Classification 1

IE500 Data Mining

Outline

- **Decision Trees**
- **Overfitting**
- **Evaluation Metrics**
- Naïve Bayes
- Evaluation Methods
- Support Vector Machines
- Artificial Neural Networks
- Hyperparameter Selection

Introduction to Classification

- Goal: Learn a model for recognizing a concept, e.g. trees

"tree"

"tree"

"tree"

"not a tree"

"not a tree"

"not a tree"

Introduction to Classification

- Example: learning a new concept, e.g., "Tree"
  - we look at (positive and negative) examples (**training data**)
  - ...and derive a model e.g., "Trees are big, green plants"






- Goal: Classification of **unseen instances**

"tree?"

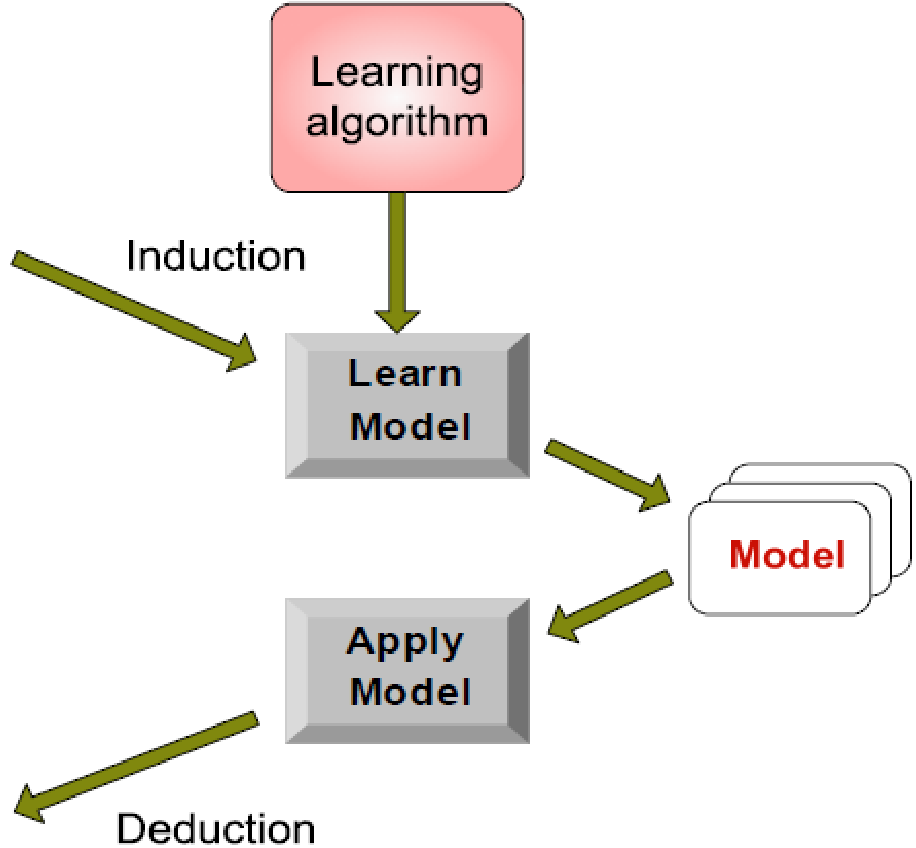
"tree?"

Warning:
Models are only approximating examples!
Not guaranteed to be correct or complete!

The Classification Workflow

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	?

Unseen Records

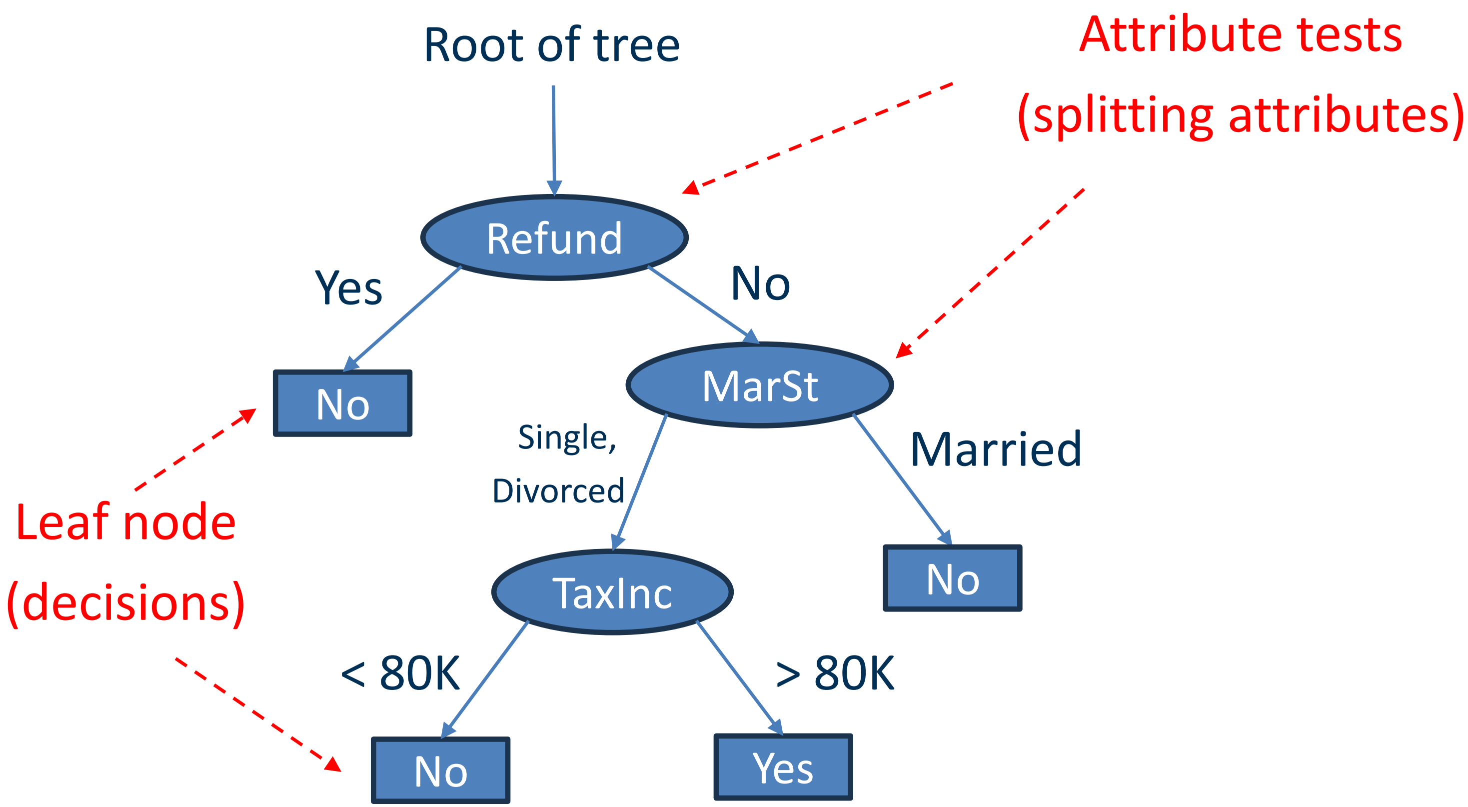
Lazy vs. Eager Learning

- **Lazy Learning**
  - Instance-based learning approaches, like KNN are lazy methods
  - Do not build a model
    - "learning" is only performed on demand for unseen records
    - <u>Single goal</u>: Classify unseen records as accurately as possible

- **Eager Learning**
  - but actually, we might have <u>two goals</u>
    1. classify unseen records
    2. understand the application domain as a human
  - Eager learning approaches generate models that are (might be) interpretable by humans
  - Examples of eager techniques: decision tree learning

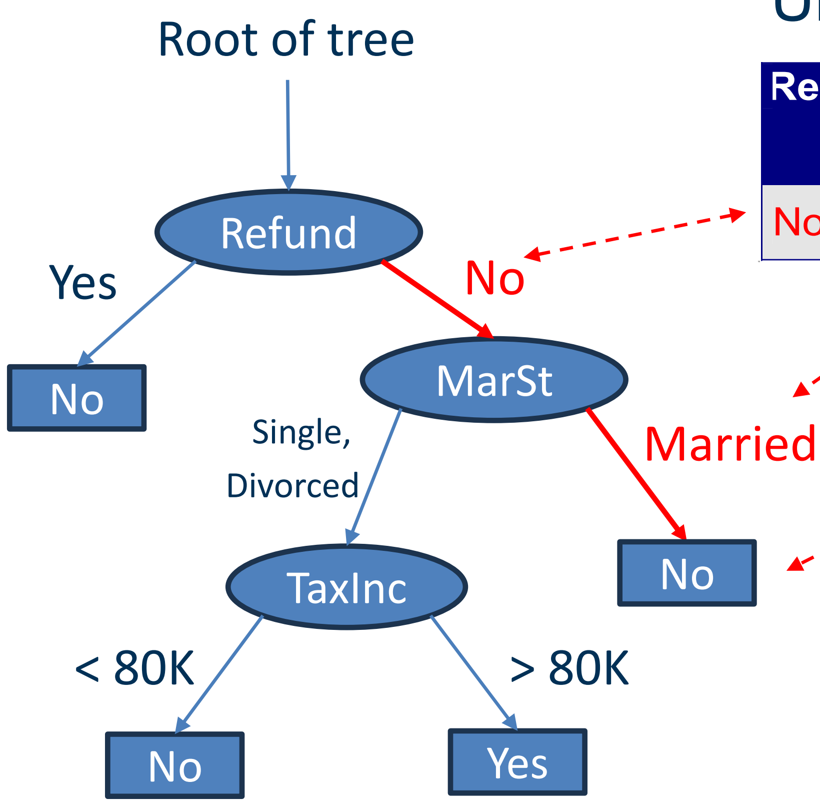
Explainable AI

Decision Tree Classifiers



- Decision trees encode a procedure for taking a classification decision

Applying a Decision Tree to Unseen Data

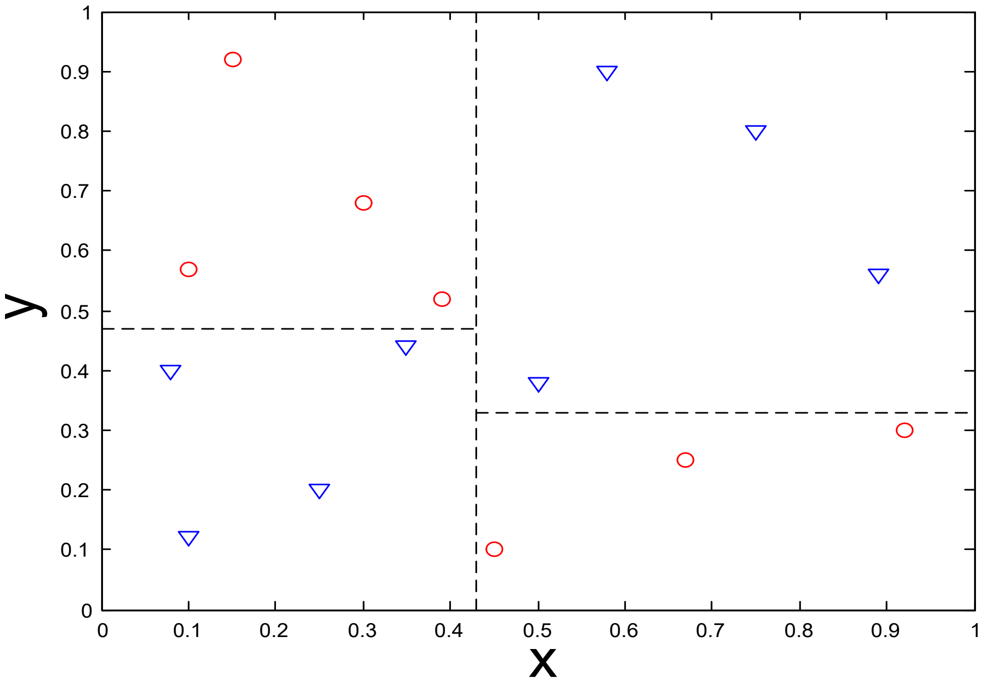


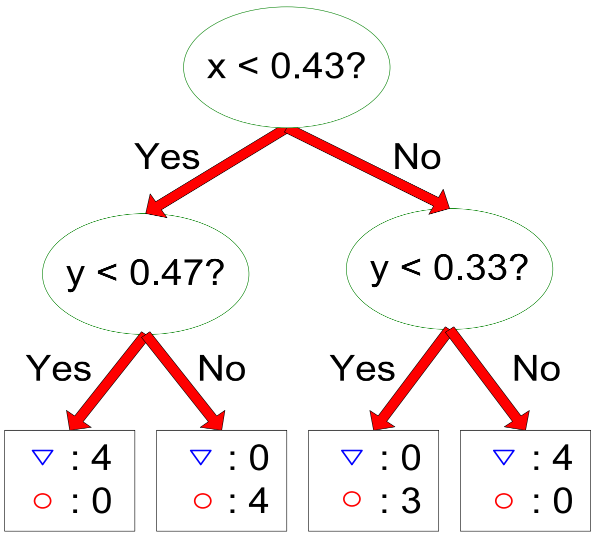
Unseen Record

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Decision Boundary





- The decision boundaries are parallel to the axes because the test condition involves a single attribute at-a-time

Learning a Decision Tree

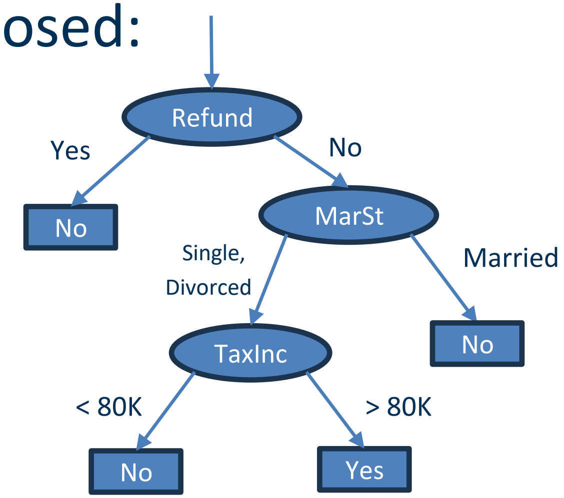
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- How to learn a decision tree from training data?
  - Finding an optimal decision tree is NP-hard
  - Tree building algorithms thus use a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
    - also known as: divide and conquer
- Many different algorithms have been proposed:
  - Hunt's Algorithm
  - ID3
  - C4.5
  - CHAID

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

Learning Algorithm

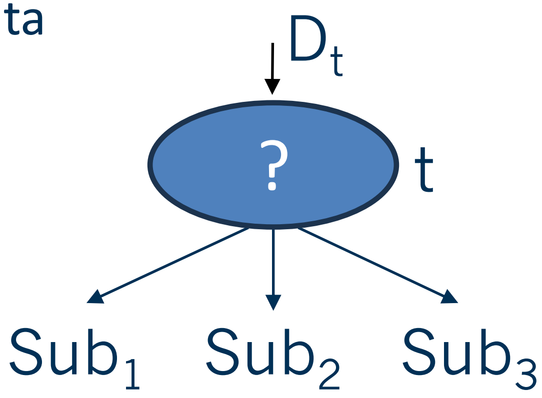


Model: Decision Tree

Hunt's Algorithm

- Let D_t be the set of training records that reach a node t
- Generate leaf node or attribute test:
  - if D_t only contains records that belong to the **same class** y_t , then t is a **leaf node** labeled as y_t
  - if D_t contains records that belong to **more than one class**, use an **attribute test** to split the data into subsets having a higher **purity**.
    - for all possible tests: calculate purity of the resulting subsets
    - choose test resulting in highest purity
- **Recursively** apply this procedure to each subset

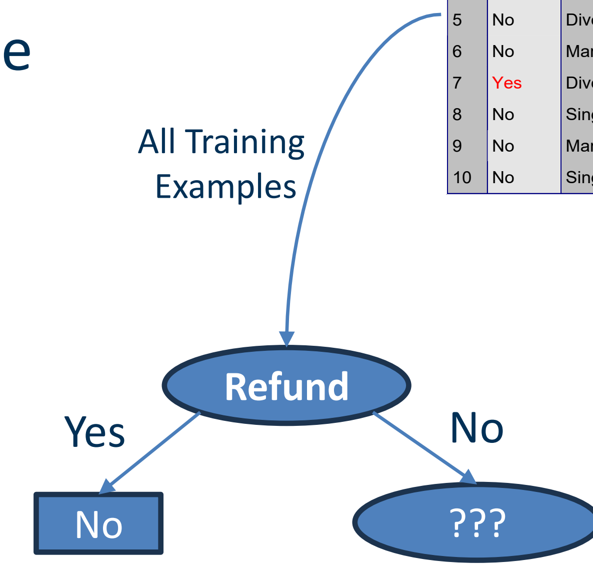
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt's Algorithm – Step 1

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

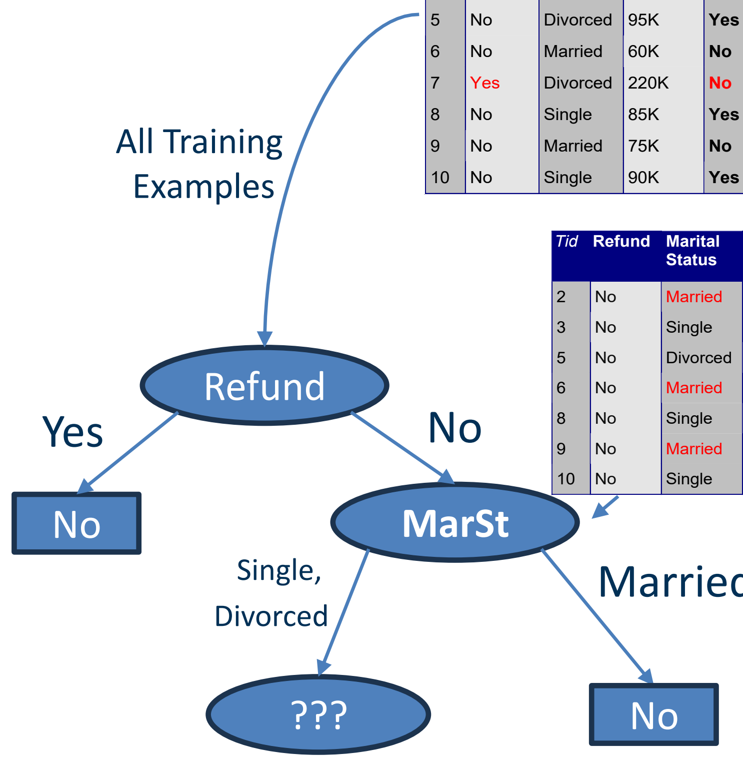
- We calculate the purity of the resulting subsets for all possible splits
  - Purity of split on Refund
  - Purity of split on Marital Status
  - Purity of split on Taxable Income
- We find the split on Refund to produce the purest subsets



Hunt's Algorithm – Step 2

- We further examine the Refund=No records
- Again, we test all possible splits
- We find the split on Marital Status to produce the purest subsets

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

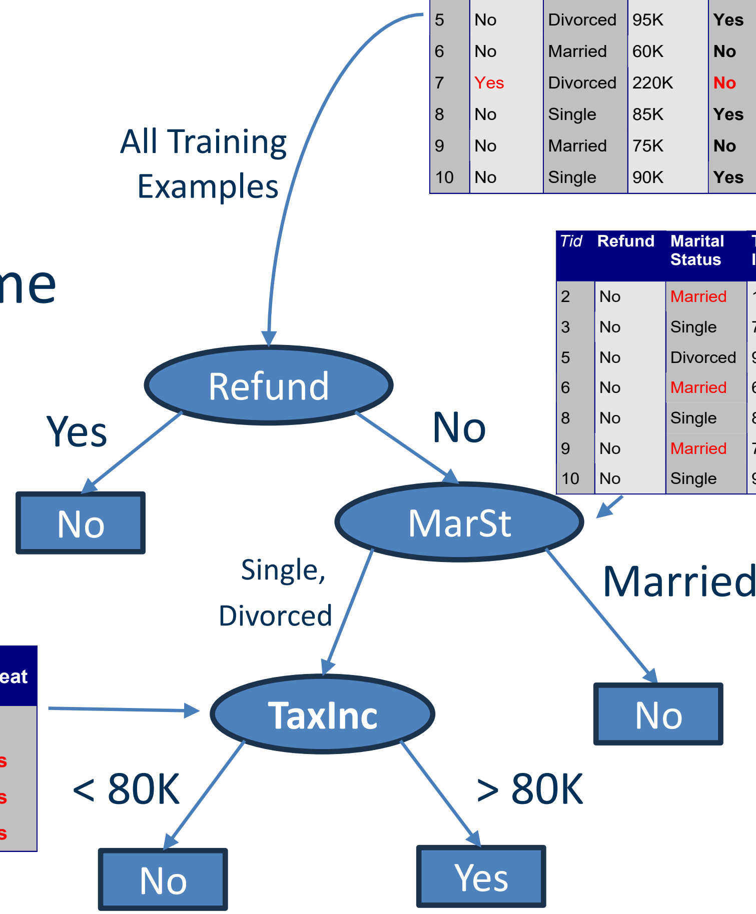


Tid	Refund	Marital Status	Taxable Income	Cheat
2	No	Married	100K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm – Step 3

- We further examine the
  - Marital Status=Single or
  - Marital Status= Divorced records
- We find a split on Taxable Income to produce pure subsets
- We stop splitting as no sets containing different classes are left

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Tid	Refund	Marital Status	Taxable Income	Cheat
2	No	Married	100K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tid	Refund	Marital Status	Taxable Income	Cheat
3	No	Single	70K	No
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes

Tree Induction Issues

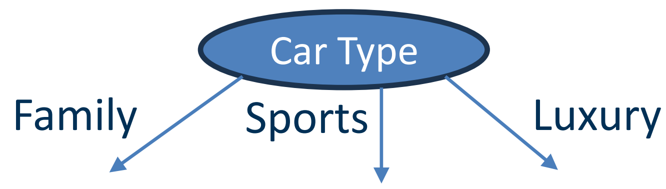
- Determine how to split the records
  - How to specify the attribute test condition?
    - Depends on attribute types
      - Nominal
      - Ordinal
      - Continuous
    - Depends on number of ways to split
      - 2-way split
      - Multi-way split
  - How to determine the best split?

Each will be discussed in the next slides

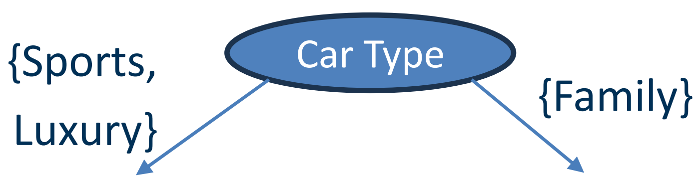
- Determine when to stop splitting

Splitting of Nominal Attributes

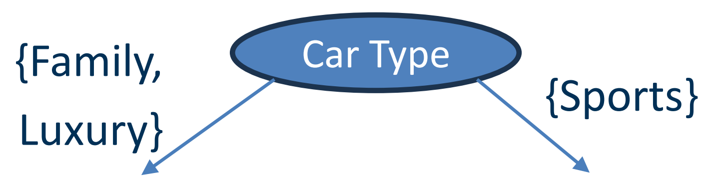
- **Multi-way split**: Use as many partitions as distinct values



- **Binary split**: Divides values into two subsets

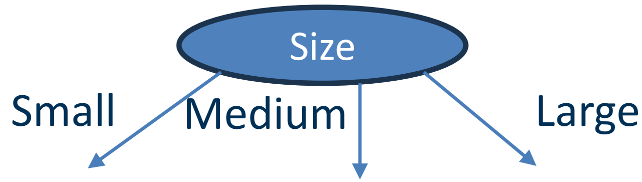


OR



Splitting of Ordinal Attributes

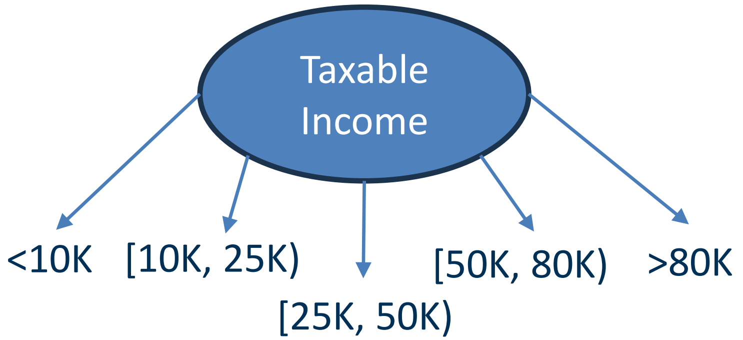
- **Multi-way split**: Use as many partitions as distinct values



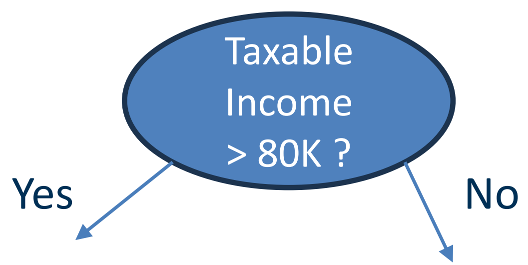
- **Binary split**: Divides values into two subsets
  while keeping the order

Splitting of Continuous Attributes

- **Multi-way split**: Discretization to form an ordinal attribute
  - equal-interval binning
  - equal-frequency binning
  - binning based on user-provided boundaries

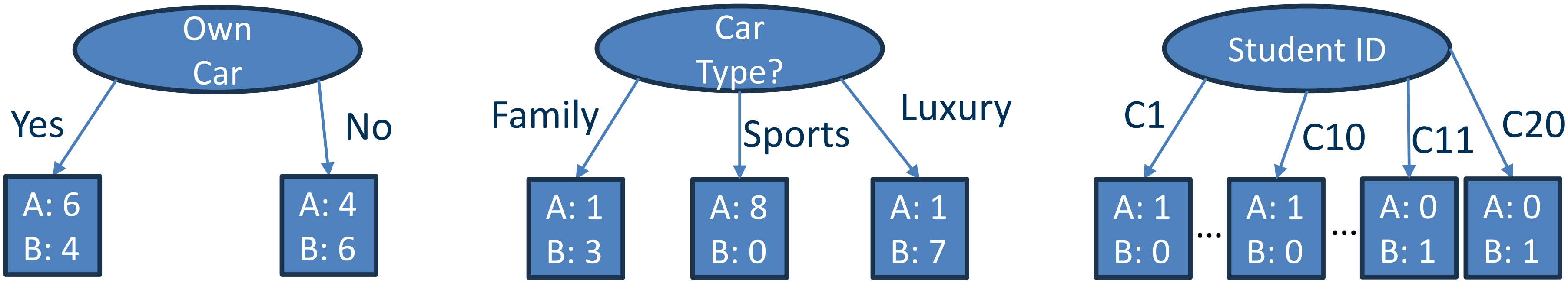


- **Binary split**: $(A < v)$ or $(A \geq v)$
  - usually sufficient in practice
  - find the best cut (i.e. the best v) based on a purity measure (see later)
  - can be computationally expensive



How to determine the Best Split?

- Before splitting the dataset contains:
  - 10 records of class A
  - 10 records of class B



Which attribute test is the best?

How to determine the Best Split?

UNIVERSITY OF MANNHEIM
Data and Web Science Group

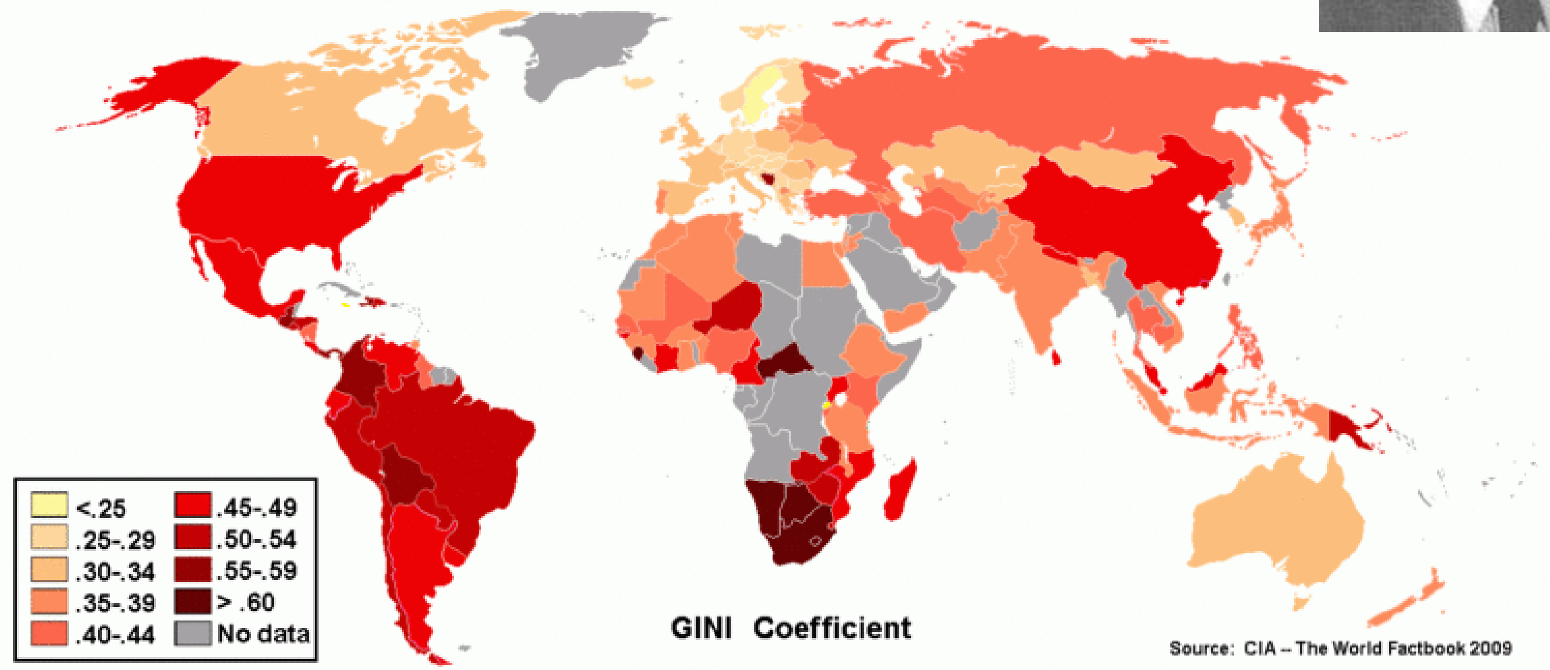
- Nodes with **homogeneous** class distribution are preferred
- Need a measure of **node impurity**:

A: 5 B: 5	A: 9 B: 1
Non-homogeneous	Homogeneous
High degree of node impurity	Low degree of node impurity

- Common measures of node impurity:
  - **GINI Index** (focus in this lecture)
  - Many other exist as well (e.g. Entropy)

Gini Index

- Named after Corrado Gini (1885-1965)
- Used to measure the distribution of income
  - 1: somebody gets everything
  - 0: everybody gets an equal share



Measure of Impurity: GINI

- Gini-based purity measure for a given node t :

$$\boxed{GINI(t) = 1 - \sum_j [p(j \mid t)]^2}$$

$p(j|t)$ is the relative frequency of class j at node t

- Minimum (0.0) when all records belong to one class, implying most interesting information

n_c = number of classes

- Maximum $(1 - \frac{1}{n_c})$ when records are equally distributed among all classes, implying least interesting information

A	0
B	6
Gini=0.000	

A	1
B	5
Gini=0.278	

A	2
B	4
Gini=0.444	

A	3
B	3
Gini=0.5	

Impurity increase

Examples for Computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

A	0
B	6

  $$P(A) = \frac{0}{6} = 0 \qquad P(B) = \frac{6}{6} = 1$$

  $$Gini(t) = 1 - P(A)^2 - P(B)^2 = 1 - 0 - 1 = 0$$

A	1
B	5

  $$P(A) = \frac{1}{6} \qquad P(B) = \frac{5}{6}$$

  $$Gini(t) = 1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = \frac{10}{36}\ 0.278$$

A	2
B	4

  $$P(A) = \frac{2}{6} \qquad P(B) = \frac{4}{6}$$

  $$Gini(t) = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = \frac{16}{36} = 0.444$$

Splitting Based on GINI

- When a node p is split into k partitions (children), the quality of split is computed as:

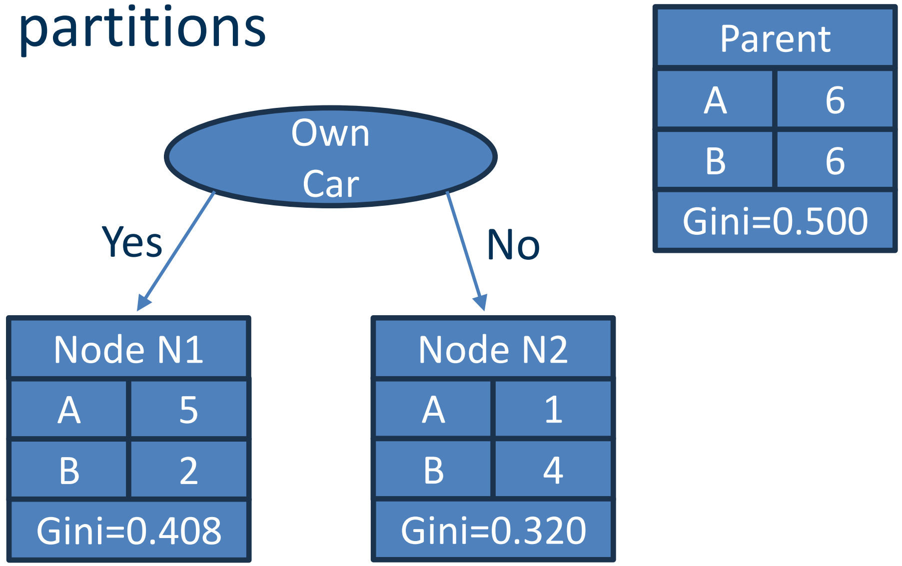
$$\boxed{GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)}$$

  - where n_i = number of records at child i,
  - n = number of records at node p

- Intuition:
  - The GINI index of each partition is weighted according to the partition's size

Computing GINI Split

- Split into two partitions



$$GINI_{split} = \frac{7}{12} * 0.408 \quad + \quad \frac{5}{12} * 0.320 = 0.371$$

- Purity **Gain** = impurity measure before splitting - after splitting

  = 0.500 – 0.371 = 0.129

  – Purity Gain is used to decide for the best split (highest purity gain or lowest $GINI_{split}$)
  – When using Entropy, then it is called **Information Gain**

Categorical Attributes: Computing Gini Index

- For each distinct attribute value, gather counts for each class

Multi-way split

	Car Type		
	Family	Sports	Luxury
A	1	2	1
B	4	1	1
Gini=0.393			

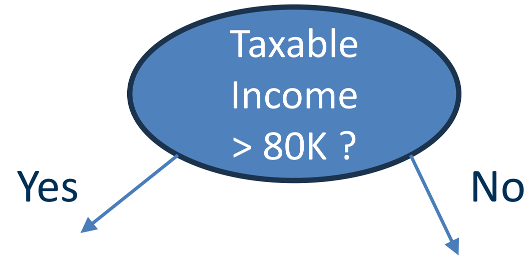
Two-way split
(find best partition of values)

	Car Type	
	{Sports, Luxury}	Family
A	3	1
B	2	4
Gini=0.400		

	Car Type	
	{Sports}	{Family, Luxury}
A	2	2
B	1	5
Gini=0.419		

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient!
  - Repetition of work



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Continuous Attributes: Computing Gini Index

- Efficient computation:
  1. sort the attribute on values
  2. linearly scan these values, each time updating the count matrix and computing the gini index
  3. choose the split position that has the smallest gini index

	Taxable Income																					
Sorted Values →		60		70		75		85		90		95		100		120		125		220		
Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

Continuous Attributes: Computing Gini Index

- Note: it is enough to compute the GINI for those positions where the label changes!

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income	60	70	75	85	90	95	100	120	125	220

	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400		0.420	

Discussion of Decision Trees

- Advantages:
  - Inexpensive to construct
  - Fast at classifying unknown records
  - **Easy to interpret by humans for small-sized trees**
  - Accuracy is comparable to other classification techniques for many simple data sets

Explainable model!

- Disadvantages:
  - Decisions are based only one a single attribute at a time
  - Can only represent decision boundaries that are parallel to the axes

Comparing Decision Trees and k-NN

- Decision boundaries
  - k-NN: arbitrary
  - Decision trees: rectangular

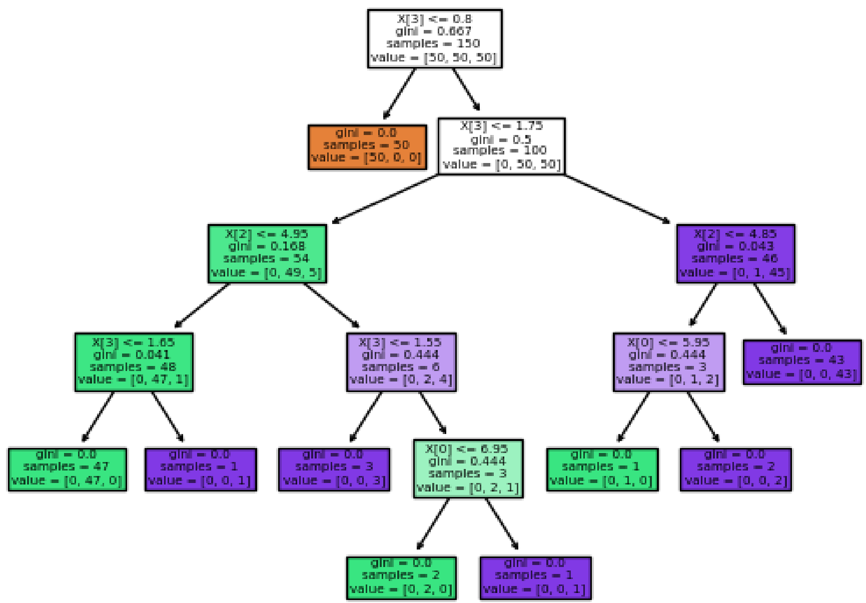
- Sensitivity to scales
  - k-NN: needs normalization
  - Decision tree: does not require normalization (recap: Gini splitting)

- Runtime & memory
  - k-NN is more cheap to train, but more expensive for classification
  - Decision tree is more expensive to train,
    but more cheap for classification

Tree Induction in Python

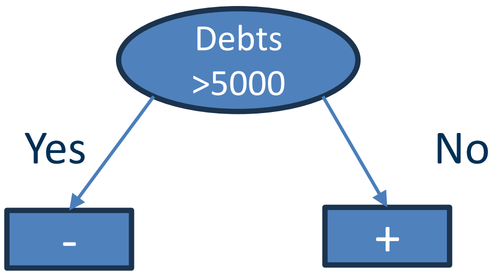
Python

```python
from sklearn.tree import DecisionTreeClassifier

# Train classifier
dt_learner = DecisionTreeClassifier(criterion='gini', max_depth=10)
dt_learner.fit(preprocessed_training_data, training_labels)

# Use classifier to predict labels
prediction = dt_learner.predict(preprocessed_unseen_data)
```



Practical Issue: Overfitting

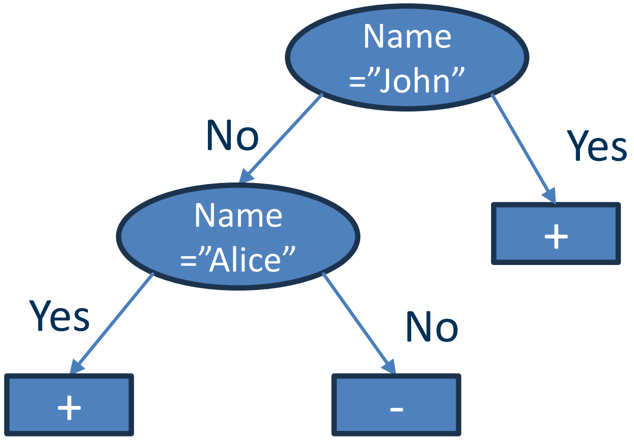
- Example: Predict credit rating
  - possible decision tree:



Name	Net Income	Job status	Debts	Rating
John	40000	employed	0	+
Mary	38000	employed	10000	-
Stephen	21000	self-employed	20000	-
Eric	2000	student	10000	-
Alice	35000	employed	4000	+

Practical Issue: Overfitting

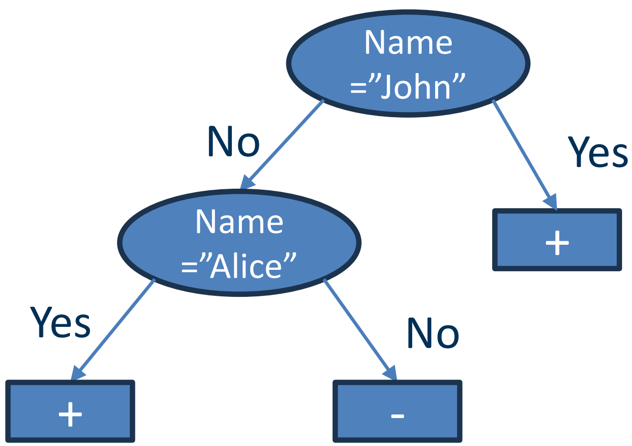
- Example: Predict credit rating
  - Alternative decision tree:

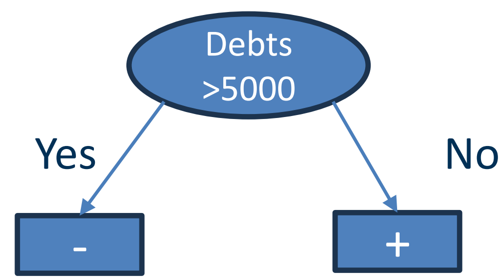


Name	Net Income	Job status	Debts	Rating
John	40000	employed	0	+
Mary	38000	employed	10000	-
Stephen	21000	self-employed	20000	-
Eric	2000	student	10000	-
Alice	35000	employed	4000	+

Practical Issue: Overfitting

- Both trees seem equally good
  - Classify all instances in the training set correctly
- Which one do you prefer?





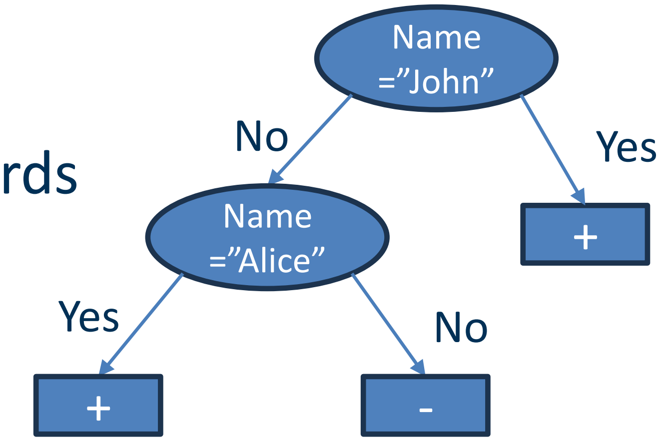
Occam's Razor

- Named after William of Ockham (1287-1347)
- A fundamental principle of science
  - If you have two theories
  - that explain a phenomenon equally well
  - choose the simpler one

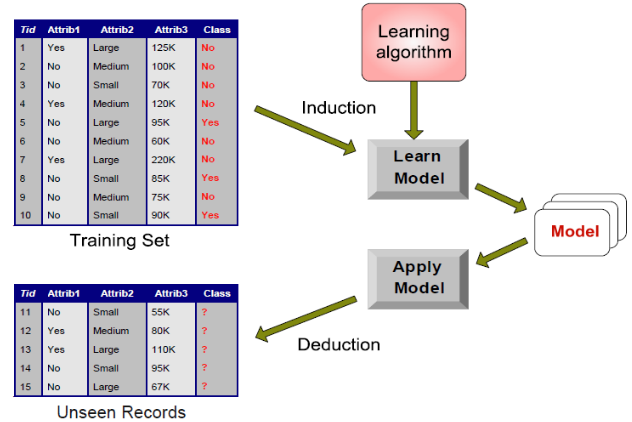
- Example:
  - Phenomenon: the street is wet
  - Theory 1: it has rained
  - Theory 2: a beer truck has had an accident, and beer has spilled. The truck has been towed, and magpies picked the glass pieces, so only the beer remains

Training and Testing Data

- Consider the decision tree again
- Our ultimate goal: classify unseen records



- Assume you measure the performance using the training data
- Conclusion:
  - We need separate data for testing



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

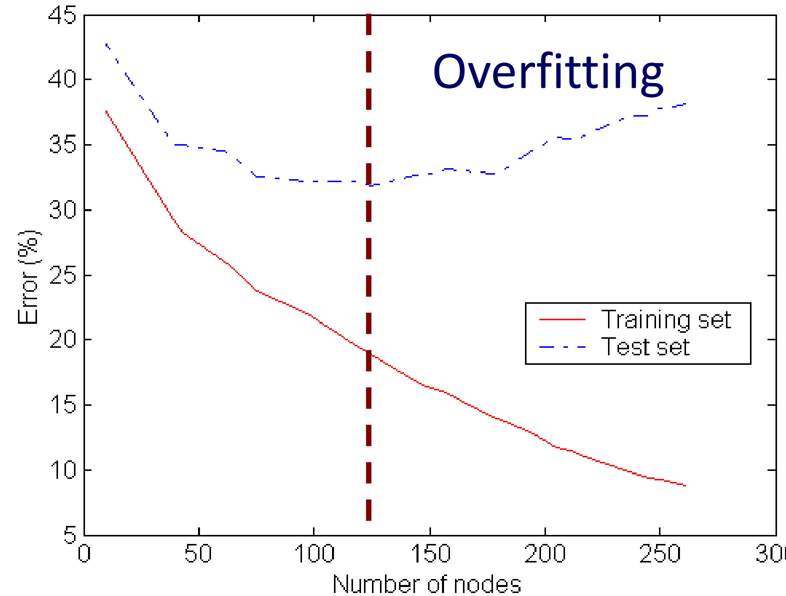
Training Set

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	?

Unseen Records

Overfitting: Symptoms and Causes

- Symptoms:
  - Decision tree too deep
  - Too many branches
  - **Model works well on training set but performs bad on test set**

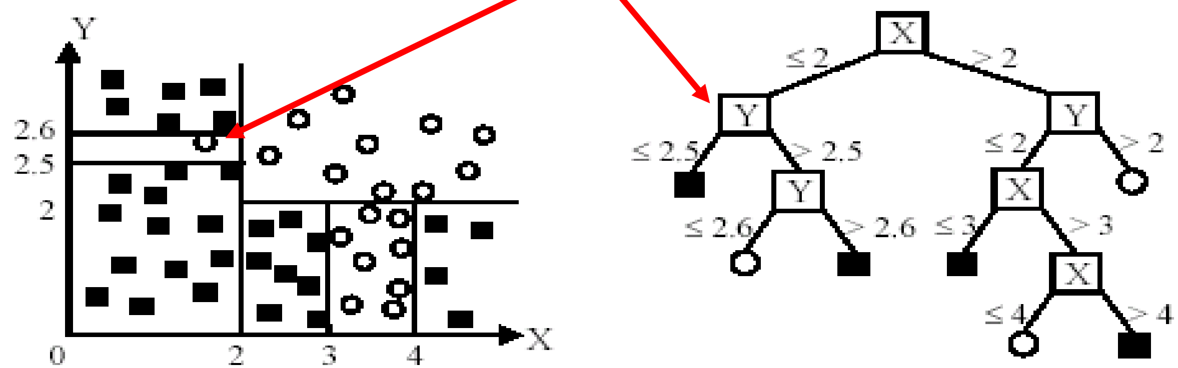


- Typical causes of overfitting
  - Noise / outliers in training data
  - Too little training data
  - High model complexity

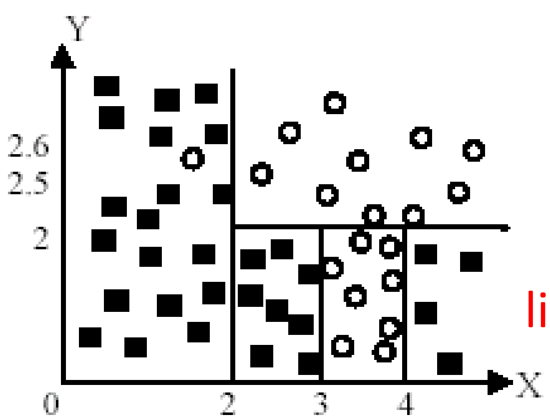
An overfitted model does not **generalize** well to **unseen data**.

Overfitting and Noise

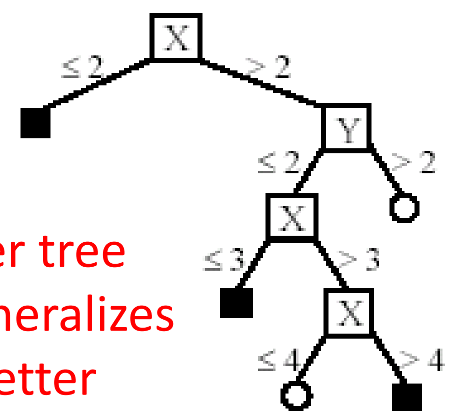
Likely to overfit the data



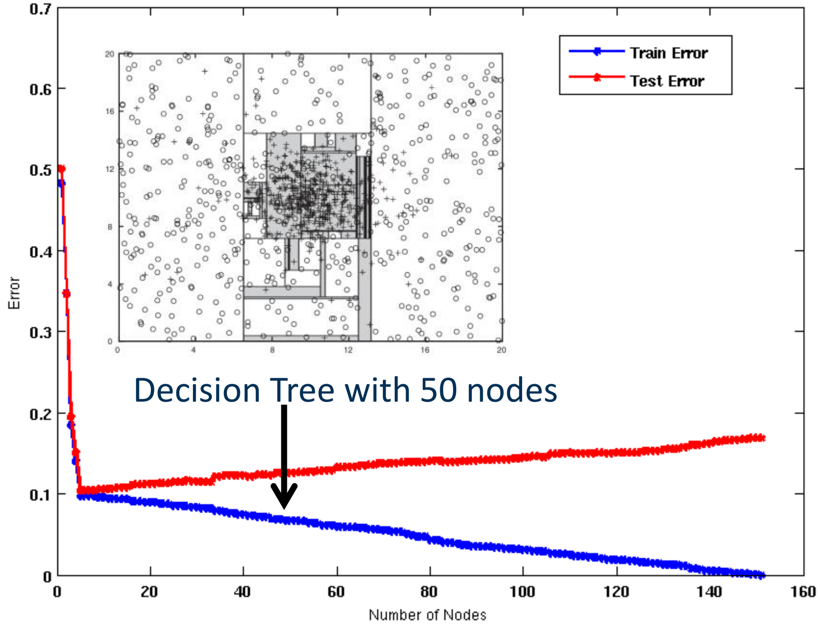
(A) A partition of the data space (B). The decision tree

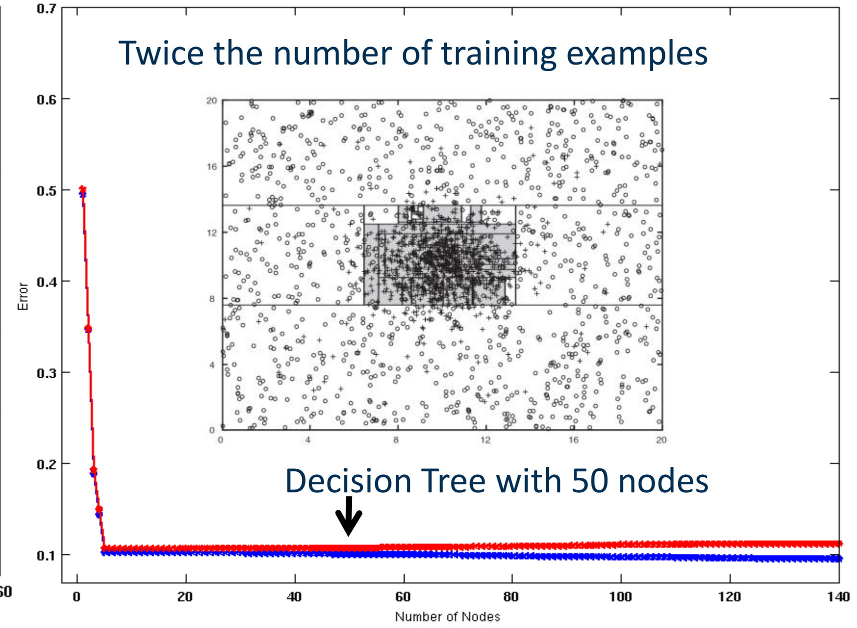


Simpler tree likely generalizes better



How to Prevent Overfitting 1: Use More Training Data





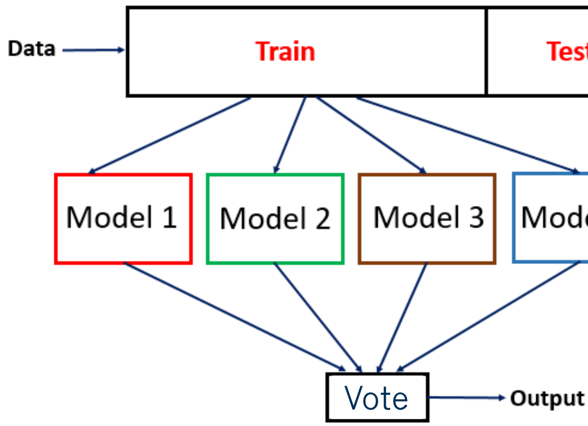
- If training data is **under-representative**, training errors decrease but testing errors increase on increasing number of nodes
- Increasing the size of training set reduces the difference between training and testing errors at a given number of nodes

How to Prevent Overfitting 2: Pre-Pruning

- **Stop the algorithm before tree becomes fully-grown**
  - shallower tree potentially generalizes better (Occam's razor)
- Normal stopping conditions for a node (no pruning):
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- **Early stopping conditions** (pre-pruning):
  - Stop if number of instances within a leaf node is less than some user-specified threshold (e.g. leaf size < 4)
  - Stop if expanding the current node only slightly improves the impurity measure (e.g. gain < 0.01)
  - Stop splitting at a specific depth (e.g. maxDepth = 5)

How to Prevent Overfitting 3: Ensembles

- Learn different models (base learners)
- Have them vote on the final classification decision



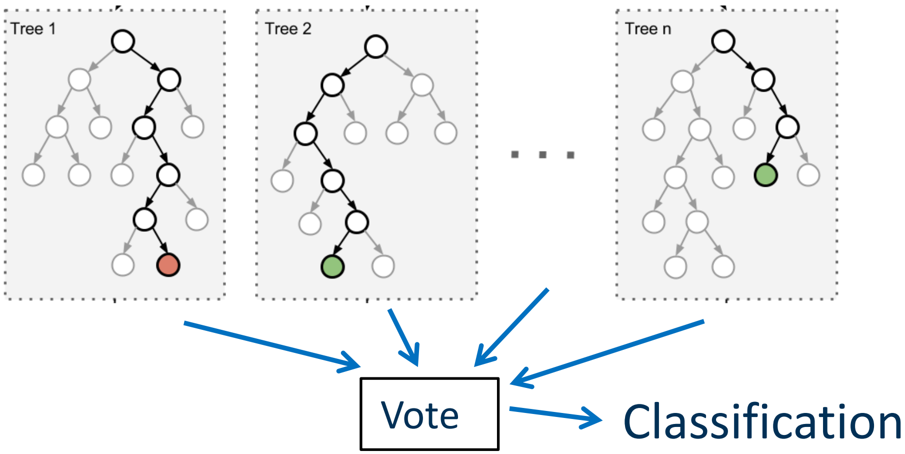
- Idea: Wisdom of the crowds applied to classification
  - A single classifier might focus too much on one aspect
  - Multiple classifiers can focus on different aspects

Algorithms to Learn Tree Ensembles

- Random Forest (Breiman 1997)
  - sklearn.ensemble.RandomForestClassifier
- Gradient Tree Boosting (Friedman 1999)
  - sklearn.ensemble.GradientBoostingClassifier
- Gradient Tree Boosting with Regularization
  - XGBoost (extra package)
- Ensembles perform better than e.g. simple decision trees
  - Disadvantage: usually the interpretability is lost due to many trees

Random Forest

- Ensemble consisting of a large number of different decision trees



- Independence of trees achieved by introducing randomness into the learning process
  - only use a random subset of the attributes at each split
  - learn on different random subsets of the data (bagging)

Random Forest

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Random forests usually outperform single decision trees

- Random Forest in Python

Python

```
from sklearn.ensemble import RandomForestClassifier

# Train classifier
forest_estimator = RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=None)
forest_estimator.fit(preprocessed_training_data, training_labels)

# Use classifier to predict labels
prediction = forest_estimator.predict(preprocessed_unseen_data)
```

XGBoost

- A Scalable System for Learning Tree Ensembles
  - Model improvement
    - Regularized objective for better model
    - Objective Function: $\underbrace{Obj(\Theta) = \underbrace{L(\Theta)}_{\text{Training Loss}} + \underbrace{\Omega(\Theta)}_{\text{Regularization (complexity of model)}}$

      Parameters w_i to be learned
    - Optimizing training loss yields good predictive models
    - Optimizing regularization yields simple models
      - making predictions stable
  - Systems optimizations
    - Out of core computing
    - Parallelization
    - Cache optimization
  - Implemented in a extra package “xgboost”

https://xgboost.readthedocs.io/en/stable/python/sklearn_estimator.html

XGBoost

- Gradient Tree Boosting
  - $Obj(\Theta) = L(\Theta) + \Omega(\Theta)$
    - We can not use methods such as stochastic gradient descent (SGD)
      - Because of discrete building steps
    - Solution: Additive Training (Boosting)
      - Start with a simple model and enhance it in every round
- Regularization
  - How to define the complexity of a tree?
    - $\Omega(\Theta) = yT + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$

Number of leaves

L2 norm of leaf scores

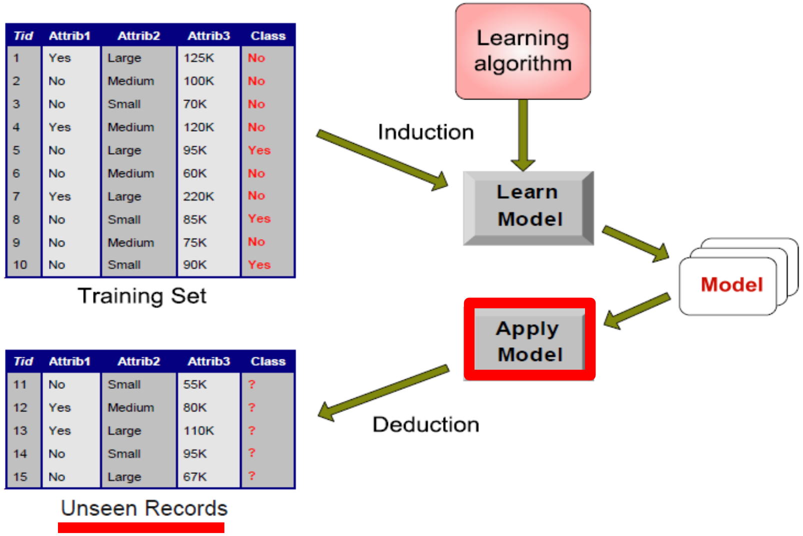
https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Model Evaluation

- Central Question:
  – How good is a model at classifying unseen records? (generalization performance)

- This week: **Evaluation Metrics**
  – How to measure the performance of a model?

- Next week: **Evaluation Methods**
  – How to obtain reliable estimates?



Confusion Matrix

- Focus on the **predictive capability** of a model
  - Looking at correctly/incorrectly classified instances
  - Two class problem (positive/negative class)
    - First word: **true**, if prediction is correct (otherwise **false**)
    - Second word: **positive** or **negative** (dependent on the **predicted** label)

		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	**True Positives (TP)**	**False Negatives (FN)**
	Class=No	**False Positives (FP)**	**True Negatives (TN)**

Metrics for Performance Evaluation

- Most frequently used metrics:
  - $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{Correct\ predictions}{All\ predictions}$

  - $Error\ Rate = 1 - Accuracy$

		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	TP 25	FN 4
	Class=No	FP 6	TN 15

$$Accuracy = \frac{25+15}{25+15+6+4} = 0.8$$

What is a Good Accuracy?

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- i.e., when are you done?
  - at 75% accuracy?
  - at 90% accuracy?
  - at 95% accuracy?
- Depends on difficulty of the problem!
- **Baseline**: naive guessing
  - always predict majority class
- Compare
  - Predicting coin tosses with accuracy of 50%
  - Predicting dice roll with accuracy of 50%
  - Predicting lottery numbers (6 out of 49) with accuracy of 50%

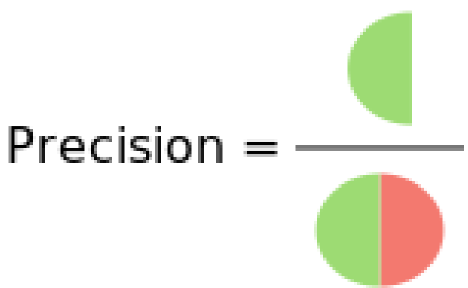
Limitation of Accuracy: Unbalanced Data

- Classes often have **very unequal frequency**
  - Fraud detection: 98% transactions OK, 2% fraud
  - E-commerce: 99% surfers don't buy, 1% buy
  - …

- Consider a 2-class problem:
  - Number of negative examples = 9990,
    Number of positive examples = 10
  - if model predicts all examples to belong to the negative class, the accuracy is
    9990/10000 = 99.9 %
  - **Accuracy is misleading** because model does not detect any positive example

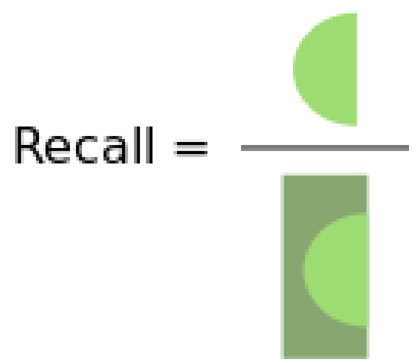
Precision and Recall

How many examples that are classified positive are actually positive?

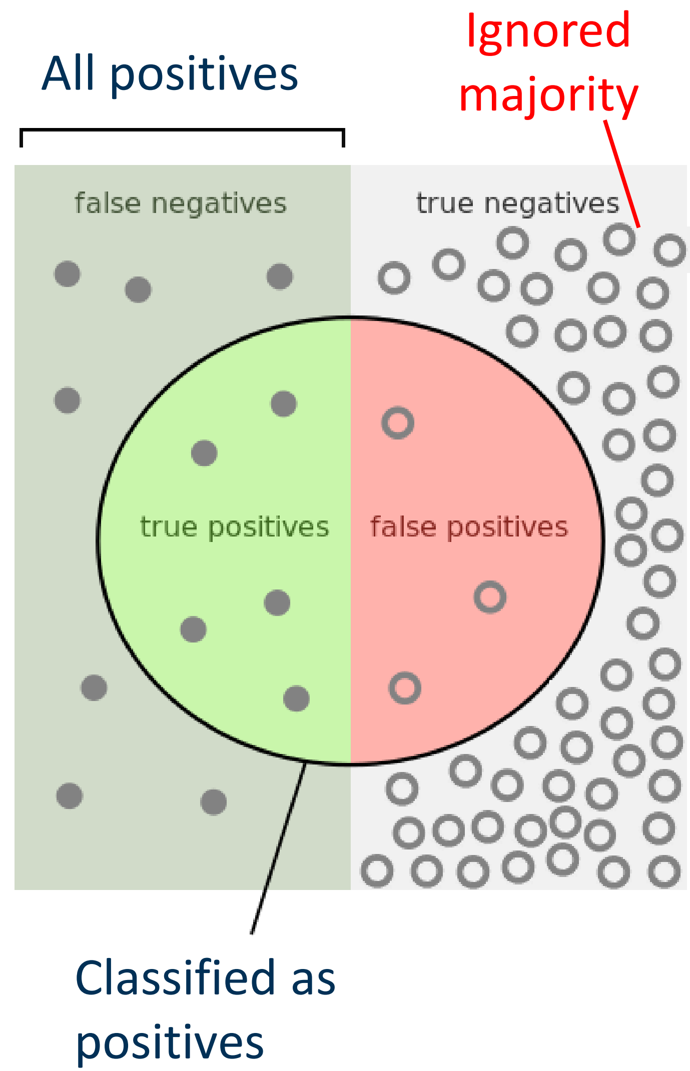


$$p = \frac{TP}{\mathrm{TP} + \mathrm{FP}}$$

Which fraction of all positive examples is classified correctly?



$$r = \frac{TP}{\mathrm{TP} + \mathrm{F}N}$$



Precision and Recall – A Problematic Case

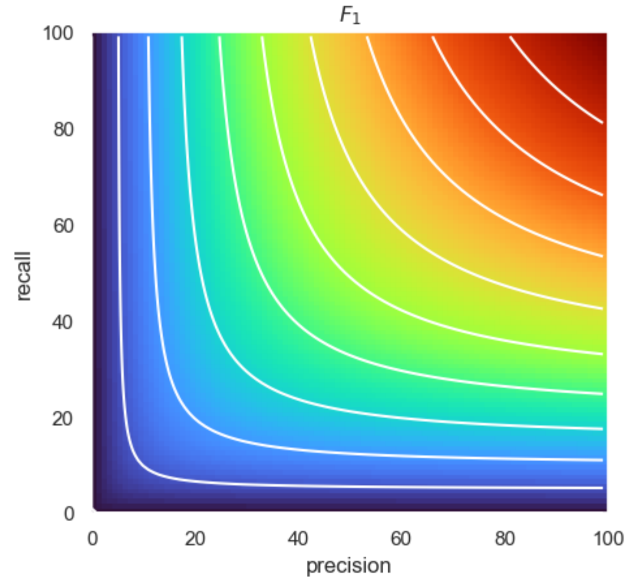
		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	TP 1	FN 99
	Class=No	FP 0	TN 1000

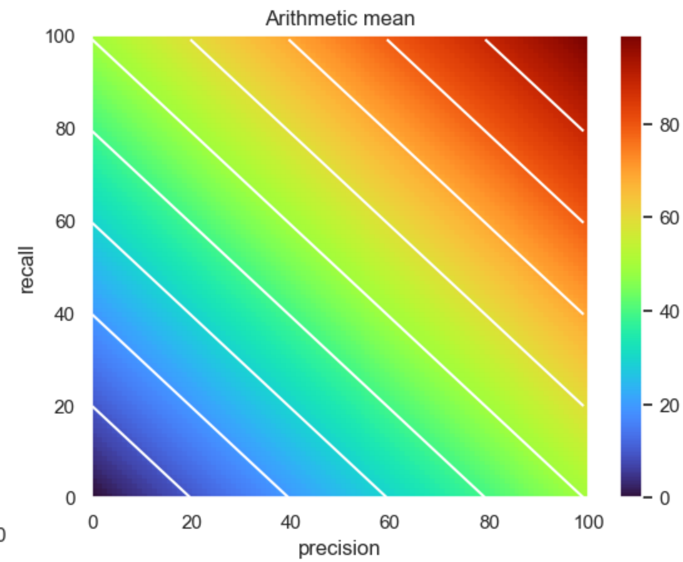
- This confusion matrix gives us
  - precision p = 100%
  - recall r = 1%
- Because we only classified one positive example correctly and no negative examples wrongly
- Thus, we want a measure that
  - combines precision and recall and is large if both values are large

F_1 -Measure

- F_1-score **combines precision and recall** into one measure
- F_1-score is the harmonic mean of precision and recall
  - The harmonic mean of two numbers tends to be closer to the smaller of the two
  - Thus for the F_1-score to be large, both *p* and *r* must be large

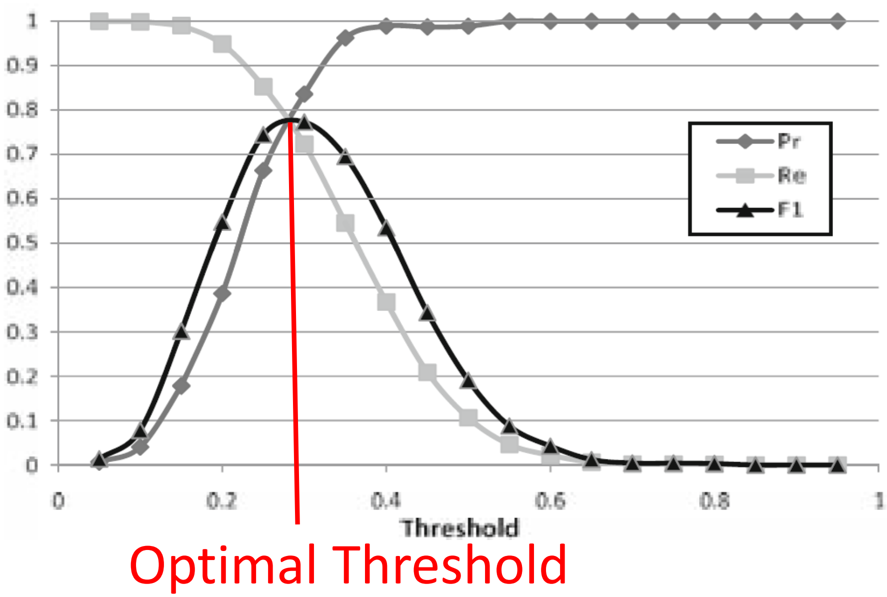
- $F_1 = 2 * \frac{p * r}{p + r}$





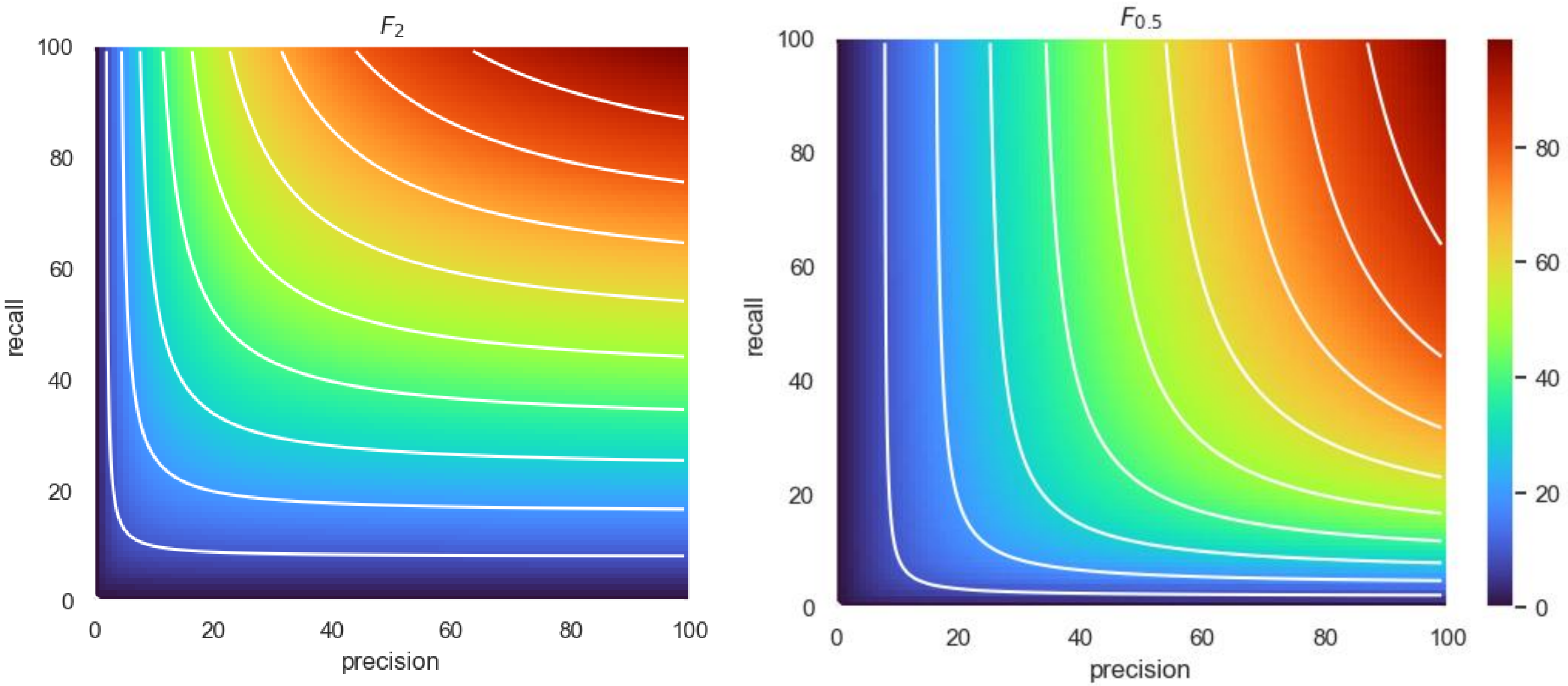
F_1 -Measure Graph

- Low threshold: Low precision, high recall
- Restrictive threshold: High precision, low recall



F_β -Measure

- More general $F_\beta = (1+\beta^2) * \frac{p * r}{(\beta^2 * p) + r}$
  - $\beta = 2$ weights recall higher, $\beta = 0.5$ weights precision higher



Cost-Sensitive Model Evaluation

UNIVERSITY OF MANNHEIM
Data and Web Science Group

Cost Matrix		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	-1	100
	Class=No	1	0

- Associate a cost for each error
  - Use case: Credit card fraud
    - it is expensive to miss fraudulent transactions
    - false alarms are not too expensive

Model M1		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	162	38
	Class=No	160	240

Model M2		Predicted Class	
		Class=Yes	Class=No
Actual Class	Class=Yes	155	45
	Class=No	5	395

Accuracy = 67%

Cost = 3798 ← Better model

Accuracy = 92%

Cost = 4350

ROC Curves

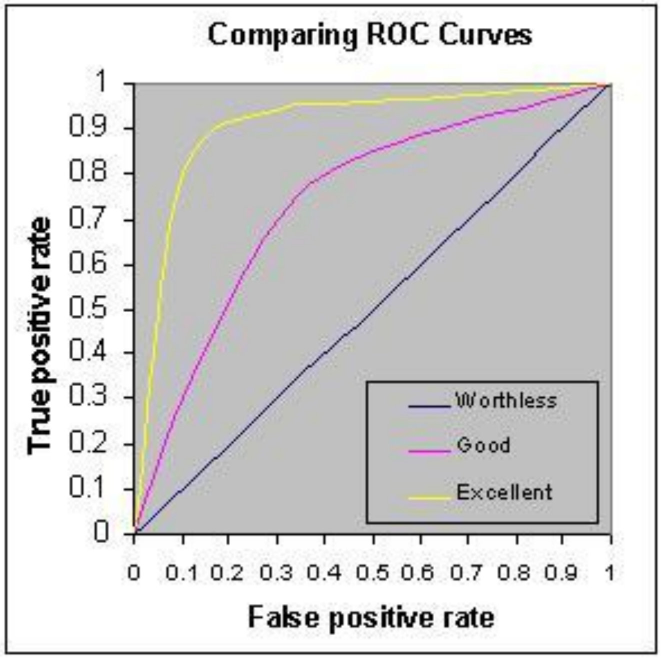
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Some classification algorithms provide **confidence scores**
  - how sure the algorithms is with its prediction
  - e.g., KNN (the neighbor's vote), Naive Bayes (the probability)
- ROC curves visualize **true positive rate** and **false positive rate** in relation to the algorithm's confidence
- Drawing a ROC Curve
  - Sort classifications according to confidence scores (e.g.: fraction of neighbours in k-NN model)
  - Evaluate
    - Correct prediction: draw one step up
    - Incorrect prediction: draw one step to the right

Interpreting ROC Curves

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Best possible result:
  - all correct predictions have higher confidence than all incorrect ones
- **The steeper, the better**
  - random guessing results in the diagonal
  - so a decent algorithm should result in a curve significantly above the diagonal
- Comparing algorithms:
  - Curve A above curve B means algorithm A better than algorithm B
- Measure for comparing models
  - Area under ROC curve (AUC)



Online Lectures

- For the exercise and exam, the online lectures are relevant as well

Week	Monday(Offline Lecture)	Online Lecture (see Ilias Course)	Thursday (Exercise)
01.09.2025	no lecture		Introduction to Python (13:45–15:15)
08.09.2025	Introduction to Data Mining (PDF, 3 MB)		Intro
15.09.2025	Preprocessing (PDF, 2 MB)		Preprocessing
22.09.2025	Classification 1	Nearest Centroids	Classification 1
29.09.2025	Classification 2	Comparing Classifiers	Classification 2
06.10.2025	Regression	Ensembles	Regression
13.10.2025	Clustering and Anomalies	Hierarchical Clustering	Clustering
20.10.2025	Feedback on project outlines	Time Series	Time Series
27.10.2025	Association Analysis and Subgroup Discovery	Multi Modal Data	Association Analysis
03.11.2025	Project feedback session		Project Work
10.11.2025	Project feedback session		Project Work
17.11.2025	Project feedback session		Project Work
24.11.2025	Project feedback session		Project Work
01.12.2025	Q&A		**Project Presentations**

Questions?

Literature for this Slideset

- Pang-Ning Tan, Michael Steinbach, Anuj Karpatne,Vipin Kumar: Introduction to Data Mining. 2nd Edition. Pearson.

- Chapter 3: Classification
  - Chapter 3.3: Decision Tree Classifier
  - Chapter 3.4: Overfitting
- Chapter 6.10.6: Random Forests