



University of Udine

Big Data Management, Analysis, and Presentation

A whirlwind introduction to Data Mining

Andrea Brunello

andrea.brunello@uniud.it

What is data mining



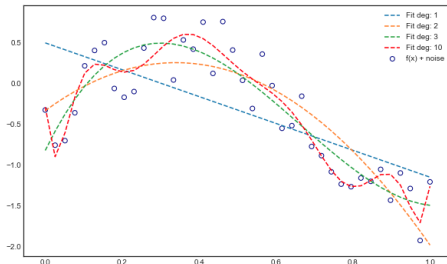
What is Data Mining

- **Data** \approx **facts** that are recorded and stored
- **Information** consists of **concepts**, regularities, or patterns that are “hidden” in the data
- **Data Mining** is the process of **discovering and presenting** useful, previously unknown information, implicitly contained in large sets of data
 - For example, finding trends in student study habits and how they relate to exam scores
- **Models** are created to represent this hidden information
 - Example: An equation predicting student scores based on study time
- These models are built automatically by **machine learning** algorithms, which learn patterns from the data

Just like a map simplifies reality to help you navigate, a model simplifies data to help you understand or predict

- In our case, the “reality” is represented by the data we have at our disposal
- We want to create a **simpler version** of reality
- A model is a simplified way to describe our data. It helps us understand or predict a particular phenomenon

Which of the colored curves best represents the phenomenon underlying the data points?





Using models

Remember our model predicting house values:

$$\begin{aligned} \text{value} = & - 34.8 * \text{surface} \\ & + 57740.3 * \text{rooms} \\ & + 39347.8 * \text{marble} \\ & + 49276.1 * \text{restored} \\ & - 7843.3 \end{aligned}$$

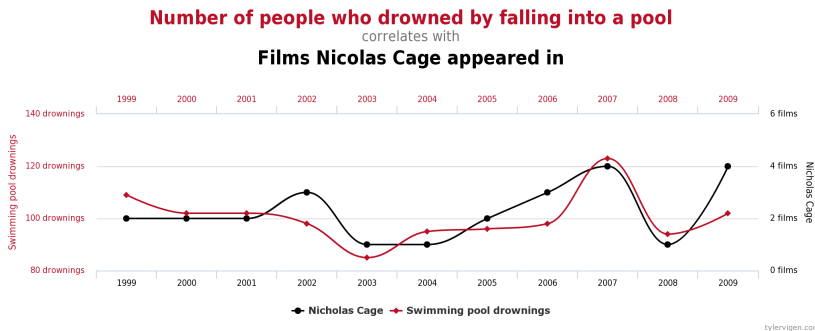
It can be used to:

- **Understand:** identify how certain features of a house are more important than others in determining its value
 - *Does anything look unusual about how the model handles certain features?*
- **Predict:** determine the final value of a house based on a set of its features
 - *What would the value be for a restored house with a surface of 300, 4 rooms, and no marble?*



Correlation does not imply causation!

Often, discovered patterns may be trivial, the result of random correlations, or not entirely accurate





Types of data

- Data Mining can be applied to various types of data:
 - **Structured:** tabular data
 - **Unstructured:** text, images, audio, video
- Traditional machine learning techniques require converting unstructured data into structured forms
- Unlike traditional methods, neural networks can directly process images or videos without converting them into tables, making it powerful for tasks like image recognition
- In this course, we will focus on structured data

A tabular dataset used by traditional machine learning algorithms has the following structure:

| Temperature | Outlook | Humidity | Windy | Played? |
|-------------|----------|----------|-------|---------|
| Mild | Sunny | 80 | No | Yes |
| Hot | Sunny | 75 | Yes | No |
| Hot | Overcast | 77 | No | Yes |
| Cool | Rain | 70 | No | Yes |
| Cool | Overcast | 72 | Yes | Yes |
| Mild | Sunny | 77 | No | No |

- Each row represents **an instance**, i.e., an example of the concept to be learned
- The rows are independent of each other
- Each instance describes the values of a given set of attributes (columns), which can be numerical or categorical



The Data Mining Process

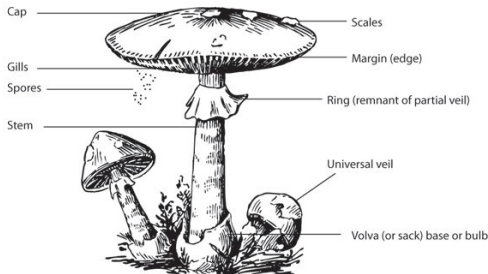
The Data Mining process typically follows these steps:

- **Start with a question:** define the problem or goal you want to solve
- **Collect relevant data:** gather the necessary data for analysis
- **Explore and preprocess the data:** get to know and massage the dataset to prepare it for analysis
- **Select key attributes:** choose the most important features for the analysis
- **Train a model:** apply a machine learning **algorithm** to learn a model from the data
- **Evaluate the model:** assess the model's performance

An example: Mushroom dataset

Let's walk through the Data Mining process with an example focused on prediction

- **Question:** is it possible to automatically distinguish between poisonous and edible mushrooms?
- **Data:** 8124 instances of already classified mushrooms, each described by 22 attributes, plus the class label
 - <https://archive.ics.uci.edu/ml/datasets/mushroom>





An example: Mushroom dataset (2)

Content of the first 11 rows of the *dataset*

| cap_shape | cap_surface | cap_color | bruises_or_no | odor | gill_attachment | gill_spacing | gill_size | gill_color | stalk_shape | stalk_root | stalk_surface_above_ring |
|-----------|-------------|-----------|---------------|---------|-----------------|--------------|-----------|------------|-------------|------------|--------------------------|
| convex | smooth | brown | bruises | pungent | free | close | narrow | black | enlarging | equal | smooth |
| convex | smooth | yellow | bruises | almond | free | close | broad | black | enlarging | club | smooth |
| bell | smooth | white | bruises | anise | free | close | broad | brown | enlarging | club | smooth |
| convex | scaly | white | bruises | pungent | free | close | narrow | brown | enlarging | equal | smooth |
| convex | smooth | gray | no | none | free | crowded | broad | black | tapering | equal | smooth |
| convex | scaly | yellow | bruises | almond | free | close | broad | brown | enlarging | club | smooth |
| bell | smooth | white | bruises | almond | free | close | broad | gray | enlarging | club | smooth |
| bell | scaly | white | bruises | anise | free | close | broad | brown | enlarging | club | smooth |
| convex | scaly | white | bruises | pungent | free | close | narrow | pink | enlarging | equal | smooth |
| bell | smooth | yellow | bruises | almond | free | close | broad | gray | enlarging | club | smooth |
| convex | scaly | yellow | bruises | anise | free | close | broad | gray | enlarging | club | smooth |

| stalk_surface_below_ring | stalk_color_above_ring | stalk_color_below_ring | veil_type | veil_color | ring_number | ring_type | spore_print_color | population | habitat | class |
|--------------------------|------------------------|------------------------|-----------|------------|-------------|------------|-------------------|------------|---------|-----------|
| smooth | white | white | partial | white | 1 | pendant | black | scattered | urban | poisonous |
| smooth | white | white | partial | white | 1 | pendant | brown | numerous | grasses | edible |
| smooth | white | white | partial | white | 1 | pendant | brown | numerous | meadows | edible |
| smooth | white | white | partial | white | 1 | pendant | black | scattered | urban | poisonous |
| smooth | white | white | partial | white | 1 | evanescent | brown | abundant | grasses | edible |
| smooth | white | white | partial | white | 1 | pendant | black | numerous | grasses | edible |
| smooth | white | white | partial | white | 1 | pendant | black | numerous | meadows | edible |
| smooth | white | white | partial | white | 1 | pendant | brown | scattered | meadows | edible |
| smooth | white | white | partial | white | 1 | pendant | black | several | grasses | poisonous |
| smooth | white | white | partial | white | 1 | pendant | black | scattered | meadows | edible |
| smooth | white | white | partial | white | 1 | pendant | brown | numerous | grasses | edible |

52% of the instances are classified as *edible*, 48% as *poisonous*



An Example: Mushroom Dataset (3)

Idea:

- 1 Partition the instances based on the value of the attribute *bruises_or_not*
- 2 Determine that an instance is classified as *edible* if and only if *bruises_or_not* = *bruises*

Based on this simple rule (= the model approximating the reality described by our data), we achieve a classification accuracy of 74% over our data

Note: The model is not error-free, meaning it doesn't perfectly describe the data we observe

What do you think might happen if we consider a more complex rule?

Model evaluation

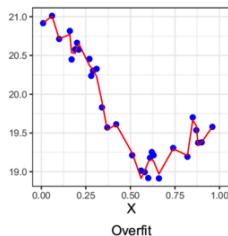
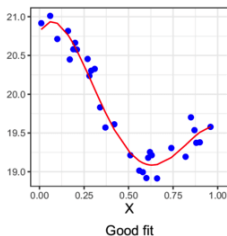
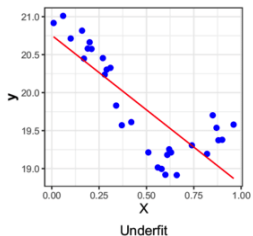


- **Generalization:** a model needs to work well not just on the data it has been built upon, but also on new, unseen data
- Therefore, the machine learning algorithm must generate a **model that can generalize well**, and work even for instances it hasn't seen during training
- *Example:* A model that can predict the value of a new house based on patterns learned from previously sold homes



Models and Generalization (2)

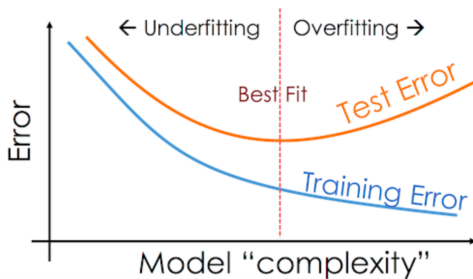
- A prediction model that achieves excellent results on the training data could still fail on new data
- Intuitively, this happens because the model might have learned spurious patterns due to “noise” (*overfitting*)
- It is like a student memorizing answers from past exams instead of properly learning the subject



What's the model with the highest chance of generalizing well here?

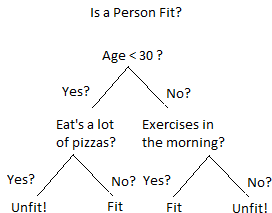
Complex models tend to perform well on training data but may generalize less effectively on new data

Conversely, **too simple models** may fail in capturing the underlying patterns in the data



Models and Generalization (4)

- Let's explore what happens when we use a more complex model, like a **decision tree**, on the Mushroom dataset
- A decision tree splits the data using multiple features, leading to more detailed classifications
- Each node in the tree represents a feature, and each split helps narrow down the classification
- This model is more powerful but also more complex, which could **increase the risk of overfitting**

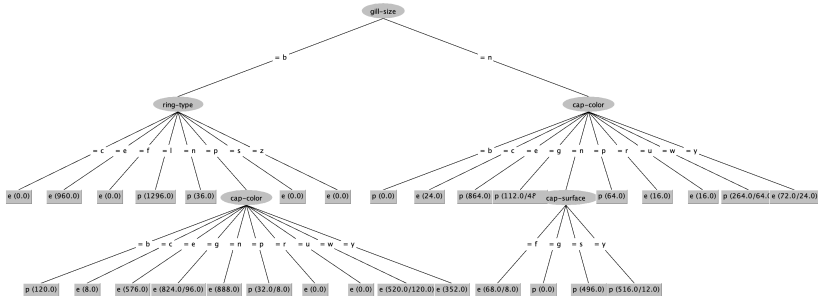




Training and test data

- We must be smart in how we build the tree, to properly evaluate its generalization capability
- To do that:
 - ① We randomly split (50-50) the dataset instances into a training and a test subsets
 - ② We build the decision tree just by looking at the training set instances
 - ③ We evaluate how the tree performs over the test instances
- We obtain an accuracy of 92% on the test data
- But, on the training data, the very same tree has an accuracy of over 95%!
- This is a sign that the model may have overfitted to the training data

Decision tree over the Mushroom dataset's training split, which resulted in 95% accuracy on the training data



Confusion matrixes

- Classification errors are not born equal
- E.g., we would prefer the model to incorrectly classify *an edible mushroom as poisonous* rather than misclassify *a poisonous one as edible*
- A confusion matrix allows us to take a closer look at the classification performance of a model

| | | ACTUAL CASE INSTANCE | |
|---------------------------------------|-----------|----------------------|--------|
| | | POISONOUS | EDIBLE |
| LABEL PREDICTED BY THE MODEL | POISONOUS | 1796 | 179 |
| | EDIBLE | 122 | 1965 |

CONFUSION MATRIX OVER THE 4062 TEST SET INSTANCES

$$\text{Accuracy} = (1796 + 1965) / (1796 + 179 + 122 + 1965) \approx 0.92$$

Formally, in a binary setting, we talk about *positive* and *negative* classes, and then distinguish between *false positive* and *false negative* errors

| | | Actual Values | |
|------------------|-----------------------|----------------|-----------------------|
| | | Has migraine | Doesn't have migraine |
| Predicted Values | Has migraine | True Positive | False Positive |
| | Doesn't have migraine | False Negative | True Negative |

Here: migraine = positive class, no-migraine = negative class

Can you think of other real-life examples where false positives are more dangerous than false negatives? (hint: ✉)



Supervised vs Unsupervised Learning

- In our mushroom example, we built a model with a clear goal: predicting whether a mushroom is **edible/poisonous**
- This is called **supervised learning**: the algorithm is given a known label (the target, i.e., "edible or not") for each instance in the dataset, and the goal is to create a model that can predict this label for new instances based on other attributes (called predictors)
- In contrast, **unsupervised learning** methods aim to find general patterns and structures in the data without using predefined labels
 - For example, finding groups of similar instances without knowing the actual categories beforehand (e.g., in customer segmentation)
- We won't cover unsupervised learning

Data exploration and preparation



- Very rarely a dataset can be used as it is for analysis tasks
- The most time-consuming phase of the entire data mining process is data exploration and preparation
- **It is a crucial step**, as all subsequent analyses depend on it
- Some typical tasks:
 - choices regarding the encoding of attributes
 - detection and handling of missing values
 - removal of columns with low variance
 - identification of outliers
 - dependencies between columns
 - feature selection, removal of irrelevant columns



Attribute encoding

- Example: attribute related to *pH*: should we keep the numerical values or use “coarser” categorical values (acidic, neutral, basic)?
- What precision should be considered for numerical attributes? (e.g., integers, decimals, how many decimal places, ...)
 - for example, if a temperature sensor is capable of providing an accurate measurement to the second decimal place, it is unnecessary to keep a number like 37.234664367

What to do when some instances have unknown (null) values?

| Temperature | Outlook | Humidity | Windy | Played? |
|-------------|----------|----------|-------|---------|
| Mild | Sunny | ? | No | Yes |
| Hot | ? | 75 | ? | No |
| Hot | Overcast | 77 | No | Yes |

Several solutions, for example:

- remove those instances
- remove columns with too many null values
- replace null values with a default value
- replace null values with the mean/mode of the column
- other more sophisticated imputation techniques

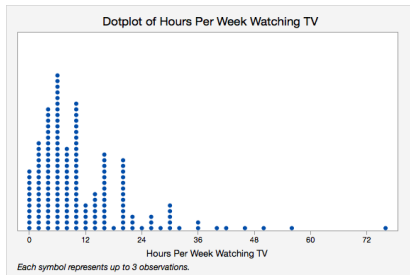
Additionally, the “type” of missing value plays an important role (why is it missing?)

Low variance columns

- In the dataset, there may be columns that have only one value or a very imbalanced distribution of values
- *Typically*, such columns provide little information
- A machine learning algorithm may detect this and disregard them when generating models
- However, removing them beforehand leads to a more efficient learning process

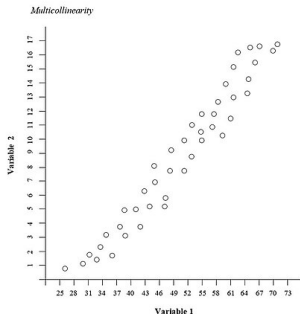
| ID | season | holiday | workingday | weather | f5 | temp | atemp | humidity | windspeed | count |
|-------|--------|---------|------------|---------|----|-------|--------|----------|-----------|-------|
| AB101 | 1 | 0 | 0 | 1 | 7 | 9.84 | 14.395 | 81 | 0.0000 | 16 |
| AB102 | 1 | 0 | 0 | 1 | 7 | 9.02 | 13.635 | 80 | 0.0000 | 40 |
| AB103 | 1 | 0 | 0 | 1 | 7 | 9.02 | 13.635 | 80 | 0.0000 | 32 |
| AB104 | 1 | 0 | 0 | 1 | 7 | 9.84 | 14.395 | 75 | 0.0000 | 13 |
| AB105 | 1 | 0 | 0 | 1 | 7 | 9.84 | 14.395 | 75 | 0.0000 | 1 |
| AB106 | 1 | 0 | 0 | 2 | 7 | 9.84 | 12.880 | 75 | 6.0032 | 1 |
| AB107 | 1 | 0 | 0 | 1 | 7 | 9.02 | 13.635 | 80 | 0.0000 | 2 |
| AB108 | 1 | 0 | 0 | 1 | 7 | 8.20 | 12.880 | 86 | 0.0000 | 3 |
| AB109 | 1 | 0 | 0 | 1 | 7 | 9.84 | 14.395 | 75 | 0.0000 | 8 |
| AB110 | 1 | 0 | 0 | 1 | 7 | 13.12 | 17.425 | 76 | 0.0000 | 14 |

- In statistics, an outlier is a term used to define, in a set of observations, an anomalous value, that is, a value clearly distant from the other observations
- An outlier may be due to errors in data collection or measurement
- There is no unique mathematical definition of an outlier
- Be cautious when removing or correcting them



Dependencies between columns

- It may happen that the values of some columns are strongly related (correlated) to each other
- In such cases, it is sufficient to keep only one column
- This not only makes the analysis phase more efficient but also allows the trained models to correctly determine the “contribution/role” of each predictor



Note. The plot depicts the predictor variables, Variable 1 and Variable 2, as highly linearly related.

$$\begin{aligned}
 \text{value} = & -34.8 * \text{surface} \\
 & + 57740.3 * \text{rooms} \\
 & + 39347.8 * \text{marble} \\
 & + 49276.1 * \text{restored} \\
 & - 7843.3
 \end{aligned}$$

- The process of determining which attributes to keep and which to discard is called *feature selection*
- In the previous slides, we saw some examples of column removal
- There are many techniques in the literature for performing feature selection, as it plays a crucial role before the actual analysis:
 - creation of simpler models
 - creation of more accurate models
 - creation of more interpretable models
 - creation of models that can better generalize to new data



Bibliography

- Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. 2016. *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques* (4th. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: Springer.
- Bishop, C. M., and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: Springer.