University of Ljubljana

SEB SCHOOL OF ECONOMICS AND BUSINESS

# 11. Machine Learning

*Prof. Dr. Miroslav Verbič*

miroslav.verbic@ef.uni-lj.si

www.miroslav-verbic.si

EFMD EQUIS ACCREDITED

AACSB ACCREDITED

ASSOCIATION AMBA ACCREDITED

Ljubljana, October 2025

University of Ljubljana

SEB SCHOOL OF ECONOMICS AND BUSINESS

# 11.1 Introduction to Machine Learning

EFMD EQUIS ACCREDITED

AACSB ACCREDITED

ASSOCIATION AMBA ACCREDITED

# Basic concepts

- **Big data** is *not* essentially a *new* concept; we were *always* surrounded by large amounts of data.
- What is new is that we can *nowadays* **record**, **store**, **process** and **analyse** large amounts of data, which has the potential to inform ***action***.
- **Machine learning (ML)** can be understood as the field of study devoted to **computer algorithms** that transform *data* into *actionable knowledge*.
- It evolved at the intersection of **rapidly advancing** *available data*, *computing power* and *statistical methods*.
- ML thus focuses on teaching computers how to use data and *autonomously* develop models to solve problems.

# Basic concepts

- A related concept is **data mining**, which focuses on teaching computers to *detect* or *identify patterns* that humans then use to *solve a problem*.
- **Most** data mining nowadays involves the use of ML, but ***not all*** ML requires data mining (ML can use e.g. classification or numeric prediction instead of pattern detection).
- Another related concept is **artificial intelligence (AI)**, which is a field of computer science focused on creating systems that can perform *tasks that typically require human intelligence*, such as understanding language, recognizing patterns, solving problems and making decisions.
- ML is thus a ***subset*** of AI, focused on enabling computers to **learn** from data **autonomously**, i.e. *without* being explicitly programmed for *specific* tasks.

# Basic concepts

- Examples of AI that do ***not*** **involve ML** include rule-based systems, pathfinding algorithms and automated scheduling.
- Some **differences** among statistics, ML and AI:

| | Traditional Statistics | Machine Learning | Artificial Intelligence |
|---|---|---|---|
| **Application** | Hypothesis testing and insight | Prediction and knowledge generation | Automation |
| **Success criterion** | Greater understanding | Ability to intervene before things happen | Efficiency and cost savings |
| **Success metric** | Statistical significance | Trustworthiness of predictions | Return on investment (ROI) |
| **Input data size** | Smaller data | Medium data | Bigger data |
| **Implementation** | Reports and presentations for knowledge sharing | Predictions or interventions in business practices | Custom applications and automated processes |

# Successful applications

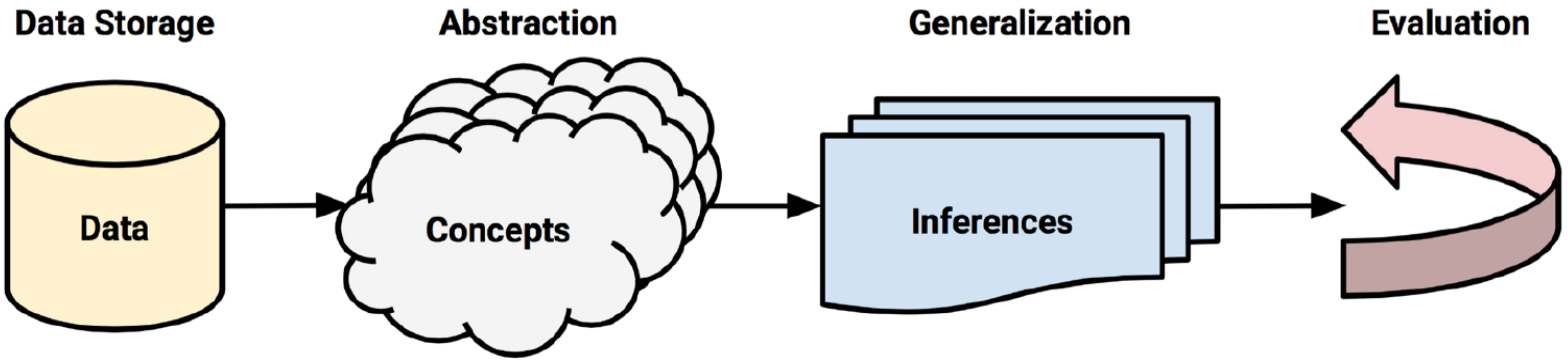
- Successful ML **applications** in economics and finance:
  - fraud detection;
  - algorithmic trading;
  - credit scoring and risk assessment
  - portfolio management;
  - economic forecasting;
  - customer segmentation and personalization;
  - sentiment analysis for financial markets;
  - loan approval and pricing;
  - financial market prediction;
  - automated financial reporting.

# Ethical issues

- **Ethics** is crucial in ML to **ensure** that the models are *fair*, *transparent* and do not perpetuate *bias* or *harm*, protecting individuals' *rights* and maintaining *trust* in AI systems.
- **Caution** should thus be exercised when obtaining or analyzing data to **avoid**:
  - breaking *laws* and *regulations*;
  - violating *terms of service* and
  - abusing *privacy*.
- Computers may ***accidentally*** learn to **discriminate** when ML models are *trained on biased data*, causing them to *replicate or amplify* existing *societal biases* in their predictions or decisions, even if the bias was *unintentional* or *implicit* in the (de-identified) data.

# How machines learn

- A machine *learns* if it utilizes its experience such that its **performance *improves*** on similar experiences *in the future*.
- The basic **learning process** can be divided into four *interrelated* components:



# Learning process: Data storage

1. **Data storage** utilizes *observation*, *memory*, and *recall* to provide a ***factual basis*** for further reasoning.

- Like humans, computers can access **short and long-term memory** (RAM and HDD in combination with CPU).
- Even though data storage is the basis for reasoning, **recall** alone is ***insufficient* for learning**; additional steps are required to make memories useful for future tasks.

- Data science is an interdisciplinary field that *combines* statistics, econometrics, machine learning, data mining and programming to *extract* meaningful insights and *knowledge from* structured and unstructured *data*.

# Learning process: Abstraction

2. **Abstraction** involves the *translation* of stored *data* into broader representations and *concepts*. It ***adds*** **meaning and context** to raw data.

- *Abstracted concepts* are the basis of knowledge representation, i.e. the formation of structures that turn sensory data into meaningful insight.
- Whereas humans make abstract connections quite *intuitively*, computers must do it *explicitly*. They summarize stored raw data **using a *model***.
- A model in ML is an *explicit* representation of *patterns* in data, intended to represent an **idea *greater* than the sum of its parts**.

# Learning process: Abstraction

- There are many *different* **types of models**:
  - mathematical *equations*,
  - relational *diagrams*, such as trees and graphs,
  - logical *rules* (if, else, etc.),
  - groups of data known as *clusters*.
- **Building a model** to describe a set of data, i.e. fitting a model to a dataset, is known in ML as training.

**Observations** → **Data** → **Model**

| Distance | Time |
|---|---|
| 4.9m | 1s |
| 19.6m | 2s |
| 44.1m | 3s |
| 78.5m | 4s |

$$g = 9.8m/s^2$$

# Learning process: Generalization

3. **Generalization** turns **abstracted knowledge** into a form that can be **utilized for future action**, on tasks that are *similar, but not identical*, to those it has seen before.

- It can be imagined as a ***search* through the set of models** (theories or inferences) that could be **obtained during *abstraction***.
- It limits **discovered *patterns*** to those likely to be **useful** for future tasks.
- It is typically *impossible* to evaluate *every* potential abstraction, therefore **shortcuts** called heuristics are employed.

# Learning process: Generalization

- Heuristics utilize *approximations* and other *rules of thumb*, which means they are ***not* guaranteed to find the *best* model** of the data.
- However, *without them*, finding useful information in a large dataset (generalization) would be *infeasible*.
- The process of generalization thus **results in a bias**, or a systematic error, when encountering specific types of cases.
- ***All* ML algorithms *have* a bias.** The bias allows the computer to observe *some* patterns at the expense of *others*.
- Necessary or inductive bias refers to the *assumptions* made by an algorithm that *allow it to generalize* from limited training data to unseen data, enabling the model to make *meaningful predictions*.

# Learning process: Evaluation

4. **Evaluation** refers to the **measurement** of a model's *success in spite of its biases*, to inform *additional training* if needed.

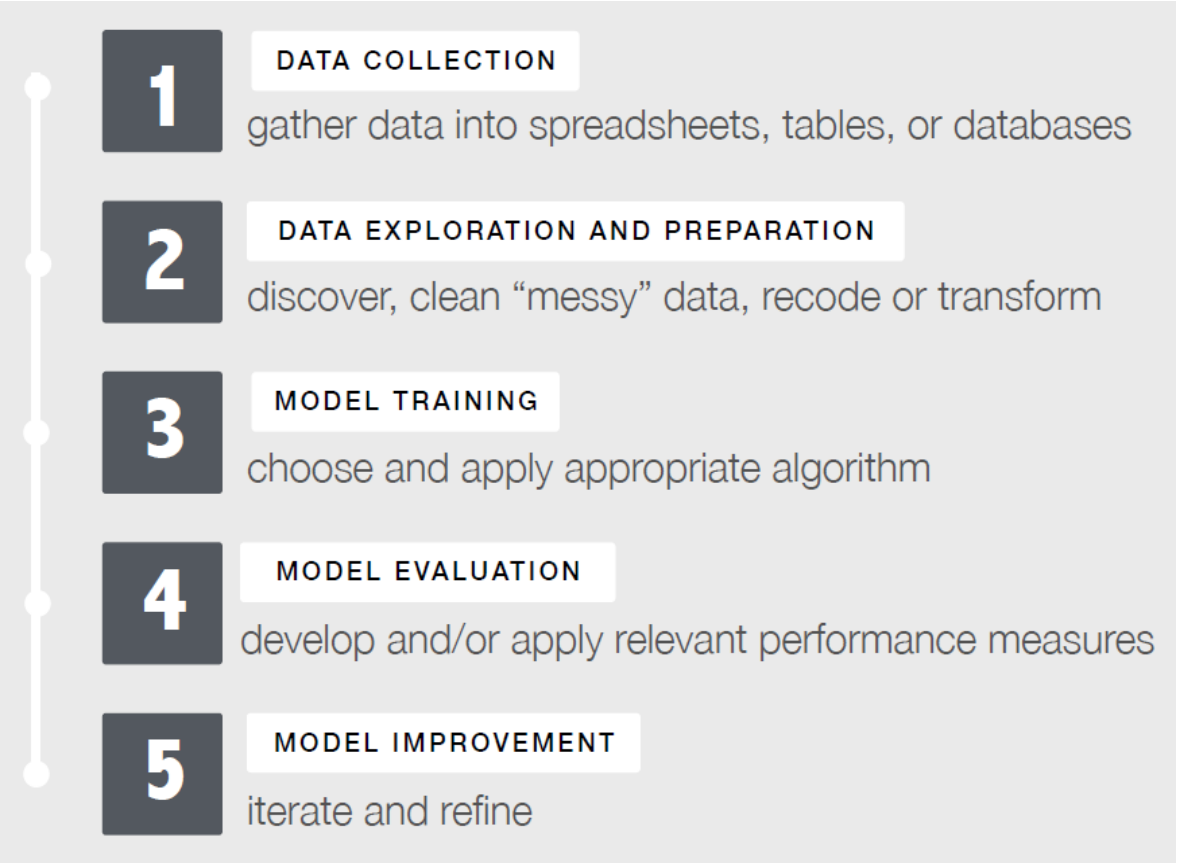
- Often, **data is divided** into *training*, *validation*, and *testing sets* in order to *build*, *select*, and *evaluate candidate models*, respectively.
- ***No* ML approach is best for *every* circumstance.** There is a cost or *trade-off* in every decision that is made. This is known as the "No free lunch theorem".
- Learning ***may* perform poorly** *either* because its biases are poorly suited to a particular problem *or* because of noisy data that contains many unexplained or unexplainable data points.

# Learning process: Evaluation

- Noise is caused by ***seemingly* random events**, such as:
  - *measurement error* (e.g. imprecise sensors or inaccurate data entry);
  - *human subjectivity* (e.g. personal interpretation or temporary emotions);
  - *data quality issues* (e.g. missing, null, truncated or incorrectly coded values);
  - *extreme complexity* (e.g. high-dimensional data or chaotic systems).
- Attempting to explain the noise results in overfitting. Model performs *well* during training, but *poorly* during evaluation.
- An overly complex model, overfitted to the training dataset, may ***not* generalize** well to future cases.

# Applying learning algorithms to data

- To *apply* the learning process to real-world tasks, we use a five-step process:



# Understanding data (1/4)

- Steps 1 and 2 are sometimes called data wrangling. These tasks can consume a surprisingly *large portion of the ML process*; 80% or even more of the effort in some cases.
- A unit of observation is the *smallest entity to be examined*:
  - objects,
  - persons,
  - geographical regions,
  - time points,
  - measurements,
  - combined units, e.g. person-years.
- A related concept is the unit of analysis, which *sometimes differs* from the unit of observation (e.g. persons might be the unit of observation and countries the unit of analysis).

# Understanding data (2/4)

- *Datasets* that describe the units of observation *consist of*:
  - examples, cases or instances of the units for which properties have been recorded;
  - features, which are the recorded properties that may be useful for learning.
- In statistics or econometrics, we usually call **examples** *observations* and **features** *variables*.
- *Data* can be:
  - unstructured, such as free-form text, pictures or sound;
  - structured, where each example of the phenomenon has exactly the *same set of features*.
- Examples and features are often stored in matrix format, which gives *each example* exactly the *same features*.

# Understanding data (3/4)

- *Unstructured* datasets usually require a **transformation** of the input data to a *structured* form.
- A dataset in matrix format describing automobiles for sale:

features

| year | model | price | mileage | color | transmission |
|---|---|---|---|---|---|
| 2011 | SEL | 21992 | 7413 | Yellow | AUTO |
| 2011 | SEL | 20995 | 10926 | Gray | AUTO |
| 2011 | SEL | 19995 | 7351 | Silver | AUTO |
| 2011 | SEL | 17809 | 11613 | Gray | AUTO |
| 2012 | SE | 17500 | 8367 | White | MANUAL |
| 2010 | SEL | 17495 | 25125 | Silver | AUTO |
| 2011 | SEL | 17000 | 27393 | Blue | AUTO |
| 2010 | SEL | 16995 | 21026 | Silver | AUTO |
| 2011 | SES | 16995 | 32655 | Silver | AUTO |

examples

# Understanding data (4/4)

- *Features* can be recorded using *different* **types of values**:
  - numeric: measured in numbers,
  - categorical – nominal: consists of unordered categories,
  - categorical – ordinal: consists of ordered categories.
- A target feature is a feature, which is a function of (i.e. predicted by) the other features, called input features.
- Analogy with the *dependent variable* and *explanatory variables* in econometrics comes naturally.
- Labeled data is a dataset that includes *both* the *input features and* their corresponding *target values* or *labels*.
- Unlabeled data is a dataset that contains *only* the *input features without* any associated *target values* or *labels*.

# Learning approaches and tasks

- We distinguish between these **types of ML approaches**:
  - predictive models, where the target feature is being *predicted* by the other features (*labeled* data);
  - descriptive models, which *summarize* the data in new ways (*unlabeled* data);
  - meta-learning algorithms, which are *not* tied to a *specific learning task*, but focus on *learning how to learn*.
- Further, we distinguish among *four* **key learning tasks**:
  - classification,
  - numeric prediction,
  - pattern detection,
  - clustering.

# ML approaches: Predictive models

- Training a **predictive model** is known as **supervised learning**, because of the *clear task* to be accomplished.
- The learning algorithm *optimizes a function* to find the feature values that produce the target.
- Prediction can be done for the **target feature** being:
  - a categorical feature, known as the **class**, which is divided into categories called **levels** (may or may not be ordinal);
  - a numeric feature.
- Predictive models thus cover **two learning tasks**:
  - predicting a *categorical feature* is called classification,
  - whereas predicting a *numeric feature* is called numeric prediction.

# ML approaches: Descriptive models

- Training a **descriptive model** is called **unsupervised learning**, because there is *no specific target* to be learned (no *single* feature is of particular interest).
- Descriptive models are primarily used for these **two tasks**:
  - pattern detection, which is a process of identifying underlying *structures*, *relationships* or *regularities* in the data (used often in data mining within large datasets);
  - clustering, which *groups similar examples together* based on their features, without predefined labels (it divides the dataset *into homogenous groups*).
- **Combining** supervised and unsupervised learning results in *additional categories* of learning.

# ML approaches: Descriptive models

- Namely, unsupervised learning can be used to ***assist* with supervised learning** tasks where labeled data is *unavailable* or *costly to obtain* (labelling e.g. loan defaults, trading patterns, mergers, earnings announcements etc.).
- **Semi-supervised learning** uses a small amount of *labeled data* with unsupervised learning to help *categorize* the large amount of *unlabeled records*, which can then be used directly in a supervised learning model.
- **Self-supervised learning** is a *two-step approach* in which a sophisticated model first attempts to identify *meaningful groupings* among records, and the second model attempts to identify the *key distinctions* between the groups.

# ML approaches: Meta-learning algorithms

- Here, the *results* from one or more *previous attempts* to learn a task are used to inform *additional learning*.
- They are useful for **challenging problems** or when **optimal performance** is required.
- Meta learning encompasses algorithms or learners that learn to *work together in teams* called ensembles.
- Meta learning is performed through:
  - **reinforcement learning**, which uses **simulations** that *reward* the learner for success or *punish* the learner for failure, and **iterate repeatedly** to find the *highest cumulative reward* (much like evolution);
  - **adversarial learning**, where models are trained to be robust against **adversarial examples** (inputs designed to deceive the model) by either *generating* such examples during training or *creating defenses* against them.

# Most often used learning algorithms

| Model | Learning task |
|---|---|
| **Supervised learning algorithms** | |
| k-nearest neighbors | Classification |
| Naive Bayes | Classification |
| Decision trees | Classification |
| Classification rule learners | Classification |
| Linear regression | Numeric prediction |
| Regression trees | Numeric prediction |
| Model trees | Numeric prediction |
| Logistic regression | Classification |
| Neural networks | Dual use |
| Support vector machines | Dual use |
| **Unsupervised learning algorithms** | |
| Association rules | Pattern detection |
| k-means clustering | Clustering |
| **Meta-learning algorithms** | |
| Bagging | Dual use |
| Boosting | Dual use |
| Random forests | Dual use |
| Gradient boosting | Dual use |

The *task* will drive the choice of *algorithm*.

University of Ljubljana

SEB SCHOOL OF ECONOMICS AND BUSINESS

# 11.2 Classification Using Decision Trees

EFMD EQUIS ACCREDITED

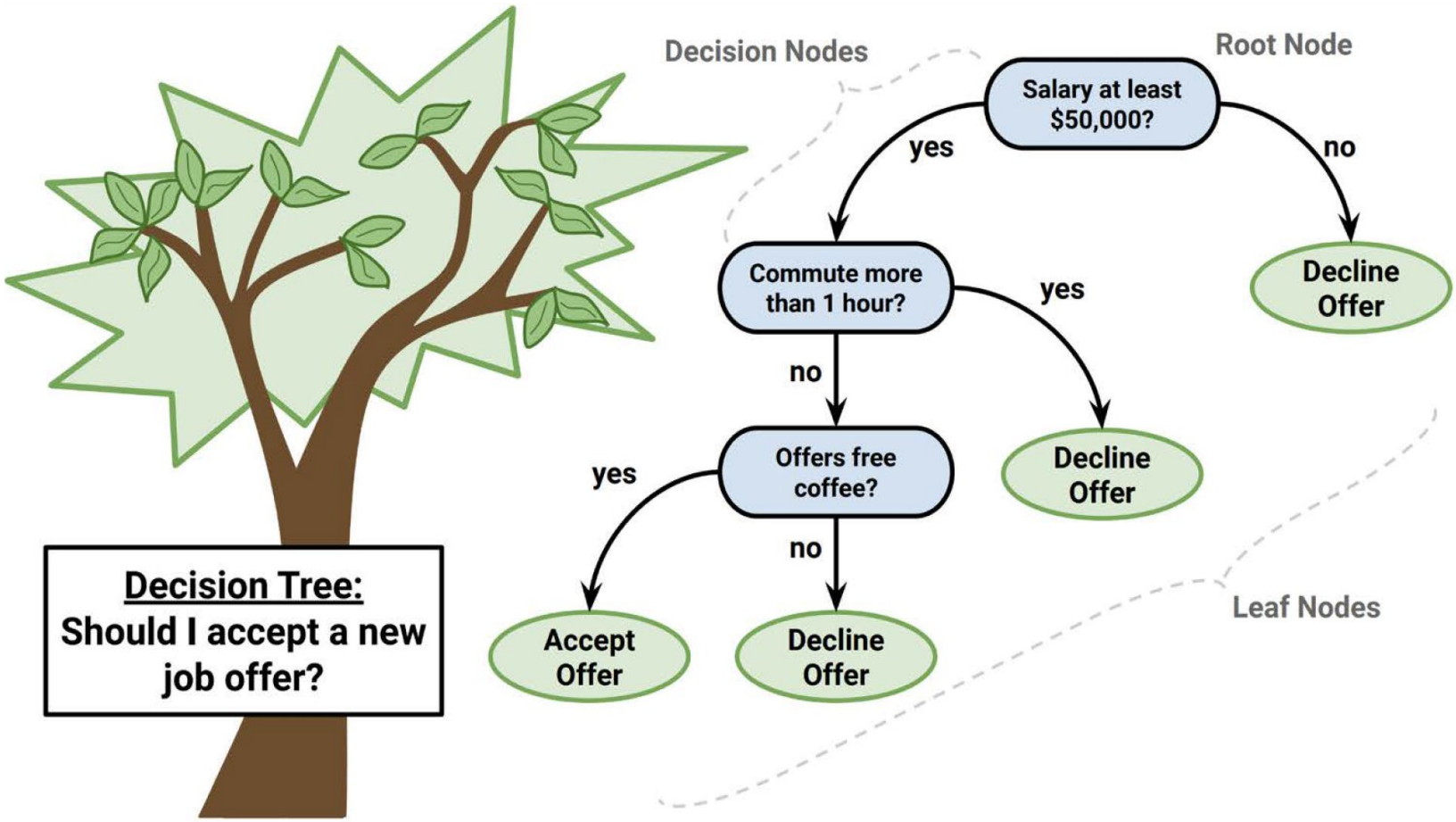
AACSB ACCREDITED

ASSOCIATION AMBA ACCREDITED

# Basic concepts

- *Complex* decisions can often be reduced to a *series* of simpler *if-else statements*.
- Classification using decision trees is one of the machine learning approaches that makes *complex* decisions from *sets* of simple choices.
- **Decision tree learners** utilize a **tree structure** to model the relationships between the *features* and the *outcome*.
- Decision trees are composed of ***nodes*** and ***branches***.
- Beginning at the root node, data flows through decision nodes that decide according to the data's attributes (root node is thus *also* a decision node).
- Branches indicate how the decisions split the data, whereas leaf nodes or terminal nodes denote the final choices.

# Basic concepts



# Decision trees

- Decision trees thus utilize *training data* to learn how to **represent a model as a tree**.
- When a decision tree is *used for classification*, it is often called classification tree.
- The resulting structure is *readily interpretable*, making it well-suited to tasks where the model should be **well-understood** to inform future practices or **transparent** for legal reasons.
- Decision trees exhibit ***excellent*** out-of-the-box ***performance*** and are ***easy to use*** for *almost* any task. It is probably the single *most widely used* machine learning technique.

# Decision trees

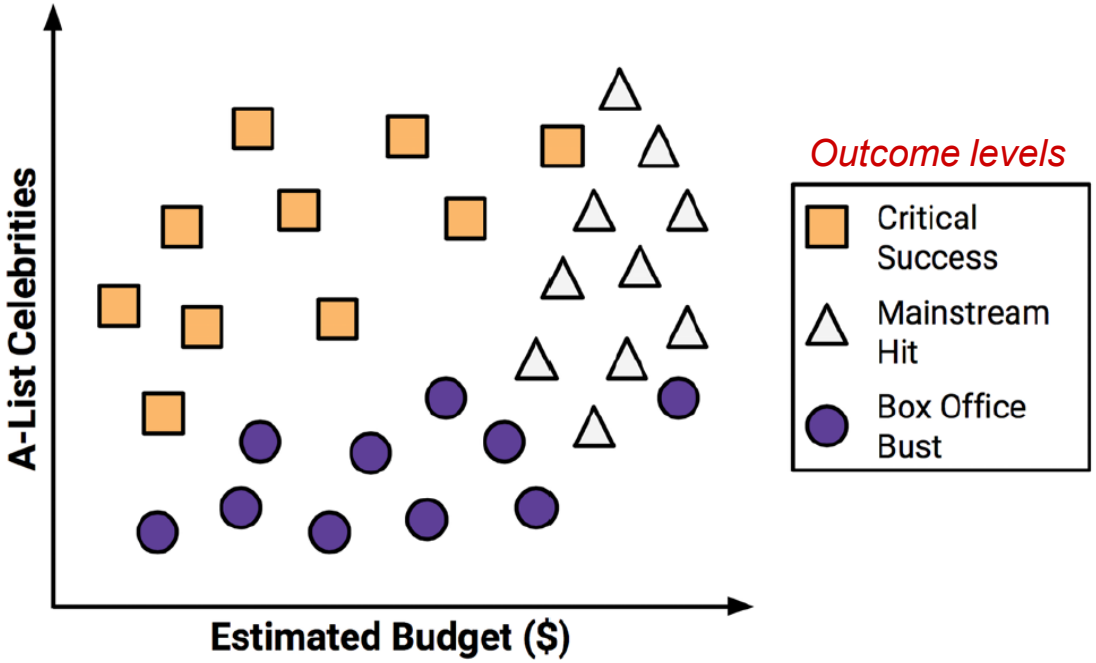
- Some typical **successful applications** include:
  - *credit scoring models*, in which the criteria that cause an applicant to be rejected need to be clearly documented and free from bias;
  - *marketing studies of customer behavior*, such as satisfaction or churn, which will be shared with management or advertising agencies;
  - *diagnosis of medical conditions* based on laboratory measurements, symptoms, or rates of disease progression.
- However, decision trees are *less* ideal for data with ***categorical* features with *many levels*** or data with ***many numeric features***.

# Decision trees

- Decision trees use a **recursive partitioning heuristic** (strategy) that ***splits the data into*** smaller-and-smaller ***subsets*** (also known as the **"divide and conquer" heuristic**).
- The heuristic *repeatedly* chooses a feature to split on, then *partitions* data by its *levels* (values of a feature).
- Partitioning **stops** when the subsets are sufficiently homogenous or a stopping criterion has been met:
  - all (or nearly all) examples have the same class;
  - there are no remaining features that distinguish examples;
  - the tree has hit a predefined size limit.
- Let us demonstrate this ML strategy with an example by ignoring the mathematics employed for the time being.

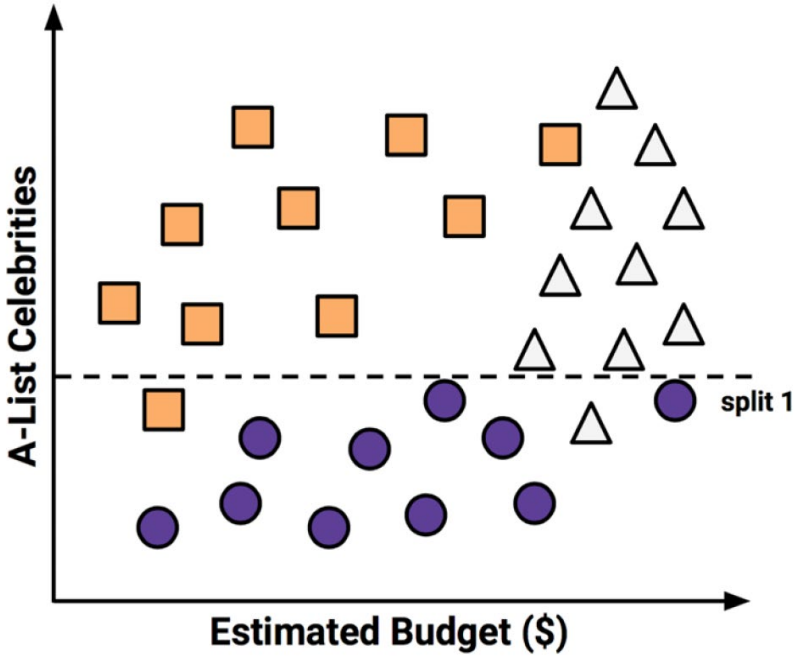
# The movie company example

We examine the factors leading to the success or failure of a movie company's 30 most recent releases. Success is thus the **outcome**, and we identify **two features**: estimated budget and the number of A-list celebrities lined up for starring roles.



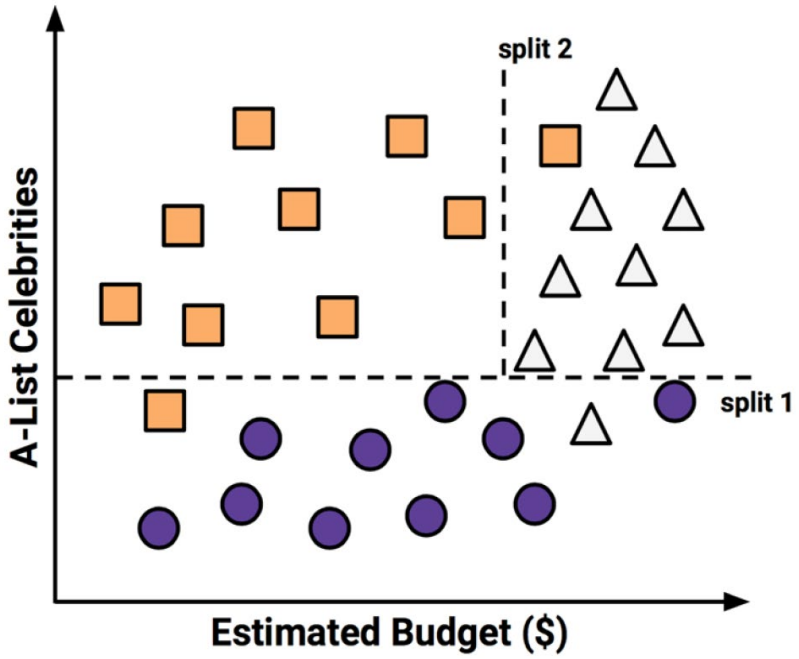
# The movie company example

To create the tree's **root node**, we split the feature indicating the number of celebrities, partitioning the movies into groups; with and without a significant number of A-list stars.

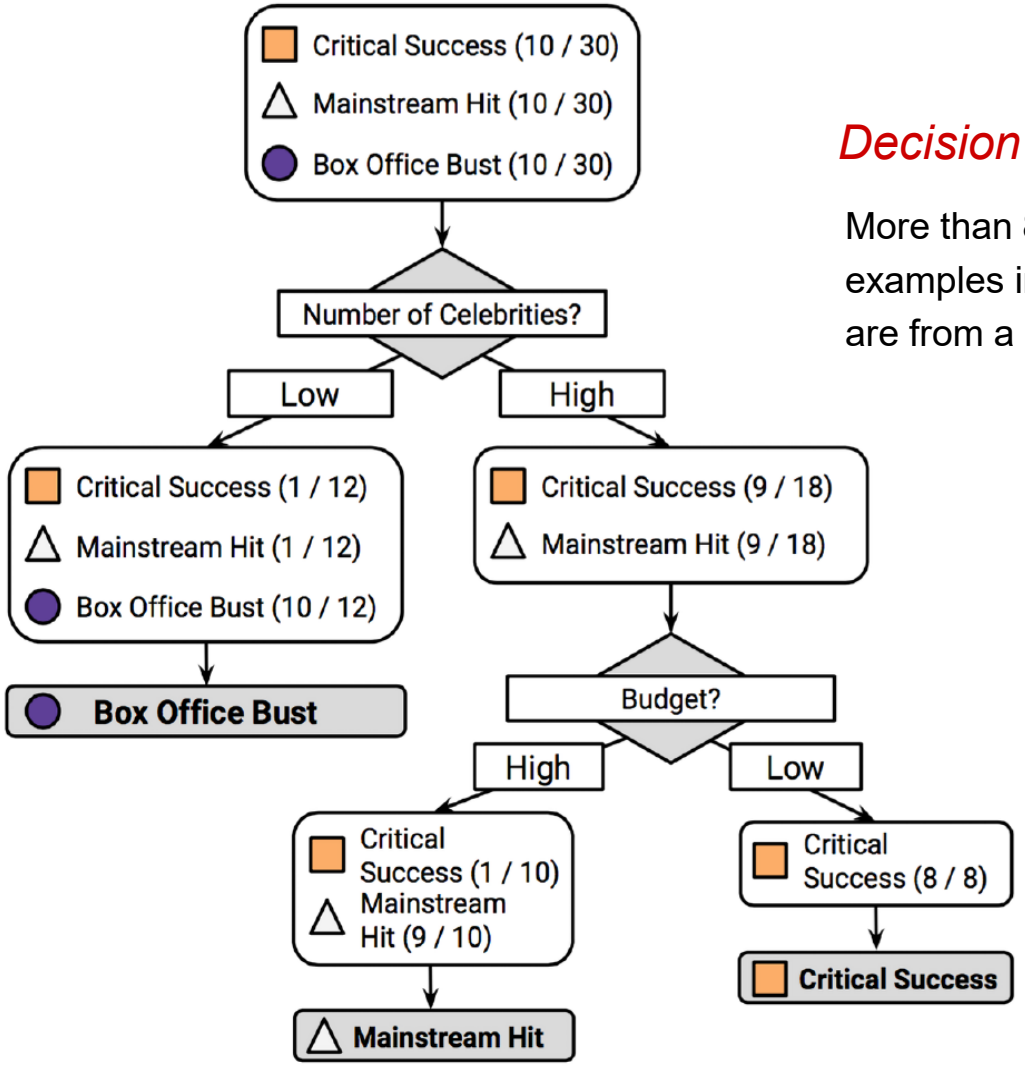


# The movie company example

Next, among the group of movies with a larger number of celebrities, we make another split (**decision node**) between movies with and without a high budget. We could **stop** here.



# The movie company example




## *Decision tree*

More than 80 percent of the examples in each group are from a single class.

# Decision trees

- We stop splitting when the partitions are "sufficiently" homogenous, as we do ***not*** want to **overfit the data**.
- Namely, overly specific decisions often do ***not*** **generalize** more broadly.
- Notice that standard decision trees do *not* use *diagonal or curved lines*, which may have better represented the data.
- In fact, standard decision trees use axis-parallel splits, which consider ***only*** **one feature at a time**.
- Diagonal lines would require **combinations of features** (e.g., instead of $F_2 \leq 3$, we would have $-2F_1 + 5F_2 \leq 10$).

# The C5.0 decision tree algorithm

- There are *numerous implementations* of decision trees, but one of the most well known is the **C5.0 algorithm**.
- It has become the **industry standard** for producing decision trees, because it does well for ***most* types of problems** directly "out of the box".
- This means that compared to other advanced machine learning models, such as *neural networks* or *support vector machines*, the decision trees built by C5.0 generally perform ***nearly* as well**, but are ***much* easier to understand and deploy**, i.e. there is very little to adjust or configure.
- Let us briefly summarize its *strengths* and *weaknesses*.

# The C5.0 decision tree algorithm

| strengths | weaknesses |
|---|---|
| all-purpose classifier that does well on most problems | easy to overfit or underfit the model |
| highly automatic; can handle numeric, nominal, or missing data | axis-parallel splits limit ability to model some relationships |
| ignores unimportant features | biased toward splits on features having a large number of levels |
| can be used on both small and large datasets | large trees are difficult to interpret; splits may seem counterintuitive |
| interpretable without a statistics background (for small trees) | small changes to training data can result in large changes to tree |
| more computationally efficient than other complex models | |

# Identifying the best split

- The first challenge that a decision tree will face is to identify ***which feature* to split upon**.
- The degree to which a subset of examples contains only a single class is known as **purity**.
- Any subset composed of only a *single class* is called pure.
- There are various **measurements of purity**: entrophy, the Gini index, $\chi^2$–statistic, and the gain ratio.
- C5.0 is based on the concept of **entropy**, which is a purity measure from information theory capturing the randomness or disorder within a set of class values.
- Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality.

# Entrophy

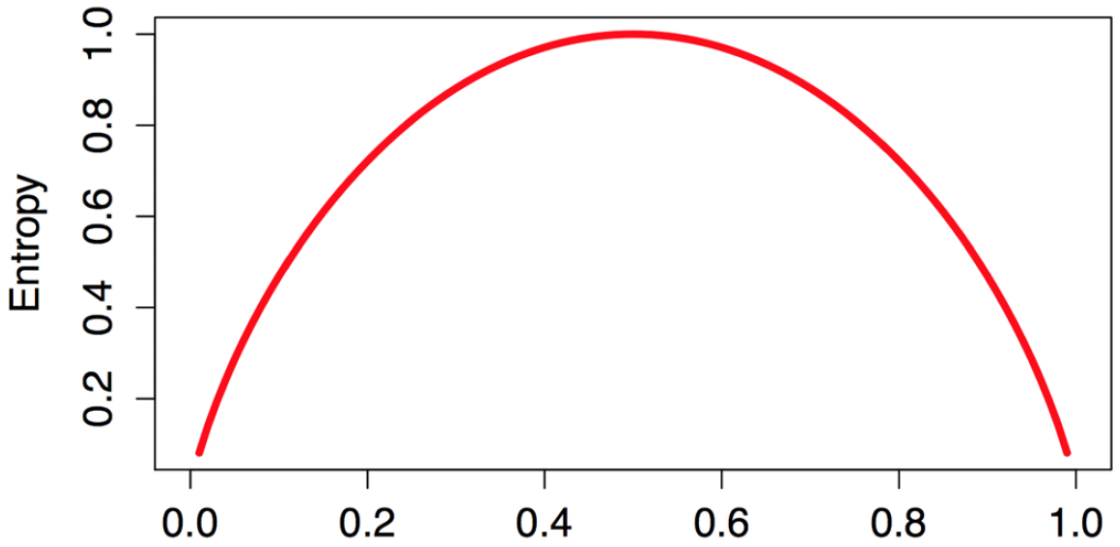
- The decision tree hopes to find splits that ***reduce* entropy**, ultimately ***increasing* homogeneity** within the groups.
- Typically, entropy is **measured** in **bits**:
  - It ranges from 0 to 1 for two class levels;
  - It ranges from 0 to $\log_2(n)$ for $n$ class levels.
- The *minimum* value (0) indicates that the sample is completely **homogenous**, while the *maximum* value indicates that the data are **as diverse as possible**.
- Entropy for a given segment of data $S$ is specified as:

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i log_2(p_i)$$

where $c$ refers to the number of class levels, and $p_i$ refers to the proportion of values falling into the $i$-th class level.

# Entrophy

- We can visualize the entropy for all possible **two-class arrangements**:




- The peak is at $x = 0.50$, as the **50-50 split** results in the ***maximum* entropy**. As one class increasingly dominates the other, the entropy reduces towards zero.

# Information gain

- To determine the optimal feature to split upon, the algorithm calculates the change in homogeneity that would result from a split on each possible feature.
- This measure is known as the **information gain**, and is calculated for a feature *F* as the difference between the entropy in the segment before the split ($S_1$) and the partitions resulting from the split ($S_2$):

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

- As there are multiple partitions in $S_2$, each partition's entropy is weighted by $w_i$, the proportion of examples in the partition:

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

# Information gain

- The *higher* the information gain, the *better* a feature is at creating homogeneous groups after a split on that feature:
  - If the **information gain is zero**, there is no reduction in entropy for splitting on this feature;
  - The **maximum information gain** is equal to the entropy prior to the split, implying that the entropy after the split is zero and we obtain completely homogeneous groups.
- Decision trees can use information gain for splitting on **numeric features** as well.
- This is usually done by reducing the numeric feature into a two-level categorical feature by emplyoing a threshold.

# Pruning the decision tree

- As already mentioned, if the tree grows **overly large**, many of the decisions it makes will be **overly specific** and the model will be **overfitted** to the training data.
- The process of **pruning** a decision tree involves *reducing its size* such that it **generalizes better** to unseen data.
- We distinguish between ***pre-pruning*** and ***post-pruning***.
- **Pre-pruning** or **early stopping** involves ***stopping*** data splitting after a ***number of decisions*** or if nodes contain ***less than a specific number of examples***.
- This *prevents* the tree from doing *needless work*, but may *miss* subtle or *important patterns* it would have discovered later on during the splitting process.

# Pruning the decision tree

- **Post-pruning**, on the other hand, grows an ***overly large tree*** and ***prunes leaves*** to reduce the size.
- In this way, nodes and branches with *minor impact* on classification error are *removed*
- This is often a ***more* effective approach** than pre-pruning because it is quite *difficult* to determine the *optimal depth* of a decision tree *without growing it first*.
- C5.0 uses these two *post-pruning strategies* automatically:
  - Subtree raising, which involves replace a parent node with one of its child branches (subtrees) when that child provides a better or simpler split;
  - Subtree replacement, which involves replacing a branch (subtree) with a leaf when the leaf gives nearly the same predictive accuracy as the full branch.

# Adaptive boosting

- In order to ***improve*** **predictive accuracy**, we can employ **adaptive boosting**, which is especially useful when a *single* decision tree is *too weak* to capture complex patterns.
- The idea is that by combining several **weak-performing learners**, you can create a team, called an ensemble, that is much *stronger* than any of the learners alone.
- The algorithm starts with training data and assigns *equal* weights to all examples.
- It then employs a learner decision tree and *increases* weights of *misclassified* examples so the next tree pays more attention to them.
- It trains the next learner decision tree on the *reweighted data* and repeats the process for *several rounds*.

# Adaptive boosting

- The number of separate decision trees to use in the boosted team is called the number of trials.
- ***Ten* trials** has become the *de facto standard*, as research suggests that this *reduces error rates* on test data *by about 25 percent* for the C5.0 algorithm.
- Some of the trees include subtrees, obtained by **post-pruning strategies**, such as *subtree raising* and *subtree replacement*.
- All weak learners are then combined into a final **strong classifier** by this process of *weighted voting*. The final decision tree is thus a weighted vote of all trees, *emphasizing* the *more accurate* ones.
- Adaptive boosting **improves accuracy** and **reduces bias**, but can be sensitive to *noise* and *outliers*.

# Cost matrix

- Overall predictive accuracy does not always tell the whole story, as *some* types of *mistakes* are *more costly* than others.
- E.g., giving a loan to an applicant who defaults (a **false negative**) can be an expensive mistake, resulting in losses that *outweigh* the interest the bank might earn on risky loans it denies but that would have been repaid (a **false positive**).
- The C5.0 algorithm allows us to *assign* a penalty to *different* types of *errors* in order to **discourage** a tree from making more **costly mistakes**. The penalties are designated in a **cost matrix**, which specifies *how many times* more costly each error is relative to any other.
- The cost matrix is then *applied* to the decision tree learner in order to ***minimize* the costly mistakes**, even at the expense of lower *overall* predictive accuracy.

University of Ljubljana

SEB SCHOOL OF ECONOMICS AND BUSINESS

# 11.3 Classification Using Rule Learners

EFMD EQUIS ACCREDITED

AACSB ACCREDITED

ASSOCIATION AMBA ACCREDITED

# Basic concepts

- **Classification rule learners** represent another machine learning approach that makes *complex* decisions from *sets* of simple choices.
- Similarly to decision trees, they employ logical *if-else statements* that *assign a class* to unlabeled examples.
- They are specified in terms of:
  - an antecedent, which comprises certain combinations of feature values ("if this"), and
  - a consequent, which specifies the class value to assign if the rule's conditions are met ("then this").
- They are *closely related* to **decision tree learners** and often used for *similar types of tasks*. In fact, they can be *generated from decision trees*.

# Basic concepts

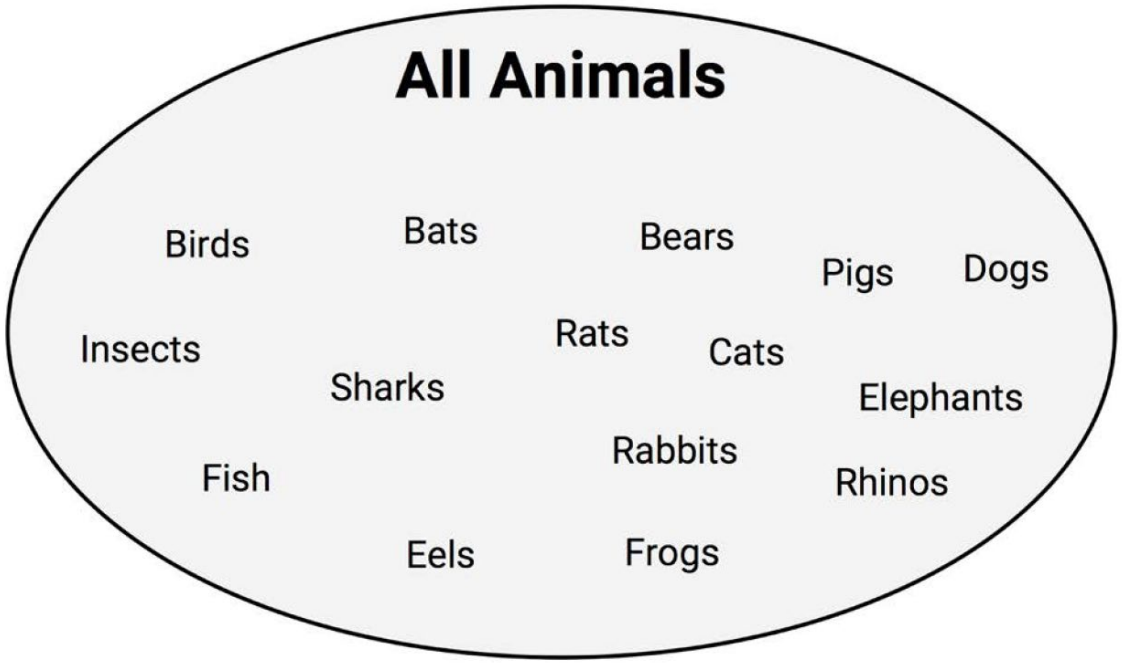
- Some typical **successful applications** include:
  - finding conditions that precede large drops or increases in the *prices of shares on the stock market*;
  - describing the key characteristics of groups of people for *customer segmentation*;
  - identifying conditions that lead to *hardware failure* in mechanical devices.
- Rule learners are generally applied to problems where the **features** are primarily or entirely ***nominal***.
- They do well at identifying **rare events**, even if these occur *only* for a *very specific interaction* among feature values.

# Separate and conquer heuristic

- Classification rule learners use a **"separate and conquer" heuristic** that involves finding rules that cover ***homogeneous* subsets** of data.
- After identifying a rule, homogeneous partitions are *separated* from the remaining data, then the remainder is "conquered" with *increasingly specific* rules until the *entire dataset* has been *covered*.
- As the rules "cover" portions of the data, separate and conquer algorithms are also known as **covering algorithms**, and the resulting rules are called **covering rules**.
- Let us demonstrate this ML strategy with the textbook animal classification example.

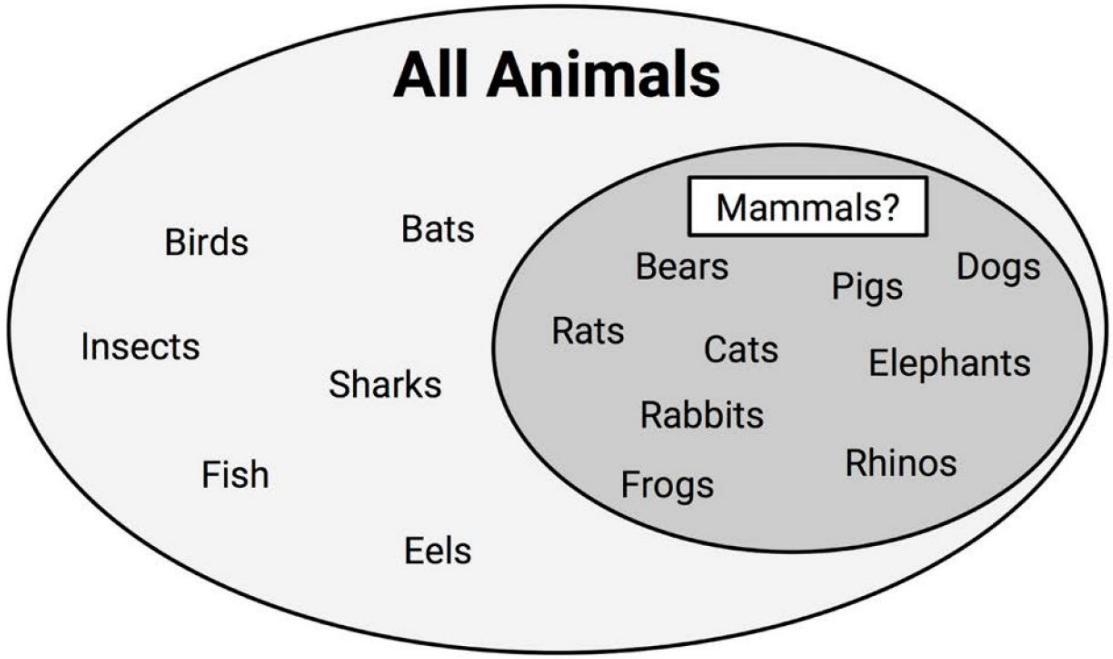
# The animal classification example

We would like to create rules to identify whether or not an animal is a mammal. We have the following set of all animals depicted as a large **space**.



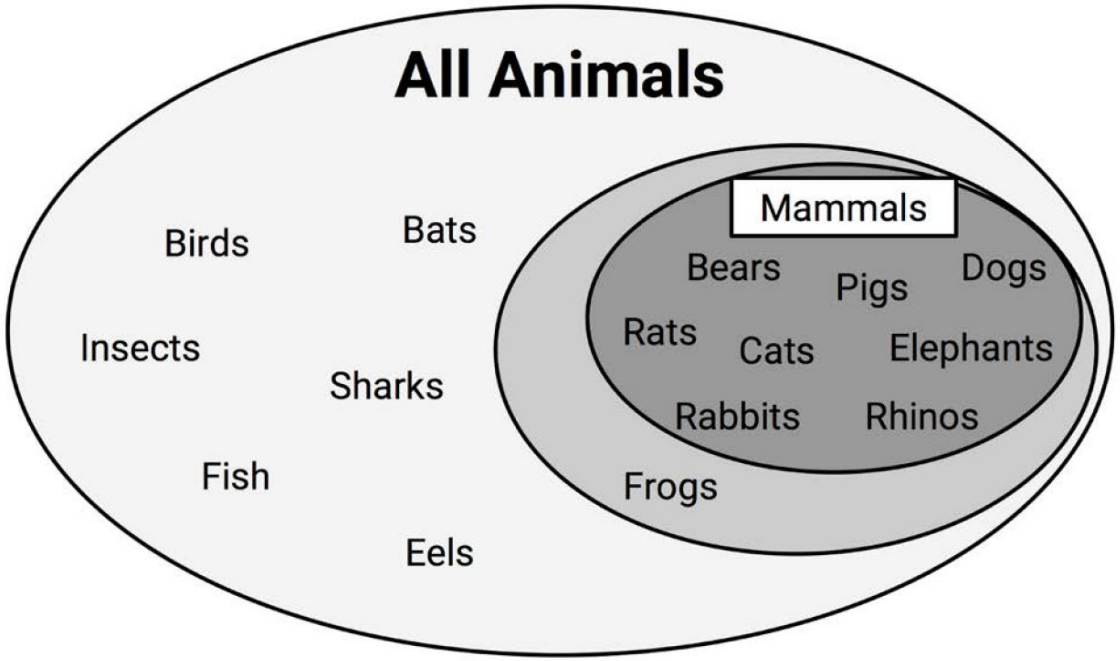
# The animal classification example

A rule learner begins by using the available **features** to find homogeneous groups. E.g., using a feature that indicates whether the species travels via land, sea, or air, the **first rule** might suggest that any land-based animals are mammals.



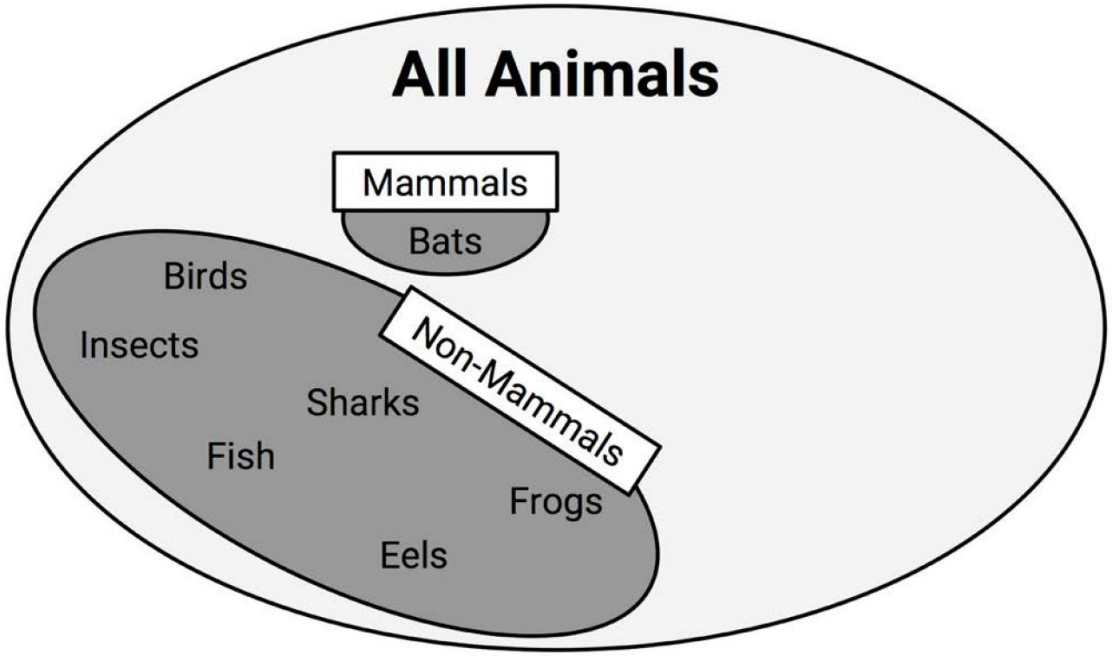
# The animal classification example

This rule resulted in one misclassified example. Let us add a **second rule** stating that an animal needs to have a tail to be a mammal. The resulting **homogeneous** subset of mammals can now be **separated from the other data**.



# The animal classification example

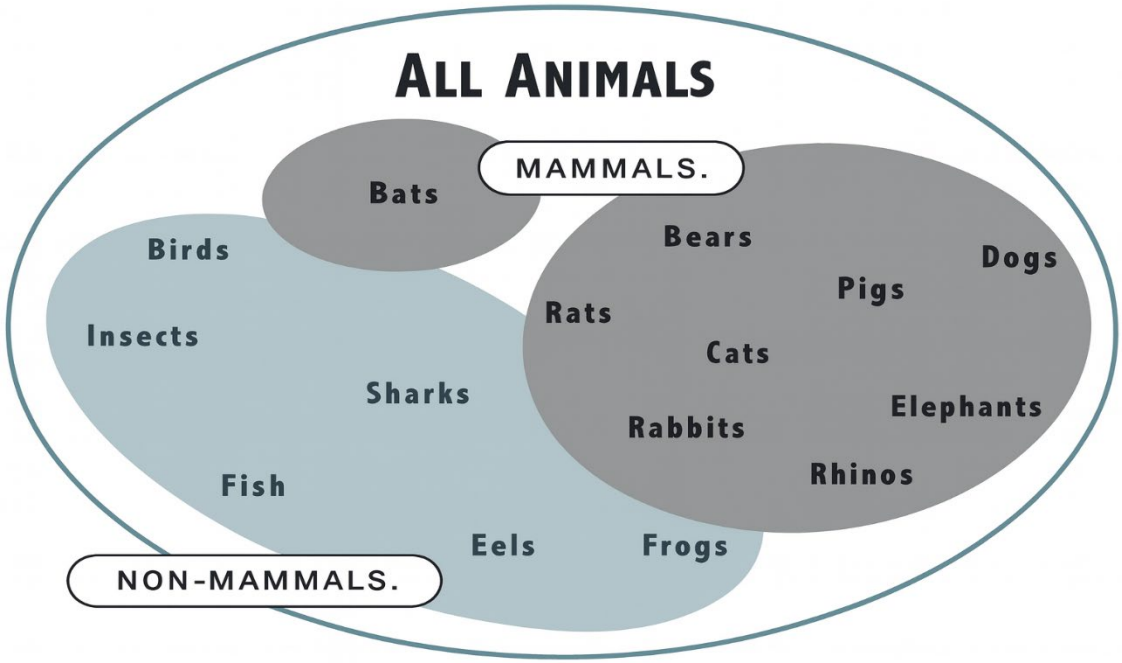
A **third rule** can be defined to separate out the only remaining mammal. A potential feature would be the presence of fur. We have now correctly identified all the examples in our dataset, and the rule learning process would **stop**.



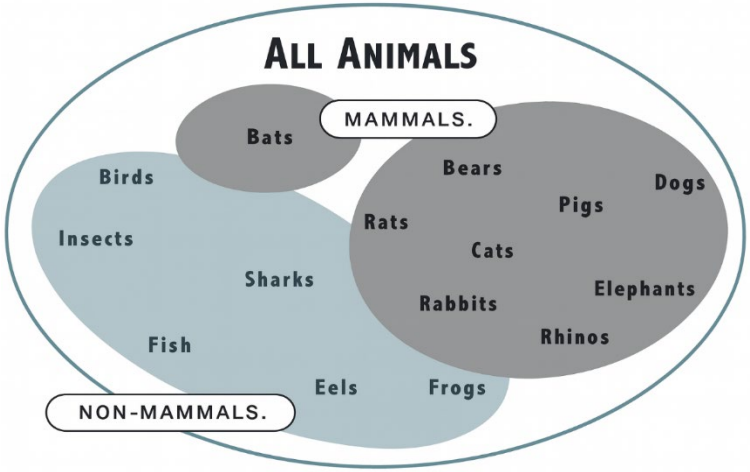
# The animal classification example

We learned a ***total* of three rules**:

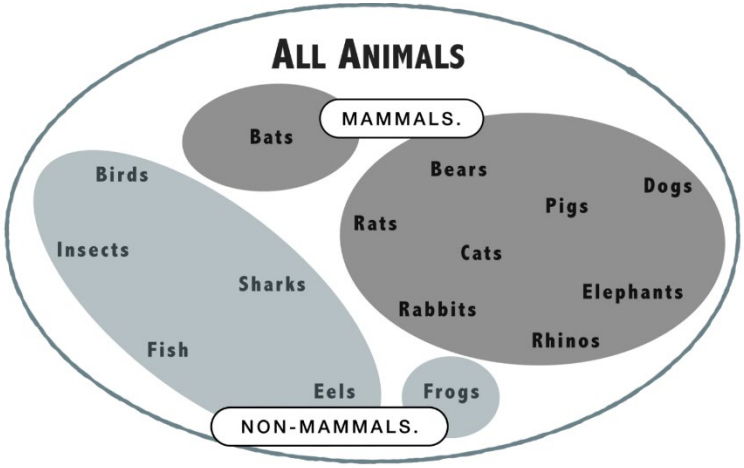
(1) animals that walk on land and have a tail are mammals;
(2) otherwise, if the animal has fur it is a mammal;
(3) otherwise, it is not a mammal.



# The animal classification example




WHICH OF THESE IS SIMPLER?

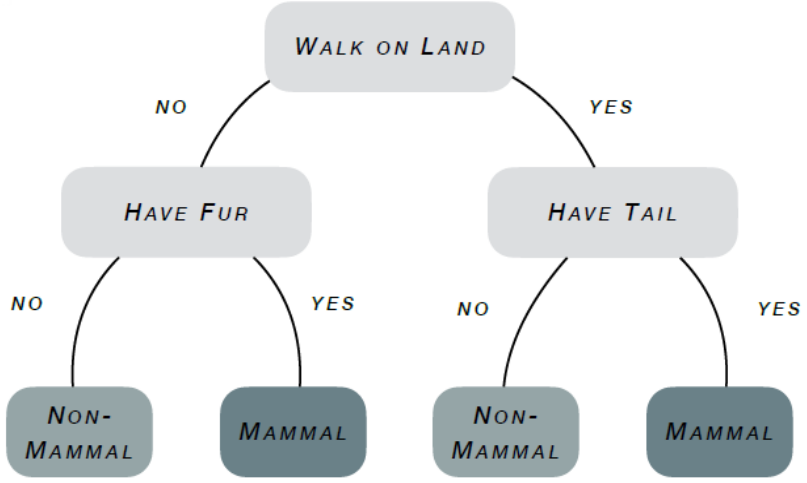



## Rules

(1) Animals that walk on land and have a tail are mammals

(2) otherwise, if the animal has fur it is a mammal

(3) otherwise, it is not a mammal.




## Tree

# Rule learners: ZeroR and OneR

- The *simplest* rule classifier is called **ZeroR**. It is a rule learner that *considers no features* and literally *learns no rules*.
- For every unlabeled example it predicts the **most common class**, regardless of the values of its features.
- It provides a simple **baseline for comparison** to other, more sophisticated rule learners.
- The **OneR** algorithm (also known as **One Rule** or **1R**) *improves* over ZeroR by generating a *single*, easy-to-understand, human-readable *rule*.
- Using a *single feature*, it exhibits surprisingly **decent performance** on many real-world classification problems.
- **How does it work?** For each feature, OneR *divides* the data into *groups* with *similar values* of the *feature*. Then, for each segment, the algorithm predicts the *majority class*.

# Rule learners: ZeroR and OneR

- The **error rate** for the rule based on *each feature* is then calculated and the **rule with the *fewest* errors is chosen** as the one (single) rule.

| Animal | Travels By | Has Fur | Mammal |
|---|---|---|---|
| Bats | Air | Yes | Yes |
| Bears | Land | Yes | Yes |
| Birds | Air | No | No |
| Cats | Land | Yes | Yes |
| Dogs | Land | Yes | Yes |
| Eels | Sea | No | No |
| Elephants | Land | No | Yes |
| Fish | Sea | No | No |
| Frogs | Land | No | No |
| Insects | Air | No | No |
| Pigs | Land | No | Yes |
| Rabbits | Land | Yes | Yes |
| Rats | Land | Yes | Yes |
| Rhinos | Land | No | Yes |
| Sharks | Sea | No | No |

**Full Dataset**

Feature 1:

| Travels By | Predicted | Mammal | |
|---|---|---|---|
| Air | No | Yes | ✖ |
| Air | No | No | |
| Air | No | No | |
| Land | Yes | Yes | |
| Land | Yes | Yes | |
| Land | Yes | Yes | |
| Land | Yes | Yes | |
| Land | Yes | No | ✖ |
| Land | Yes | Yes | |
| Land | Yes | Yes | |
| Land | Yes | Yes | |
| Land | Yes | Yes | |
| Sea | No | No | |
| Sea | No | No | |
| Sea | No | No | |

**Rule for "Travels By"**
**Error Rate = 2 / 15**

**Chosen.**

Feature 2:

| Has Fur | Predicted | Mammal | |
|---|---|---|---|
| No | No | No | |
| No | No | No | |
| No | No | Yes | ✖ |
| No | No | No | |
| No | No | No | |
| No | No | No | |
| No | No | Yes | ✖ |
| No | No | Yes | ✖ |
| No | No | No | |
| Yes | Yes | Yes | |
| Yes | Yes | Yes | |
| Yes | Yes | Yes | |
| Yes | Yes | Yes | |
| Yes | Yes | Yes | |
| Yes | Yes | Yes | |

**Rule for "Has Fur"**
**Error Rate = 3 / 15**

# Rule learners: RIPPER

- Early rule learners were notorious for being *slow* and *inaccurate on noisy data*.
- This was improved in 1994 by the **incremental reduced error pruning (IREP)** algorithm.
- It uses a combination of *pre-pruning* and *post-pruning* methods that **grow very complex rules** and **prune** them before *separating* the examples from the full dataset.
- Although this strategy **improved performance**, decision trees often still performed better.
- The performance and robustness were further improved in 1995 by the **repeated incremental pruning to produce error reduction (RIPPER)** algorithm.

# Rule learners: RIPPER

- The RIPPER algorithm can consider *more than one feature* (allowing for *multiple antecedents*) and can thus create much *more complex rules* than the OneR algorithm.
- Let us briefly summarize its *strengths* and *weaknesses*.

| strengths | weaknesses |
|---|---|
| generates easy-to-understand, human-readable rules | may result in rules that seem to defy common sense |
| efficient on large and noisy datasets | not ideal for numeric data |
| generally produces a simpler model than a decision tree | might not perform as well as more complex models |

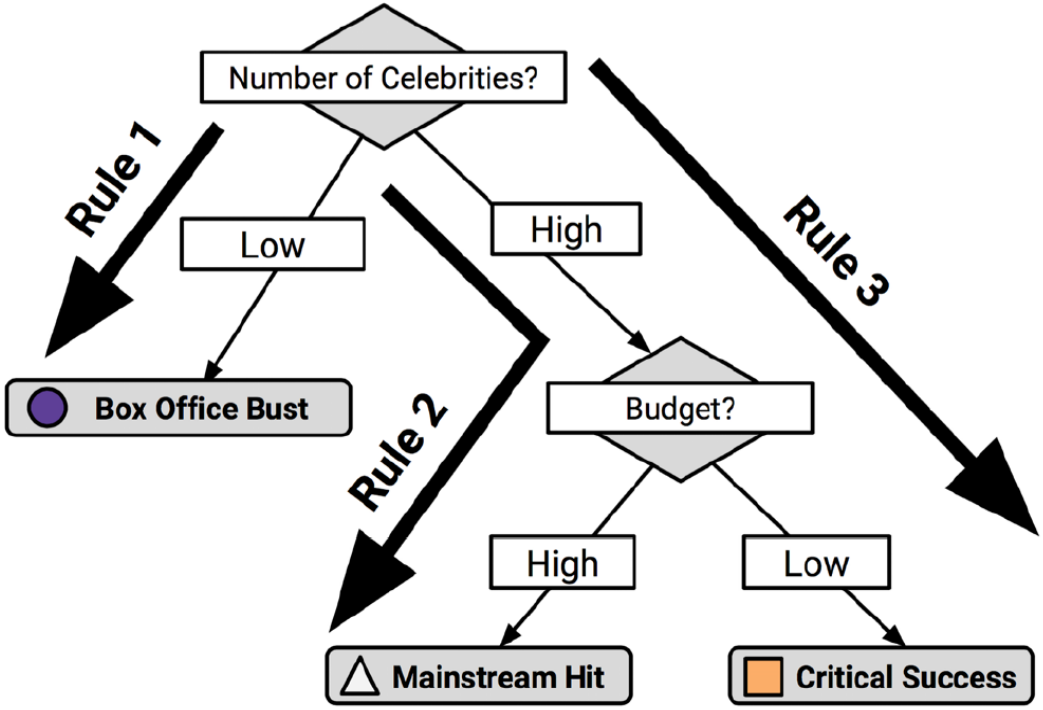
**key strength**

# Rule learners: RIPPER

- The RIPPER algorithm itself is *complex*, but it can be understood in general terms as a **three-step process**:
  - *grow*: this step employs the separate-and-conquer technique using information gain to create rules;
  - *prune*: when increasing rule specificity no longer reduces entropy, it is pruned;
  - *optimize*: after a stopping condition is met, the entire rule set is optimized by a variety of heuristics.
- It is able to model more *complex data*, but similarly to decision trees, the rules can quickly become *difficult to comprehend*.
- The **evolution** of classification rule learners has **continued** with algorithms such as IREP++, SLIPPER and TRIPPER.

# Rules from decision trees

- Rules can also be *constructed* directly *from decision trees*.
- Beginning at the *root node*, we follow *branches* to each *leaf node*, obtaining a **series of decisions** that can be combined into a *single rule*.



# Complexity versus efficiency

- However, “divide and conquer” based *decision trees* often produce **more *complex* rules** than “separate and conquer” based *rule learners*.
- Namely, once the ***divide-and-conquer heuristic*** splits on a feature, the partitions created by the split may *not* be re-conquered, *only* further subdivided. A decision tree is thus *permanently limited by its history* of past decisions.
- In contrast, once the ***separate-and-conquer heuristic*** finds a rule, any examples not covered by all the rule’s conditions may be *re-conquered*.
- Consequently, rule learners are often more *parsimonious* (less complex), whereas decision trees are often **more *computationally efficient***, as they do *not reuse* the data.

# What makes trees and rules "greedy"?

- Decision trees *and* rule learners are known as greedy learners, as they use data on a "first-come, first-served" basis.
- They are **short-sighted**: at each step of the algorithm, they grab the *currently* best-looking split or rule, without looking ahead to see if a different choice might lead to a better overall tree or rule set *later*.
- Consequently, greedy algorithms are ***not*** guaranteed to generate the **optimal, most accurate or smallest number of rules** for a particular dataset.
- However, ***without*** using the greedy approach to rule learning, it is likely that for all but the smallest of datasets, rule learning would be **computationally infeasible**.

University of Ljubljana

SEB SCHOOL OF ECONOMICS AND BUSINESS

# 11. Machine Learning

*Prof. Dr. Miroslav Verbič*

miroslav.verbic@ef.uni-lj.si

www.miroslav-verbic.si

EFMD EQUIS ACCREDITED

AACSB ACCREDITED

ASSOCIATION AMBA ACCREDITED

Ljubljana, October 2025