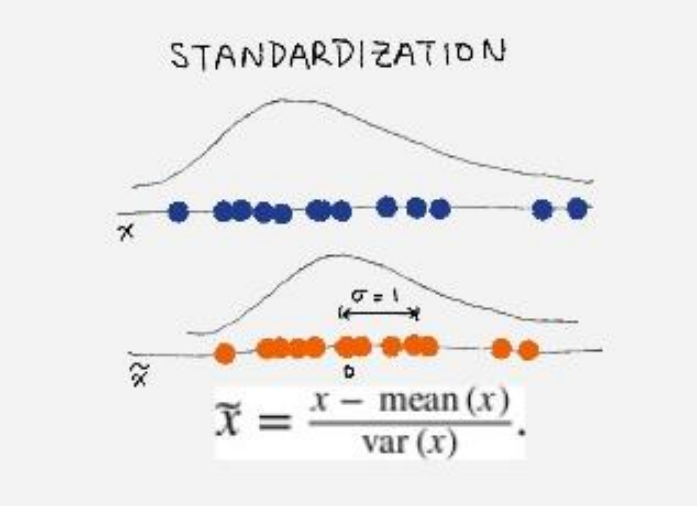
```
[[0.] [1.] [2.] [1.]]
```

Data Normalization

After min-max scaling, all feature values are within the [0, 1] range



After standardization, a feature has mean 0 and variance 1



Scaling & Normalization

- **StandardScaler** – standardization (mean = 0, std = 1)
- **MinMaxScaler** – scaling to range [0, 1]

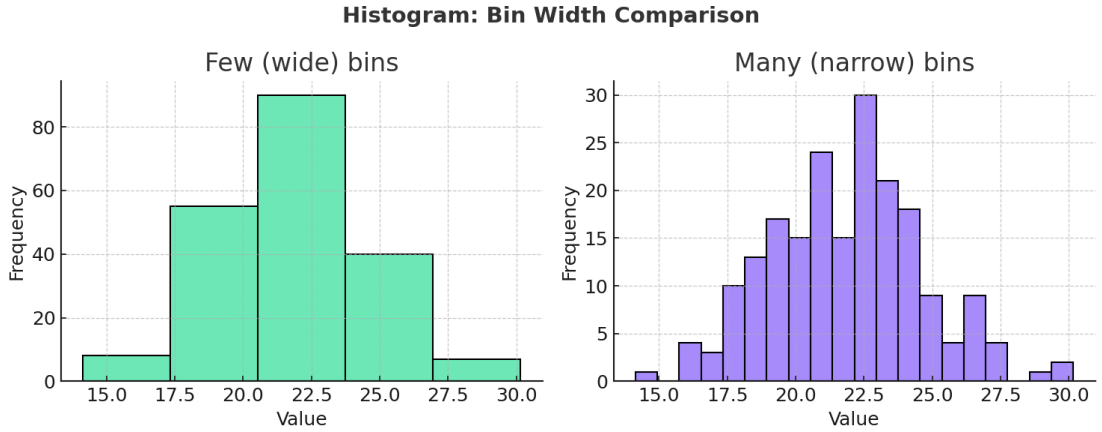
```
from sklearn.preprocessing import StandardScaler
```

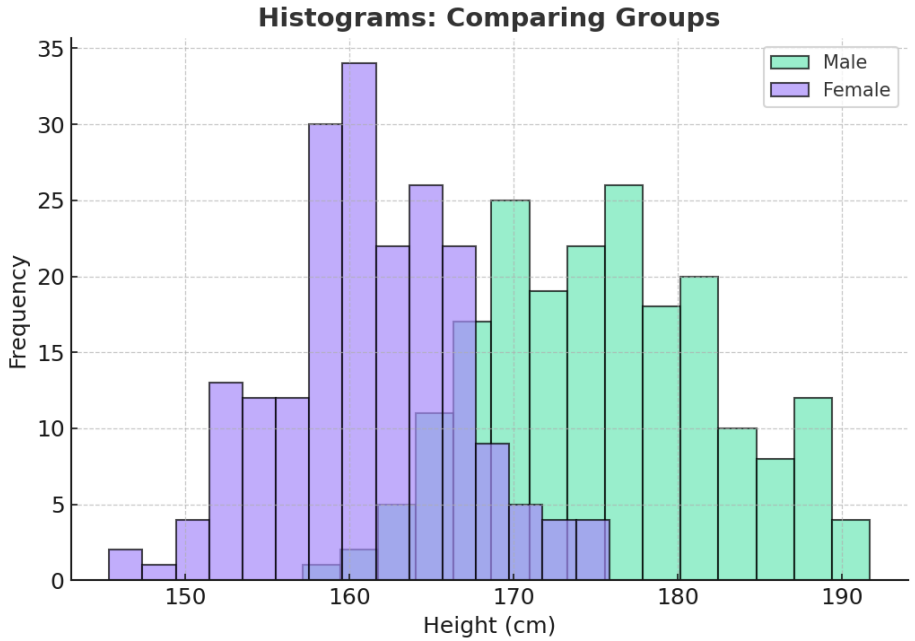
https://www.slideshare.net/gabrielspmoreira/feature-engineering-getting-most-out-of-data-for-predictive-models

Data Visualization: Histogram

- A histogram is an estimate of the probability distribution of a continuous variable.
- To construct a histogram: divide the range of values into bins and draw a rectangle over each bin with a height proportional to its frequency.

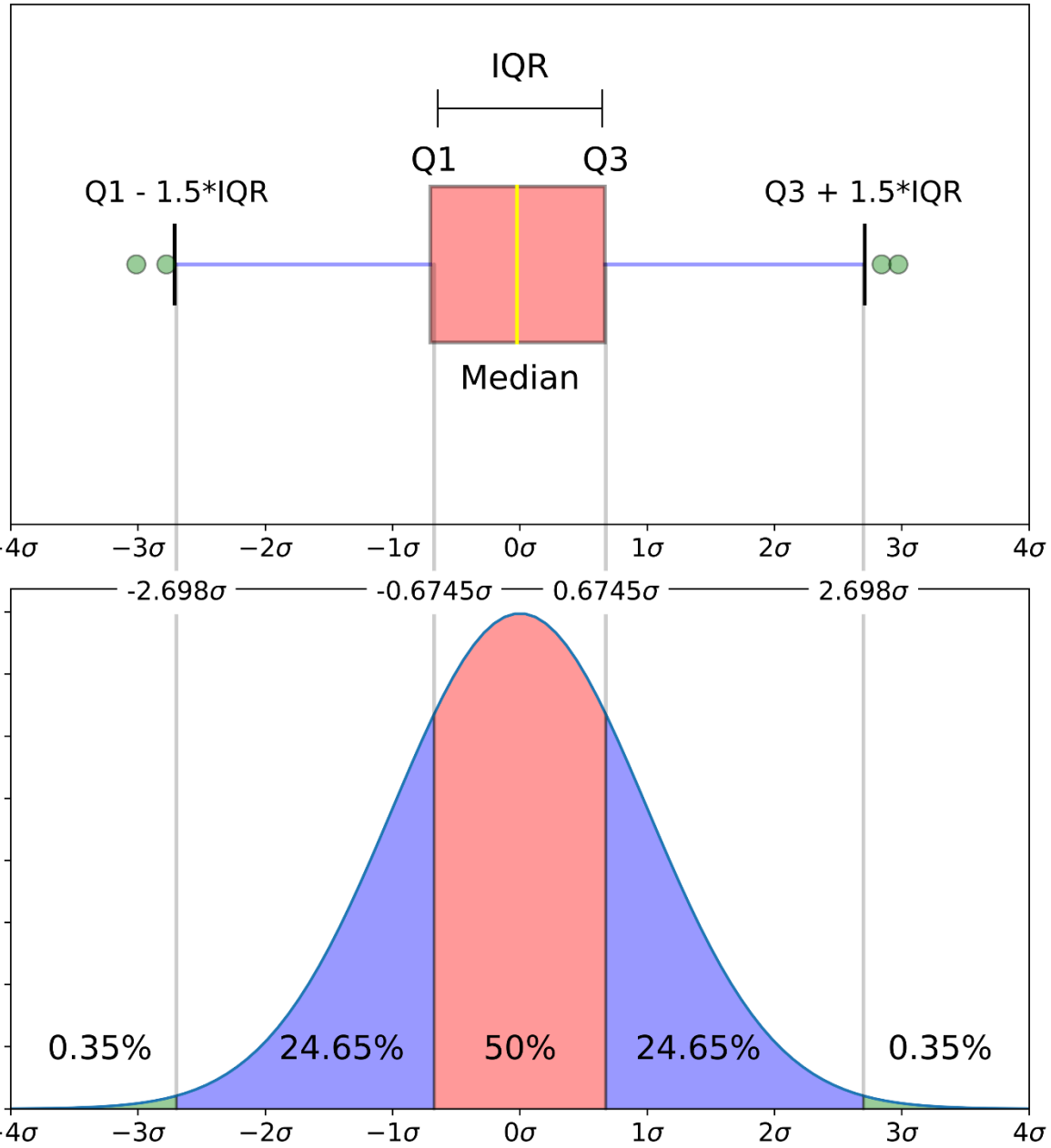
```
seaborn.distplot()
```

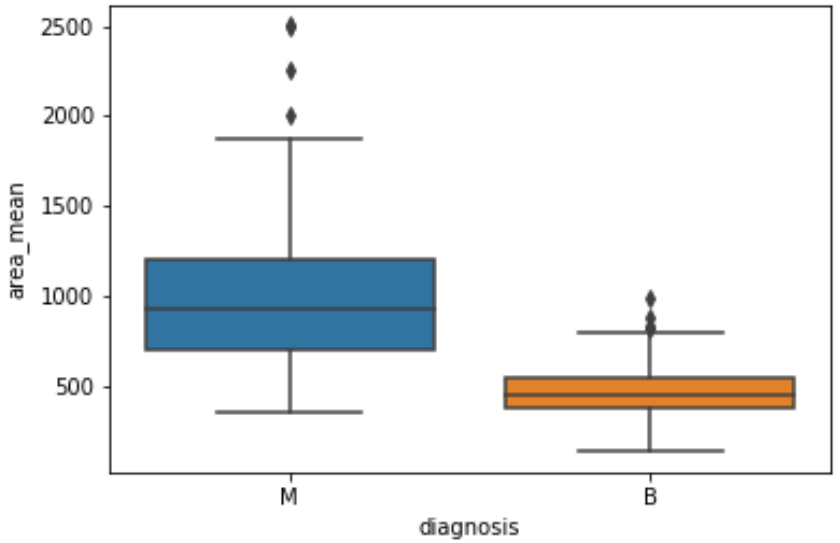




The distributions are clearly different → this feature is informative for the model

Data Visualization: Box Plot

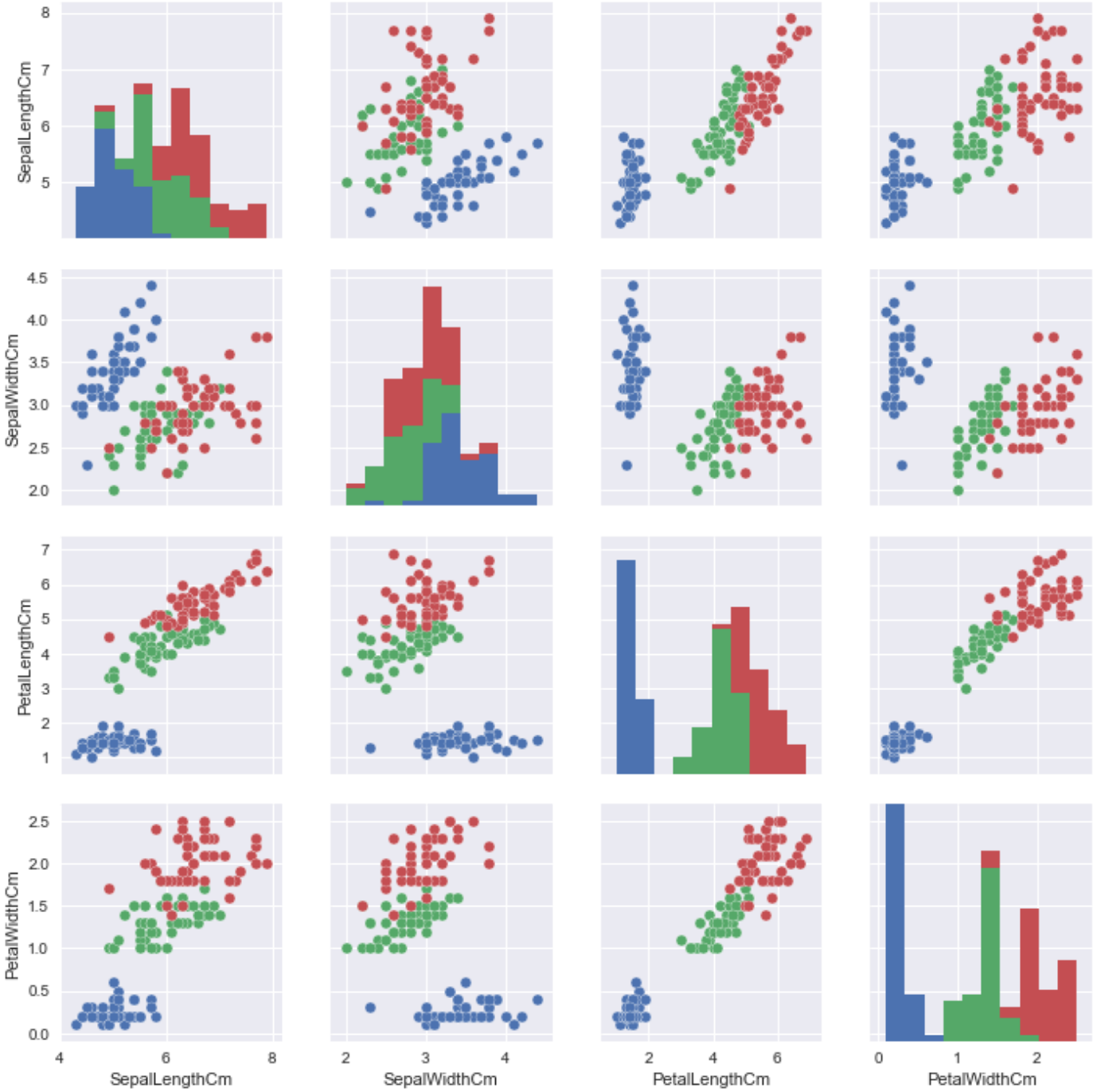




A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum").

```
seaborn.boxplot()
```

Iris dataset



Plot pairwise relationships in a dataset

```
seaborn.pairplot()
```

If colors are well separated → the feature is informative.
If colors are mixed → the feature is less useful.

Pearson correlation coefficient

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2} \sqrt{\sum (y - \bar{y})^2}}$$

Pearson's correlation coefficient is
the covariance of the two variables divided by the product of their standard deviations

```
seaborn.heatmap()
```