Data Mining

Classification: Part 2

Mohammed Brahimi & Sami Belkacem

Outline

- ❏ Characteristics of Decision Trees
- ❏ Decision Trees vs. Other models
- ❏ Model Evaluation
- ❏ Model Diagnosis

Outline

- ❏ **Characteristics of Decision Trees**
- ❏ Decision Trees vs. Other models
- ❏ Model Evaluation
- ❏ Model Diagnosis

Characteristics of Decision Trees

- **Nonparametric Approach**: No prior assumptions on data's probability distribution.
- **Wide Applicability**: Can be applied for binary, categorical, and continuous data.
- **No Data Transformation**: Attributes can be used without normalization or standardization.
- **Multiclass Problem Handling**: Handel multiclass without reducing them to binary tasks.
- **Interpretability**: Trained trees are easy to understand (particularly shorter ones).
- **Competitive Accuracy**: The result is comparable with other algorithms for simple datasets.

Characteristics of Decision Trees - Expressiveness

- **Universal Representation**
  - Decision trees can encode any function of discrete-valued attributes.
- **Efficient Encoding**
  - Discrete-valued function can be represented as an assignment table.
  - Decision tree can represent the assignment table efficiently.
  - Decision tree can group a combinations of attributes as leaf nodes.
- **Limitations**
  - Some functions, like the parity function, require a full decision tree for accurate modeling.

A	B	C	D	class
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

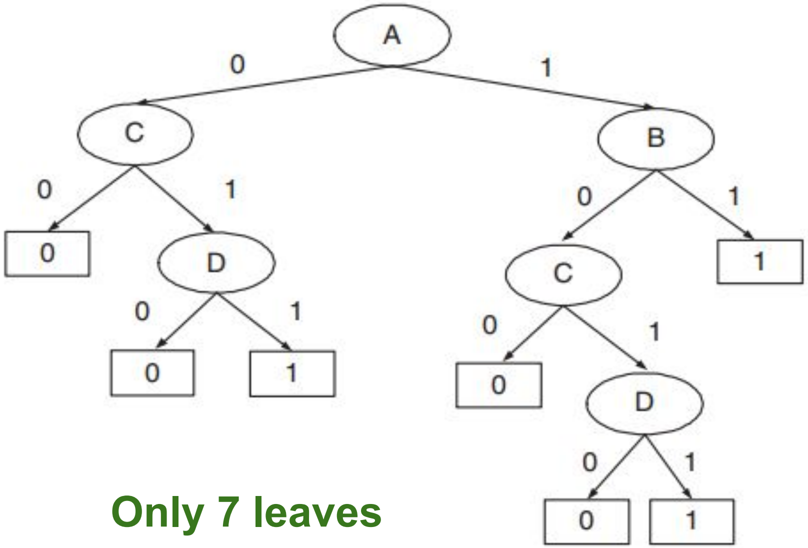
Example of Compact Representation

Boolean function using a simpler tree with fewer leaf nodes, instead of a fully-grown tree:

A	B	C	D	class
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

16 entries

$$(A \wedge B) \vee (C \wedge D)$$



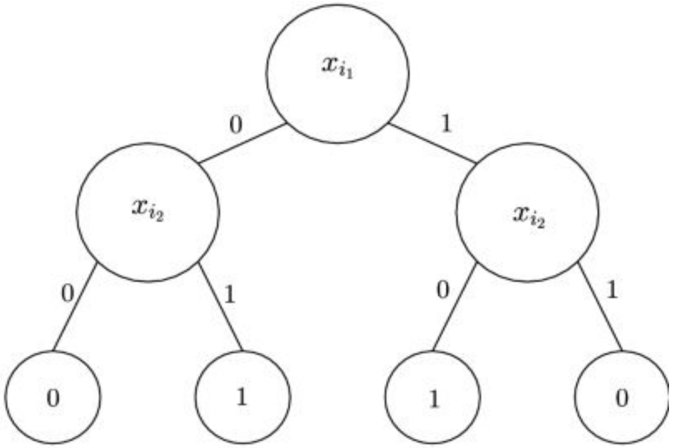
Only 7 leaves

Example of Parity Representation

Parity representation adds an extra bit to binary data to ensure an even number of 1s.

x_{i1}	x_{i2}	**Parity**
0	0	0
0	1	1
1	0	1
1	1	0

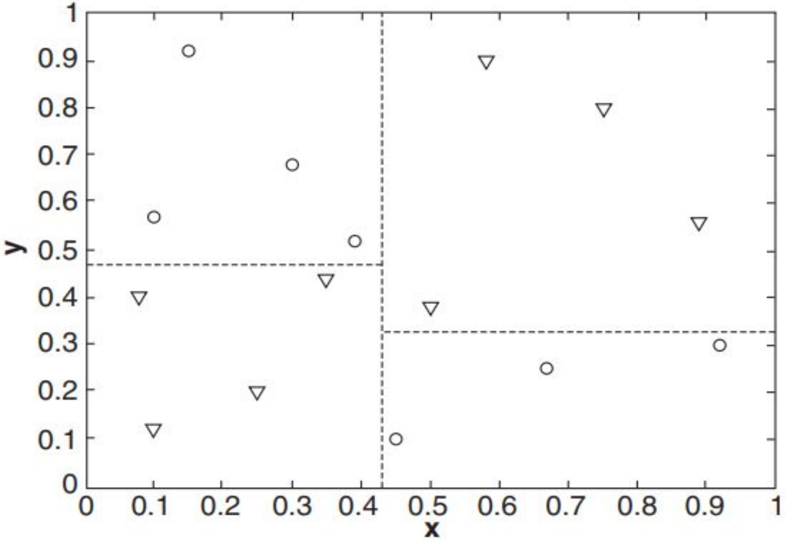
4 entries

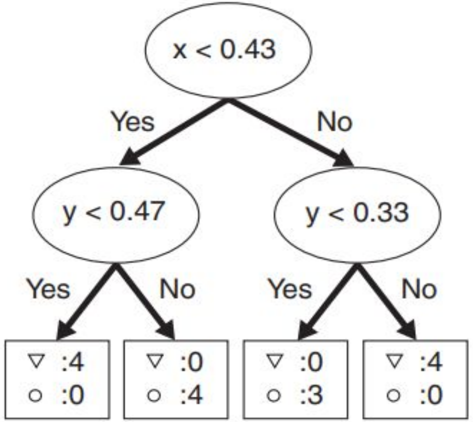


Corresponding Deterministic Decision Tree

4 leaves

Characteristics of Decision Trees - Rectilinear Splits



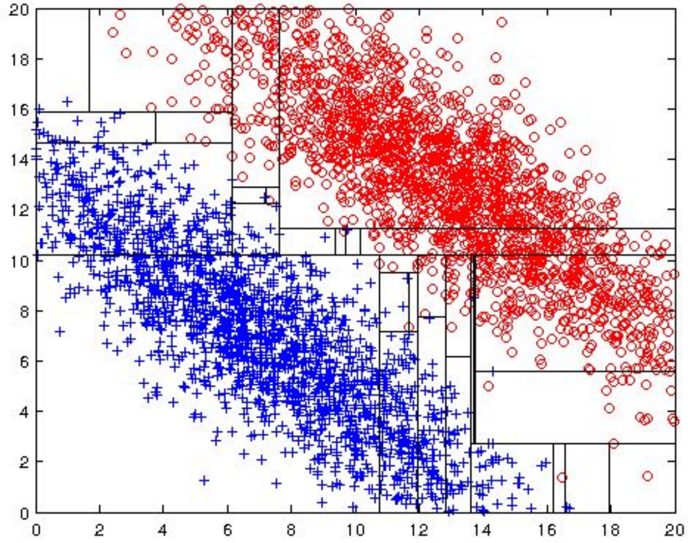


Two features ***x*** and ***y*** and two classes

- Decision Trees use rectilinear splits to divide the data space.
- Simplifies complex multidimensional data into understandable segments.
- Effective in handling both categorical and continuous variables.

Disadvantages of Rectilinear Splits

- **Struggle with Non-linear Boundaries:**
  - Ineffective in capturing complex, non-linear relationships in data.
- **Limited Flexibility:**
  - Restricts decision boundaries to orthogonal lines, limiting flexibility.
- **Oversimplification Risks:**
  - Can lead to oversimplified models that fail to capture the true nature of the data.



Two features ***x*** and ***y*** and two classes (red and blue)

Outline

- ❑ Characteristics of Decision Trees
- ❑ **Decision Trees vs. Other models**
- ❑ Model Evaluation
- ❑ Model Diagnosis

Decision Trees vs. Other models

No Free Lunch Theorem:

- **No Universal Best Algorithm**

  There is no single "best" algorithm for predictive modeling (classification and regression).
- **Advantages & Disadvantages of each Algorithm**

  Algorithms vary in training/prediction time, feature tolerance, data requirements, hyperparameters, …
- **Problem-Specific Algorithm Selection**

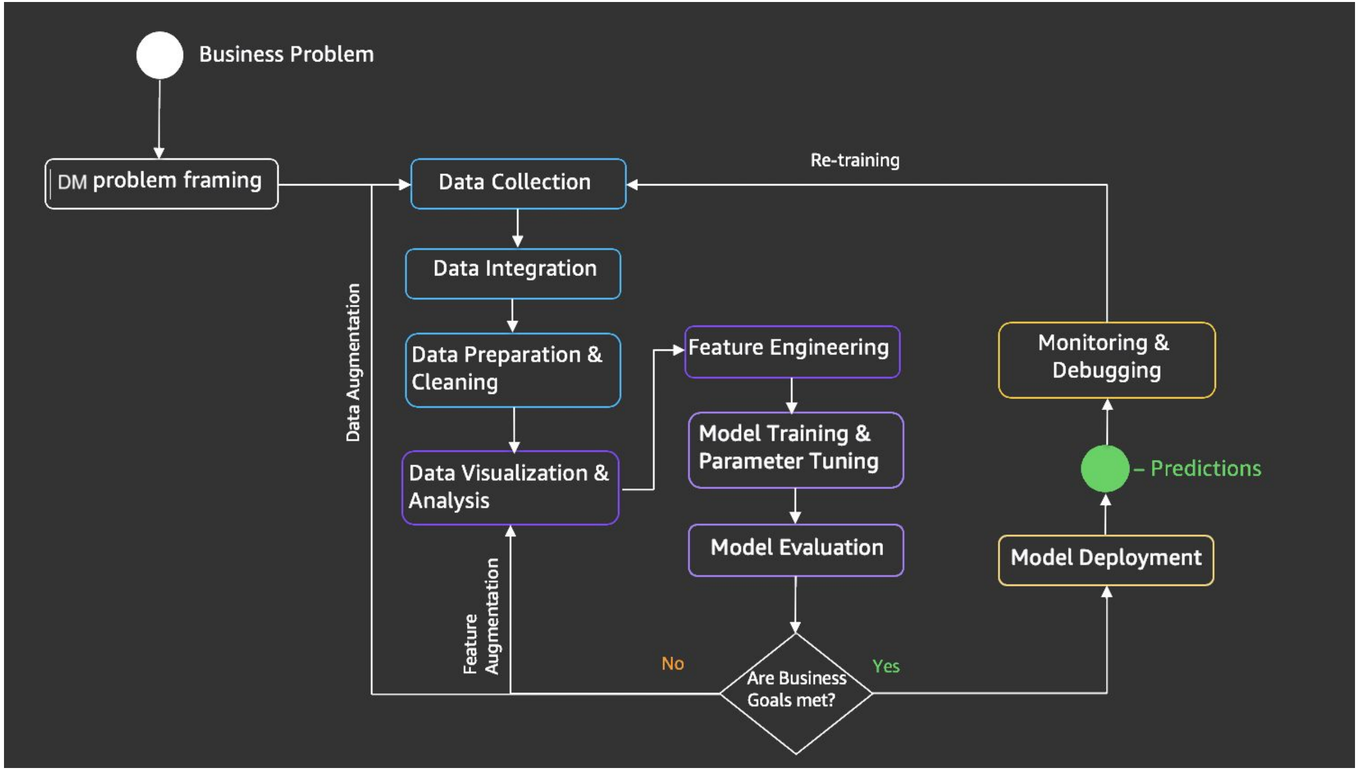
  Choose a model based on the problem type (classification, regression), number of features, data size, …
- **Experimentation & Validation**

  Often necessary to try different algorithms and validate them to identify the best model.

Outline

- ❏ Characteristics of Decision Trees
- ❏ Decision Trees vs. Other models
- ❏ **Model Evaluation**
- ❏ Model Diagnosis

From Data collection to Model Training and Evaluation

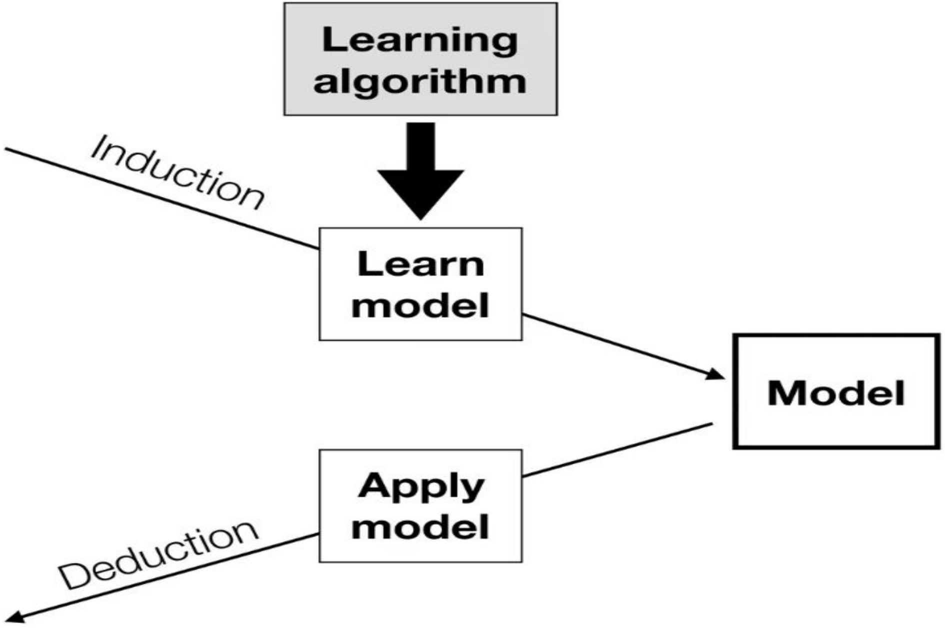


Note: Decision tree hyperparameters include max depth, min samples split, split criterion (Gini, Entropy), …

Model Training and Evaluation

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Model Evaluation

Objective: After training the model, estimate its performance on new unseen data.

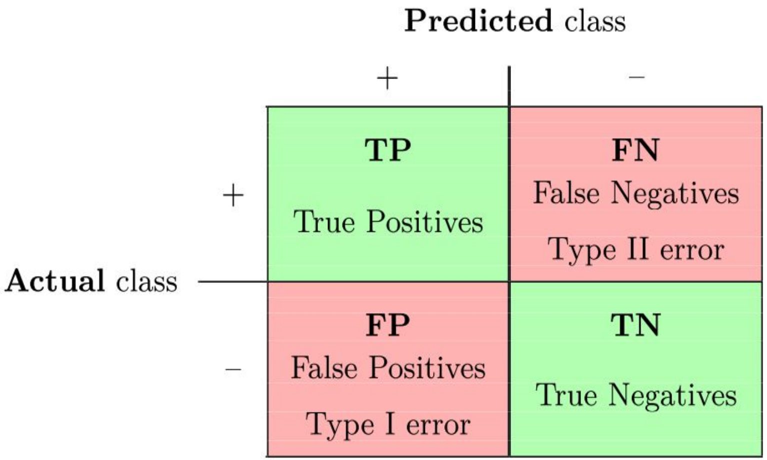
1. **Defining Evaluation Metrics**
   - **Classification Metrics:** Confusion matrix, Accuracy, Precision, Recall, F1 Score, etc.
   - **Regression Metrics:** Mean Squared Error (MSE), Mean Absolute Error (MAE), etc.

2. **Choosing a Data Splitting Strategy**
   - **Holdout:** A single division of data, reserving a portion for testing.
   - **Cross-Validation:** Repeated splits for a robust performance estimate.
   - **Stratified Sampling:** Ensures class balance in each split, especially for imbalanced data.

Evaluation Metrics for Classification

Confusion Matrix

Provides a complete view of model performance in binary classification.



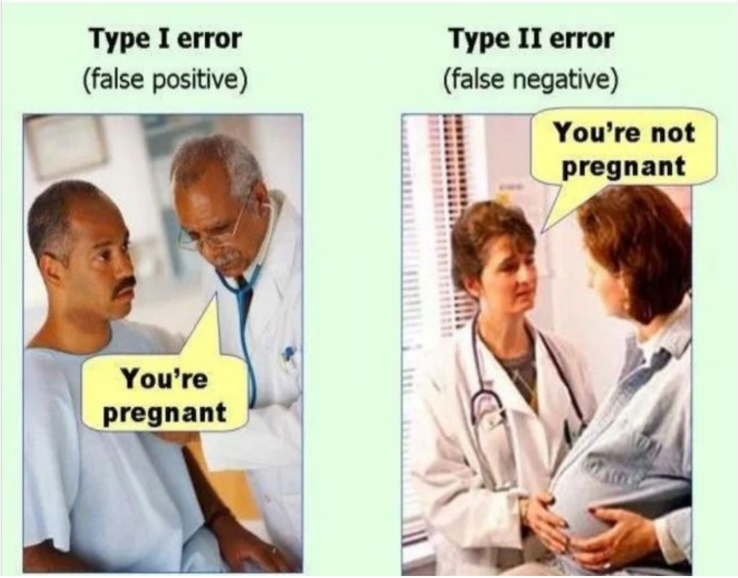
- **TP (True Positive):** N° of correctly predicted positive cases
- **TN (True Negative):** N° of correctly predicted negative cases
- **FP (False Positive):** N° of cases incorrectly predicted as positive (**Type I Error**)
- **FN (False Negative):** N° of cases incorrectly predicted as negative (**Type II Error**)

Note: Consider using a cost matrix to compare different models tailored to a specific use case.

Type I and Type II Error

In classification, detecting **Type I** and **Type II** errors is crucial because they represent different risks.

For example, in cancer prediction, False Negatives (**Type II errors**) can be a significant concern.



Main Classification Metrics

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

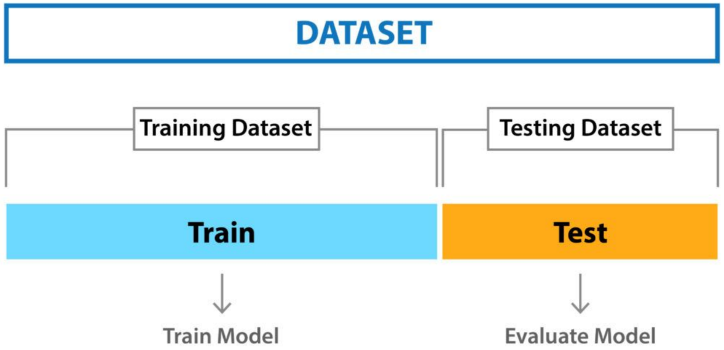
Main Classification Metrics

Metric	Formula	Interpretation
Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	Overall performance of model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$	Hybrid metric useful for unbalanced classes

Unbalanced classes have unequal distributions of class labels. In such cases, F1 score is more suitable than Accuracy

Data Splitting Strategies

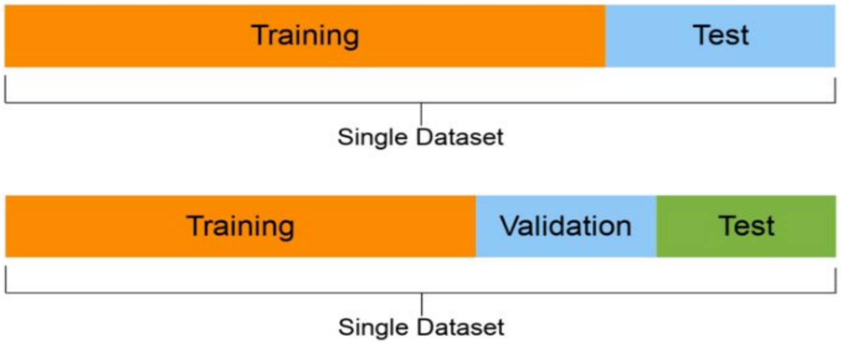
Holdout Method



Basic technique to partition data into training (D.train) and testing (D.test).

- **Error estimation**: Calculate error rate on **D.test** as a measure of generalization error.
- **Data proportion**: Analysts decide the split ratio (often **70**% training and **30**% testing).
- **Trade-off**: Balancing **D.train** size for model training and **D.test** size for reliable error estimation.
- **Repeated Holdout Method**: Enhance reliability by repeating the process and averaging error rates

Validation Set and Model Selection



Achieve an optimal balance between performance and model complexity.

Limitation of Training Error: Training error rate is insufficient for effective model selection.

Validation set: Essential to assess generalization and fine-tune hyperparameters before the test.

Model Selection: Combine model complexity with validation performance to select the best model.

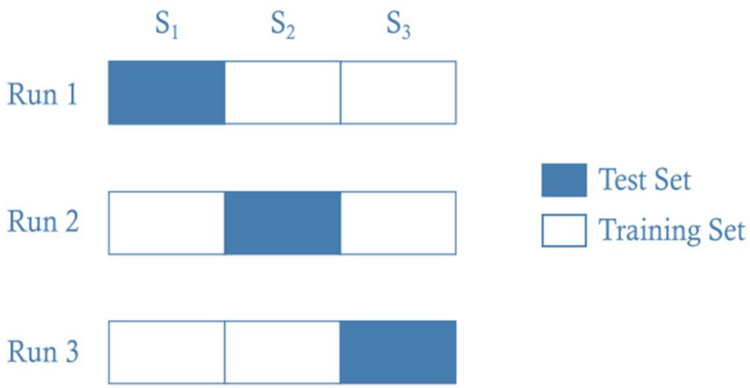
Model Complexity: In Decision Trees, measured by the ratio of leaf nodes to training instances.

Cross validation

- Cross validation helps to avoid the split bias of the holdout method.
- Divide data into ***k*** equal folds.
- Each fold is used exactly once for error calculation.
- The test error is averaged based on:

$$err_{test} = \frac{\sum_{i=1}^{k} err_{sum}(i)}{N}$$

Number of errors in one fold.

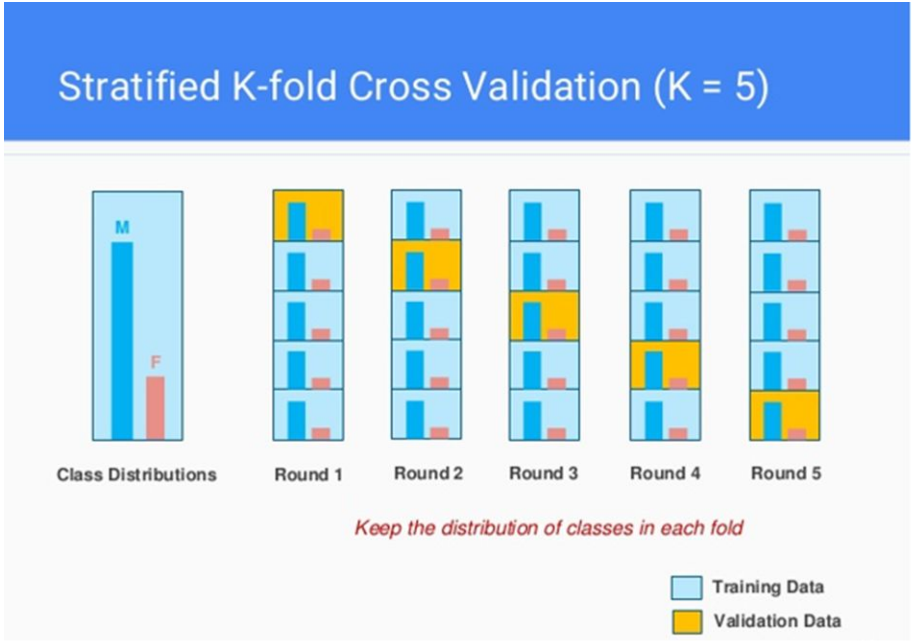


Variants of Cross validation

1. **Stratified Sampling**
   - Ensures equal representation of classes in each partition.

2. **Leave-One-Out Approach**
   - A special case where each instance is used once as a test set.
   - ***K*** = ***N***

Estimating Error Variance
   - Repeating cross-validation with different partitions provides robust error estimates.



Outline

- ❏ Characteristics of Decision Trees
- ❏ Decision Trees vs. Other models
- ❏ Model Evaluation
- ❏ **Model Diagnosis**

Model Diagnosis

Diagnosis: After model training, check it is not suffering from overfitting or underfitting.

Method: plot a learning curve of the model performance in both training and testing.

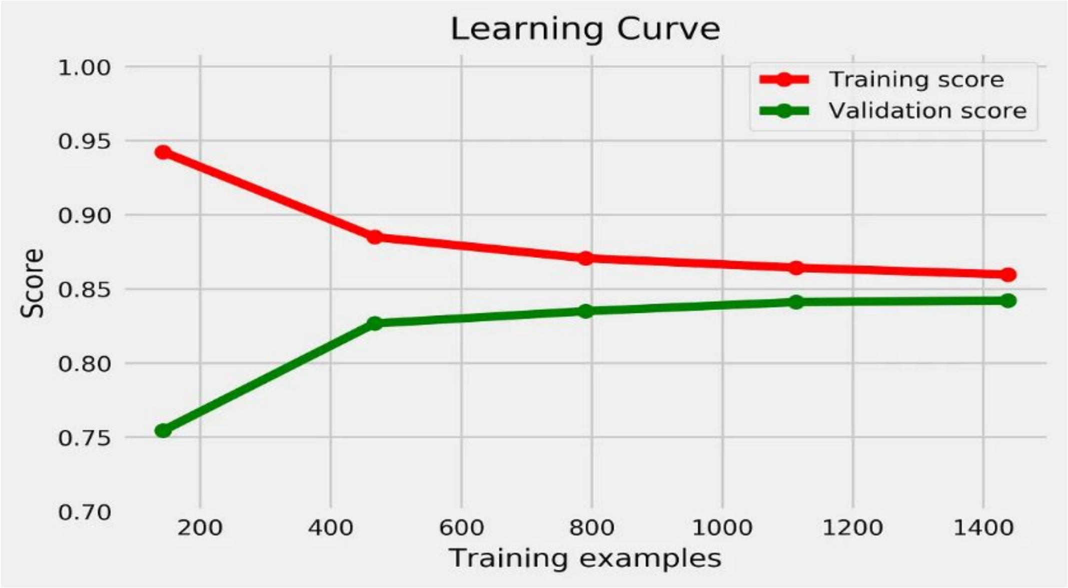
In the learning curve, check that:

- The performance is good enough
- There is no big gap between training and testing
- Whether we will get better results if we increase the size of the data

Learning curve: Accuracy score

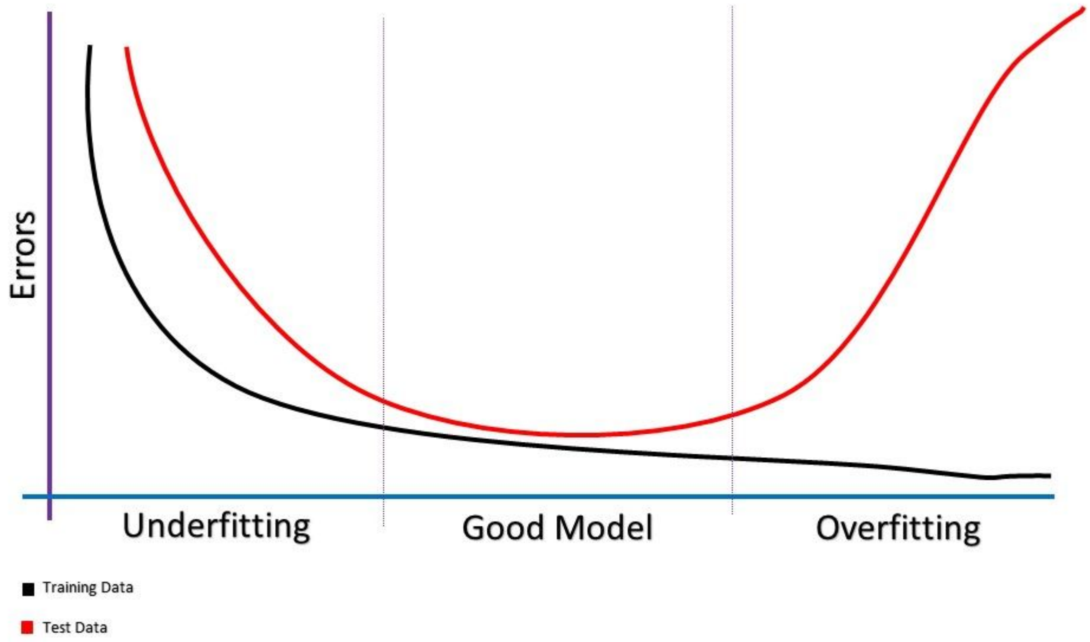
- The following example shows how accuracy score changes with increasing training examples.
- The plot shows a good performance in both train and validation.
- The plot suggests that the number of training examples is sufficient.

Accuracy



Learning curve: Error score

- The plot shows how the error score changes with increasing **model complexity**.
- Low model complexity indicates underfitting, while high model complexity indicates overfitting.
- The objective is to achieve a balance in model complexity

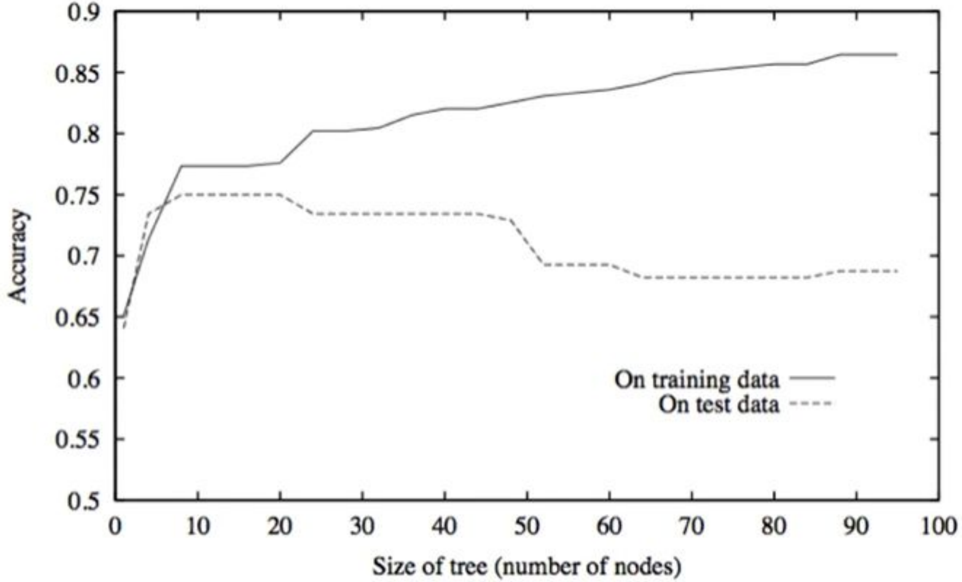


Model Overfitting

- Overfitting occurs when a model fits training data too closely, leading to poor generalization.

- An overfitted model may perform well on training data but poorly on test data.

- **Training vs Test Error:** As decision tree size increases, training error may decrease, but test error eventually increases.



Model Overfitting

Reasons:

- The data used for training is not clean and contains noise
- The size of the training data is too small
- The model is too complex (it captures training-specific patterns, reducing generalizability)
- Too many features

Remedies:

- Clean data and remove noise
- Train the model with sufficient data
- Use regularization techniques to reduce model complexity
- Improve the feature engineering process
- Use K-fold cross-validation as data splitting strategy

Model Underfitting

- The model is too simple to capture data patterns (poor performance on both train and test)

Reasons:

- The data used for training is not clean and contains noise
- The size of the training data is too small
- The model is too simple (it cannot capture training-specific patterns)
- Irrelevant features

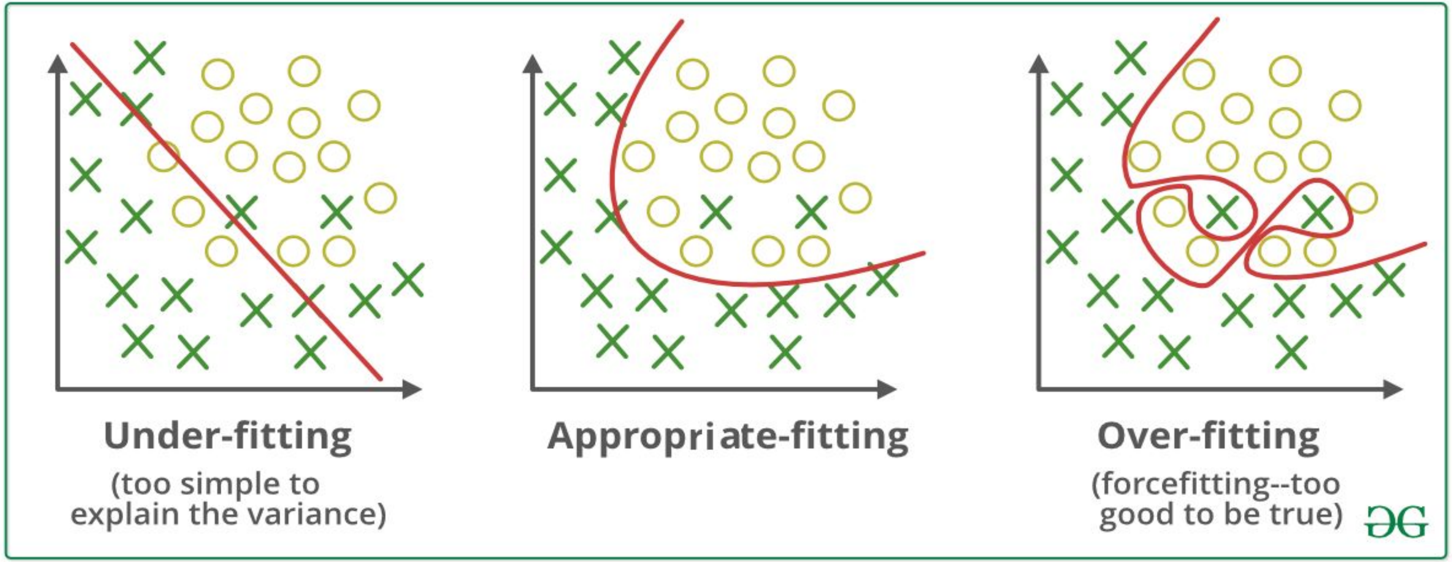
Remedies:

- Clean data and remove noise
- Train the model with sufficient data
- Increase the model complexity
- Improve the feature engineering process

Overfitting vs Underfitting:

The following example compares decision boundaries for a binary classification problem.

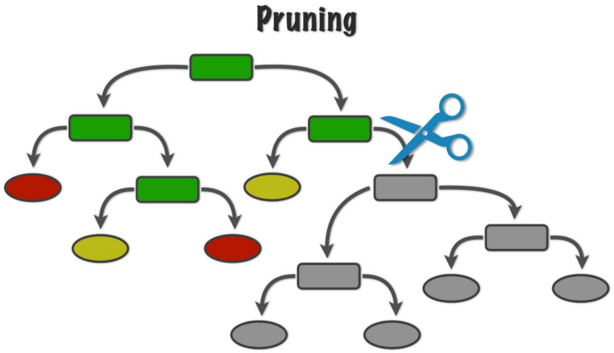
The main challenge for predictive models is to ensure an appropriate model fit.



Dealing with Overfitting in Decision Trees

Pruning: Cut away decision tree branches that may

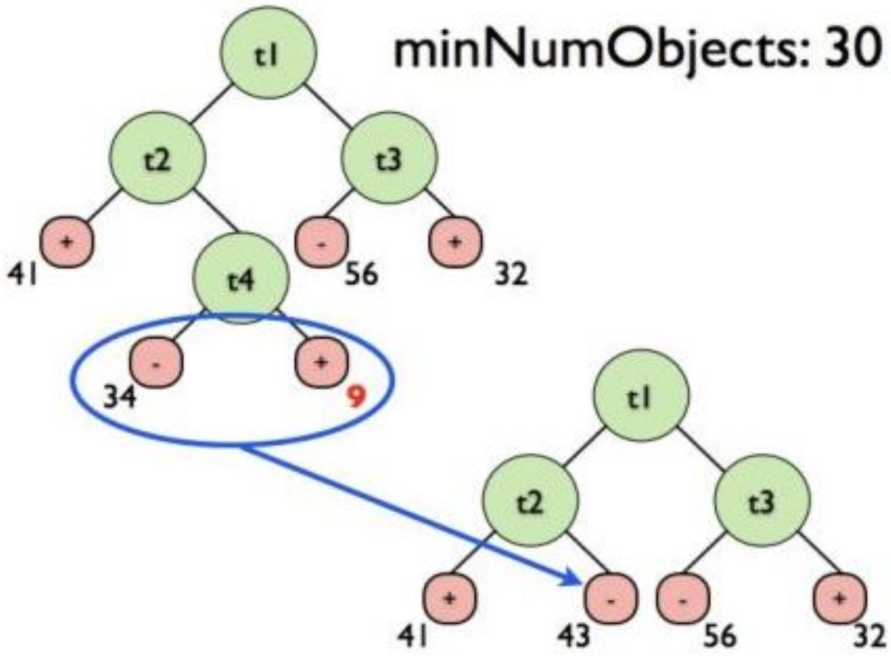
be based on noisy/misleading data to prevent overfitting.



- **Pre-pruning:** Occurs during tree construction.
  - Limits tree growth by limiting the maximum depth or minimum leaf size.
  - Prevents overfitting by avoiding overly complex models.

- **Post pruning:** Applied after the tree is fully grown.
  - Removes branches that contribute little to classification accuracy.
  - Reduces model complexity, enhancing generalization to new data.

Example: Pruning in a Decision Tree



Example: Pruning in a Decision Tree

```
MultiAgent = 0:
| depth > 2:   class 0
| depth <= 2:
| | MultiIP = 1:   class 0
| | MultiIP = 0:
| | | breadth <= 6:   class 0
| | | breadth > 6:
| | | | RepeatedAccess <= 0.322:   class 0
| | | | RepeatedAccess > 0.322:   class 1
MultiAgent = 1:
| totalPages <= 81:   class 0
| totalPages > 81:   class 1
```

Max depth = 3

```
MultiAgent = 0:   class 0
MultiAgent = 1:
| totalPages <= 81:   class 0
| totalPages > 81:   class 1
```