Department of Mathematics
and Computer Science

sdu.dk

#sdudk

Introduction to Machine Learning

Richard Röttger

BMB547

2024

SDU

Table of Contents

- **Decision Trees**
- Evaluation
- Random Forests

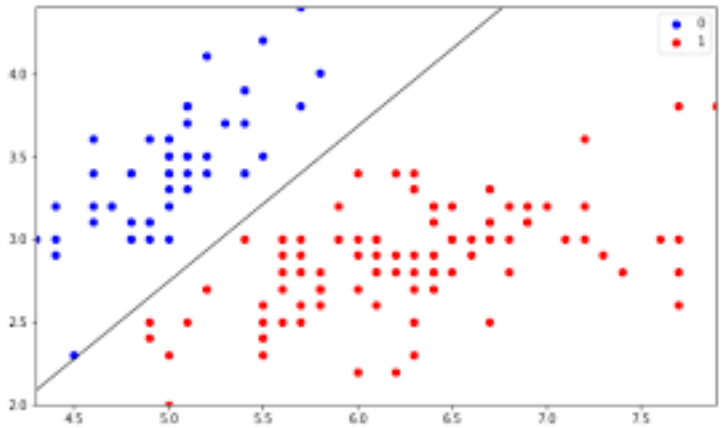
- The slides are based on the slides of Radu Ionescu, Prof. PhD. Faculty of Mathematics and Computer Science, University of Bucharest

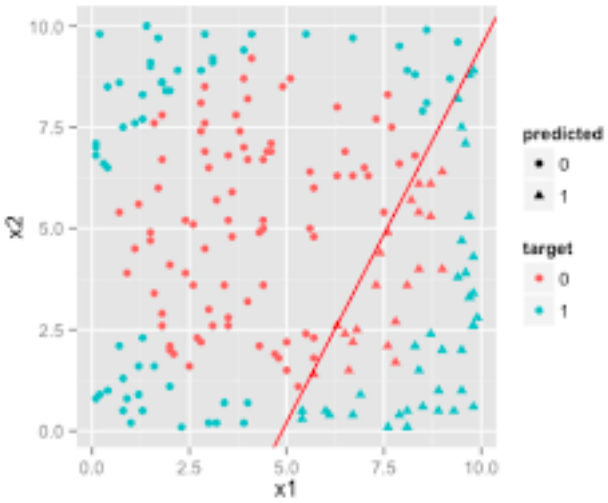
What we have learned so far

- Last lecture we have looked at classification problems
- We have learned about logistic regression

- **Advantages**
  - Easy to interpret
  - Based on what we knew already (linear regression)

- **Disadvantages**
  - Can only learn linear separations
  - Fail on more complex relationships





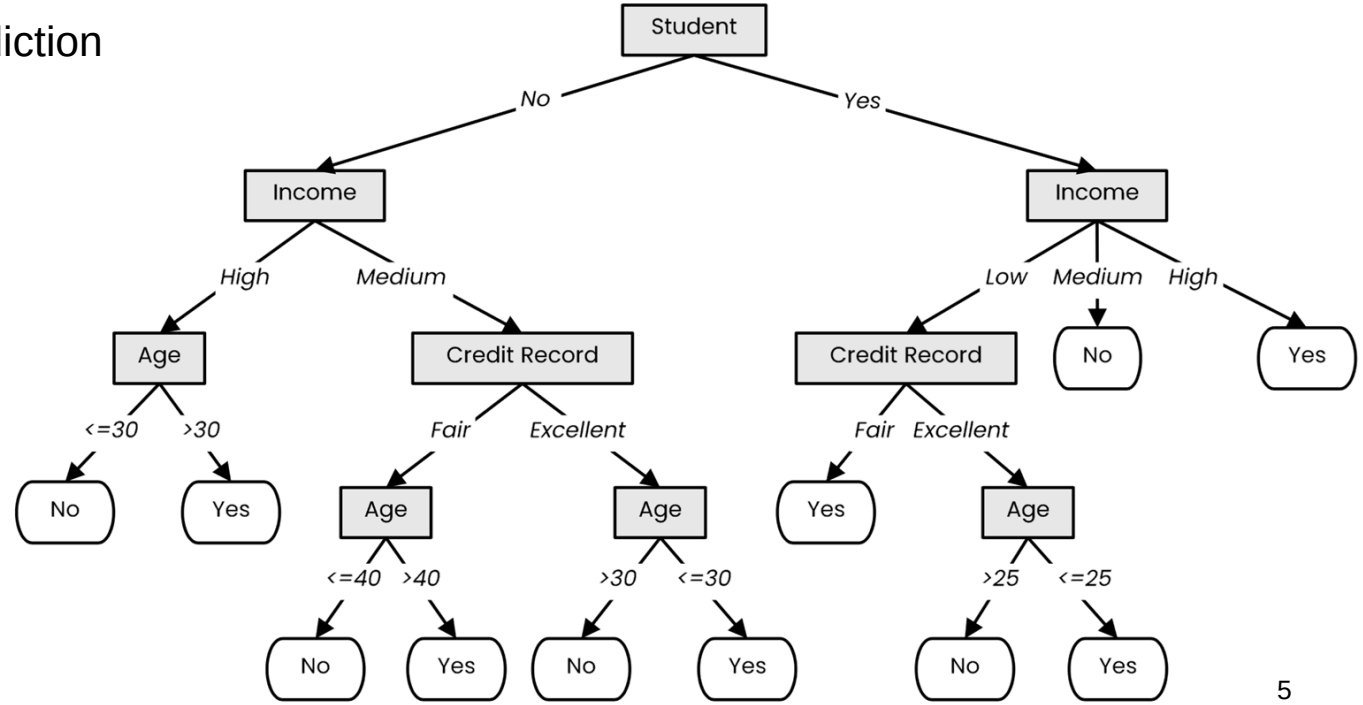
Model Tradeoff

- We want to learn more complex models

- **We are always facing trade-offs**
- Prediction accuracy versus interpretability:
  - Linear models are easy to interpret; thin-plate splines are not. Deep Neural Networks are even harder to interpret
- Good fit versus over-fit or under-fit:
  - How do we know when the fit is just right?
- Parsimony versus black-box:
  - We often prefer a simpler model involving fewer variables over a black-box predictor involving them all.

Decision Trees – What are they?

- A decision tree is a tree-like model commonly used in machine learning
- **Structure**:
  - Each internal node represents a decision or test on a specific feature or attribute
  - Each branch represents the outcome of that decision
  - Each leaf node represents the final decision or prediction
- **Advantages**:
  - Human interpretable
  - Can be sanity checked
  - Can learn more complex decision boundaries than linear models

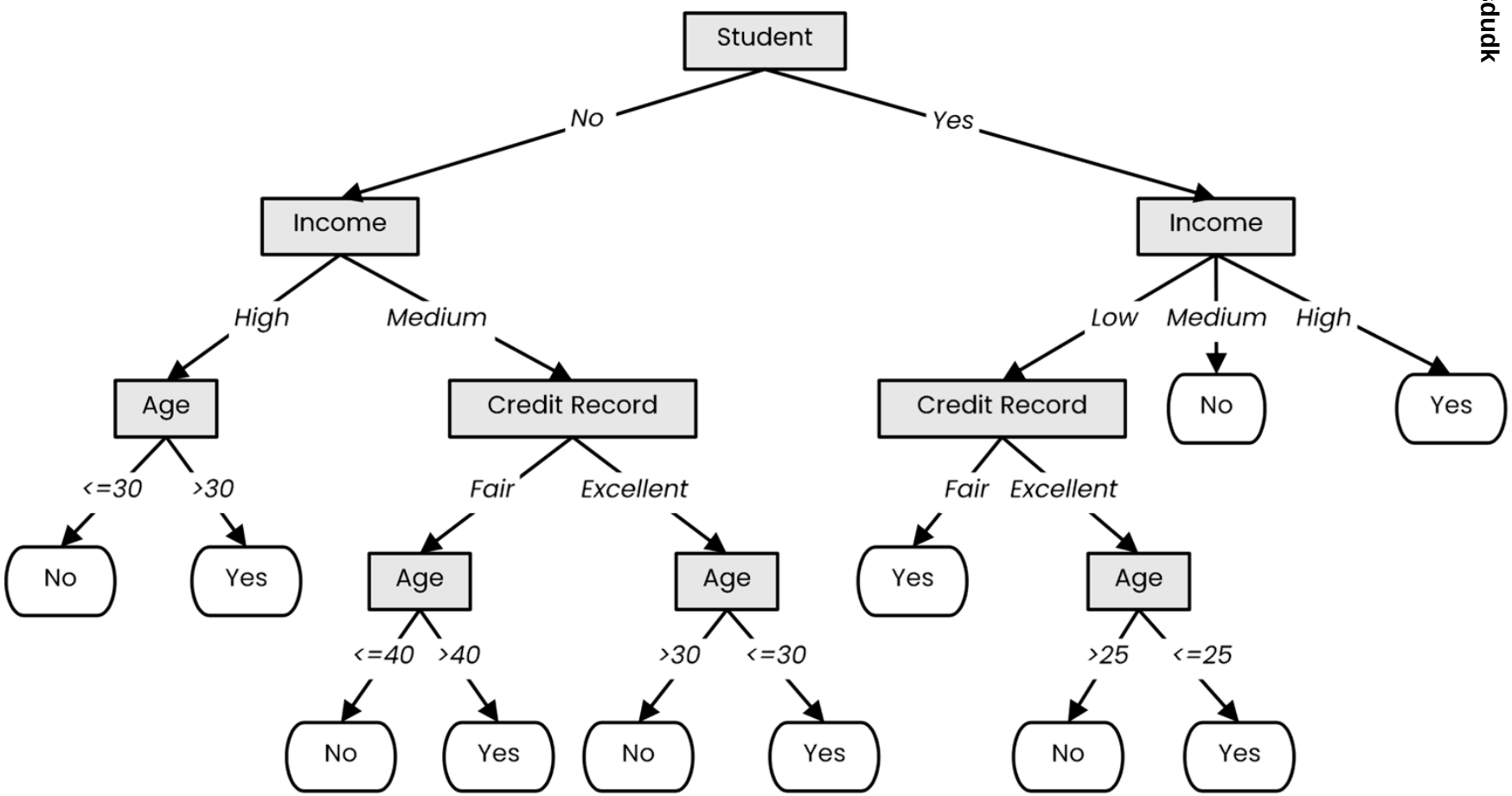


Decision Trees – How are they used?

- Lets look at two instances:

- Person A
  - Not a student
  - 50 years of age
  - Medium Income
  - Fair credit record

- Person B
  - Student
  - 25 years of age
  - Low income
  - Excellent credit record

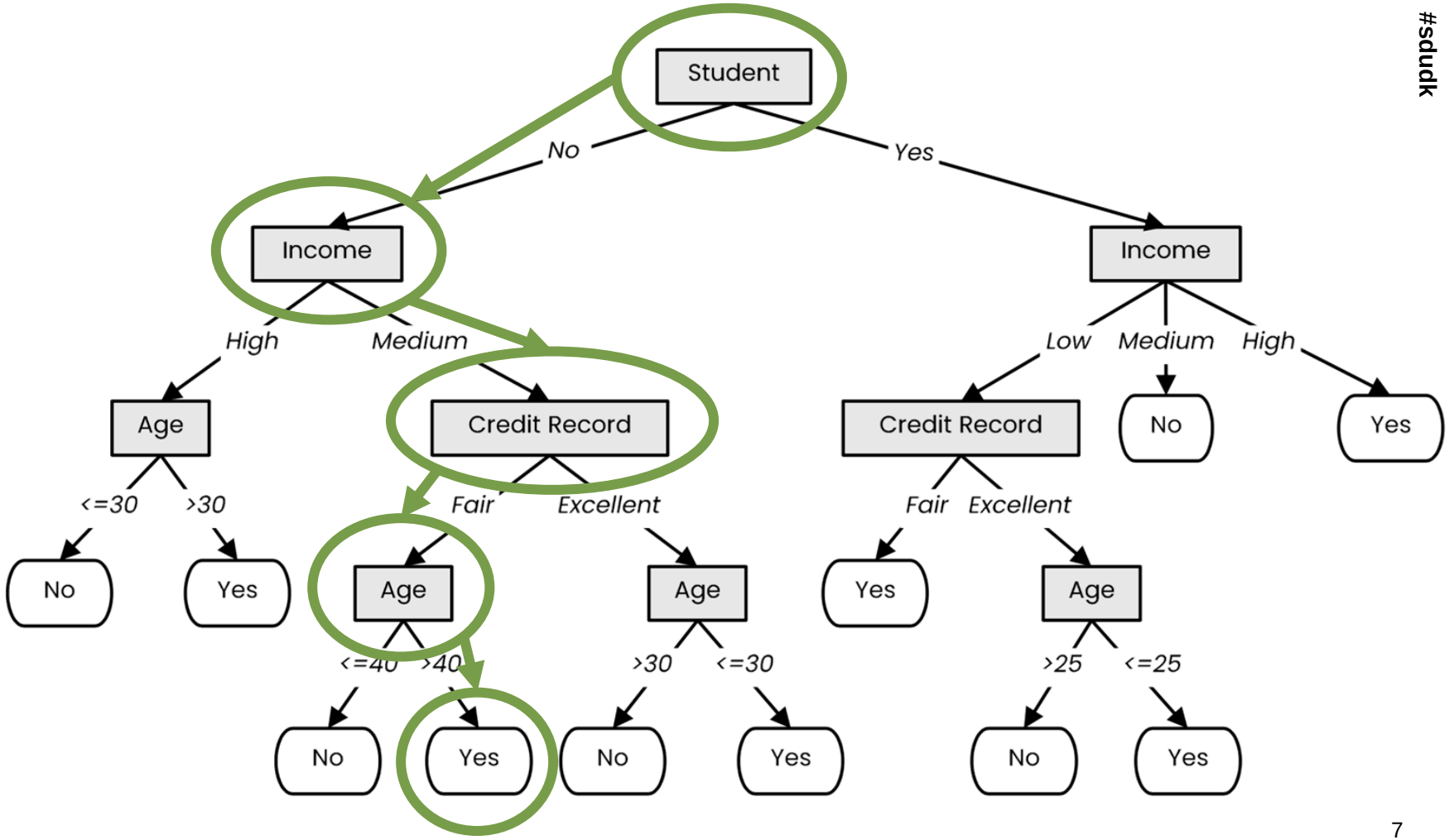


Decision Trees – How are they used?

- Lets look at two instances:

- **Person A**
  - **Not a student**
  - **50 years of age**
  - **Medium Income**
  - **Fair credit record**

- Person B
  - Student
  - 25 years of age
  - Low income
  - Excellent credit record

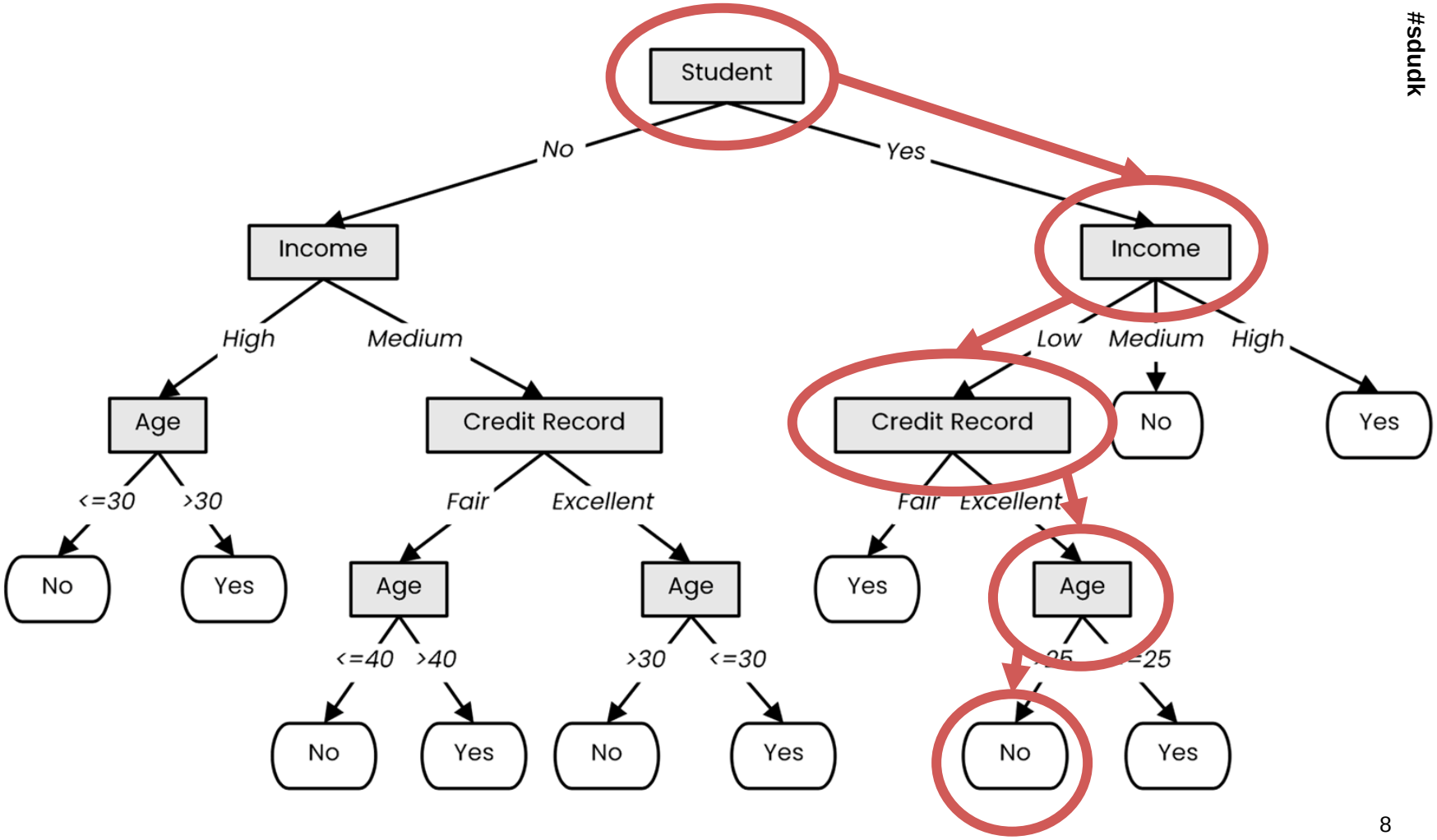


Decision Trees – How are they used?

- Lets look at two instances:

- Person A
  - Not a student
  - 50 years of age
  - Medium Income
  - Fair credit record

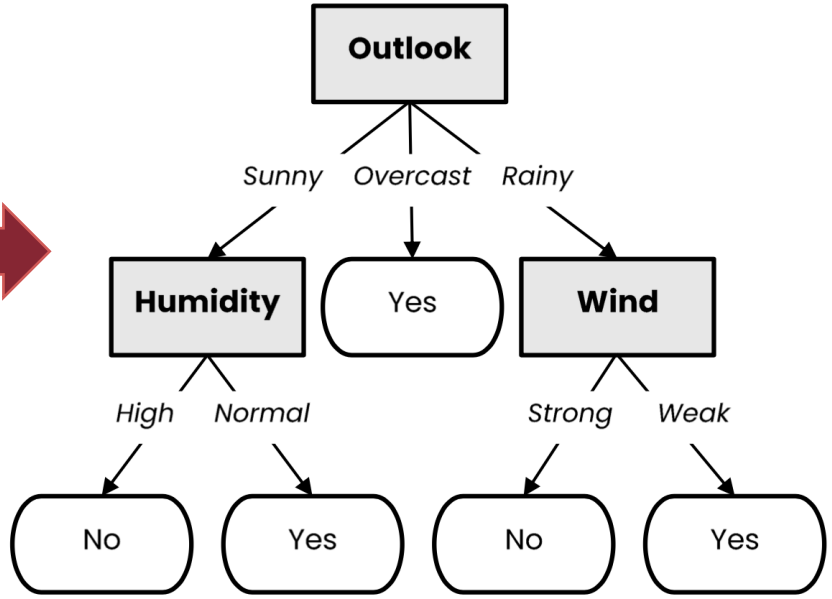
- **Person B**
  - **Student**
  - **26 years of age**
  - **Low income**
  - **Excellent credit record**



Decision Trees – How do we learn them?

- We use labeled data to obtain a suitable decision tree for future predictions
- We want a decision tree that works well on unseen data, while asking as few questions as possible

Outlook	Temperature	Humidity	Wind	Play Tennis?
Sunny	Hot	High	Weak	**No**
Sunny	Hot	High	Strong	**No**
Overcast	Hot	High	Weak	**Yes**
Rainy	Mild	High	Weak	**Yes**
Rainy	Cool	Normal	Weak	**Yes**
Rainy	Cool	Normal	Strong	**No**
Overcast	Cool	Normal	Strong	**Yes**
Sunny	Mild	High	Weak	**No**
Sunny	Cool	Normal	Weak	**Yes**
Rainy	Mild	Normal	Weak	**Yes**
Sunny	Mild	Normal	Strong	**Yes**
Overcast	Mild	High	Strong	**Yes**
Overcast	Hot	Normal	Weak	**Yes**
Rainy	Mild	High	Strong	**No**



Decision Trees – How do we learn them?

- Fundamental Idea:
  - Choose an attribute and, based on its values, split the data into smaller sets
  - Recursively repeat this step until we can surely decide the label

Outlook = *Sunny*

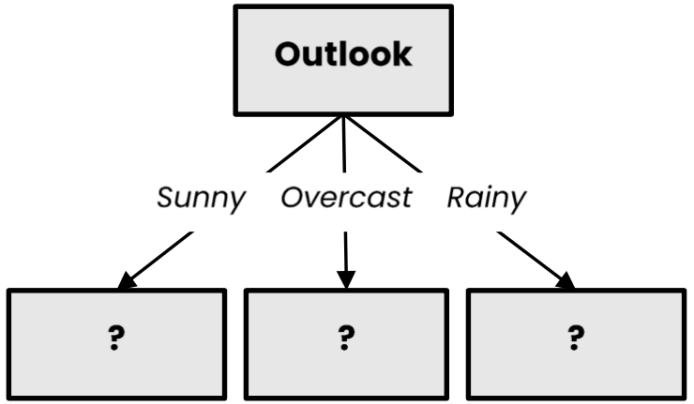
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**No**
Hot	High	Strong	**No**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**
Mild	Normal	Strong	**Yes**

Outlook = *Overcast*

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Mild	High	Strong	**Yes**
Hot	Normal	Weak	**Yes**

Outlook = *Rainy*

Temperature	Humidity	Wind	Play Tennis?
Mild	High	Weak	**Yes**
Cool	Normal	Weak	**Yes**
Cool	Normal	Strong	**No**
Mild	Normal	Weak	**Yes**
Mild	High	Strong	**No**



Decision Trees – How do we learn them?

- Fundamental Idea:
  - Choose an attribute and, based on its values, split the data into smaller sets
  - Recursively repeat this step until we can surely decide the label

Outlook = *Sunny*

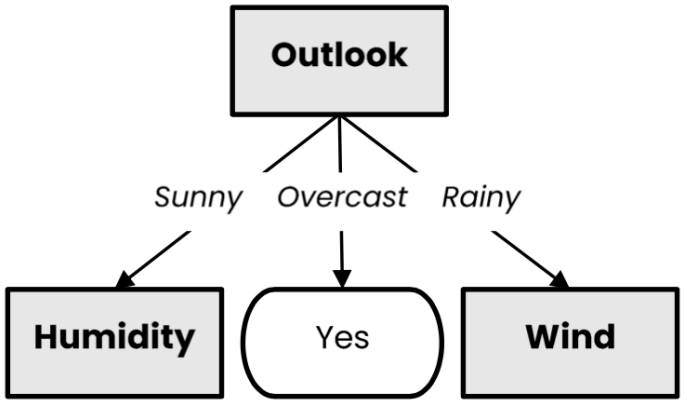
Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**No**
Hot	High	Strong	**No**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**
Mild	Normal	Strong	**Yes**

Outlook = *Overcast*

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Mild	High	Strong	**Yes**
Hot	Normal	Weak	**Yes**

Outlook = *Rainy*

Temperature	Humidity	Wind	Play Tennis?
Mild	High	Weak	**Yes**
Cool	Normal	Weak	**Yes**
Cool	Normal	Strong	**No**
Mild	Normal	Weak	**Yes**
Mild	High	Strong	**No**



Decision Trees – How do we learn them?

- Fundamental Idea:
  - Choose an attribute and, based on its values, split the data into smaller sets
  - Recursively repeat this step until we can surely decide the label

Outlook = *Sunny*

Humidity = High

Temperature	Wind	Play Tennis?
Hot	Weak	**No**
Hot	Strong	**No**
Mild	Weak	**No**

Humidity = Normal

Temperature	Wind	Play Tennis?
Cool	Weak	**Yes**
Mild	Strong	**Yes**

Outlook = *Overcast*

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Mild	High	Strong	**Yes**
Hot	Normal	Weak	**Yes**

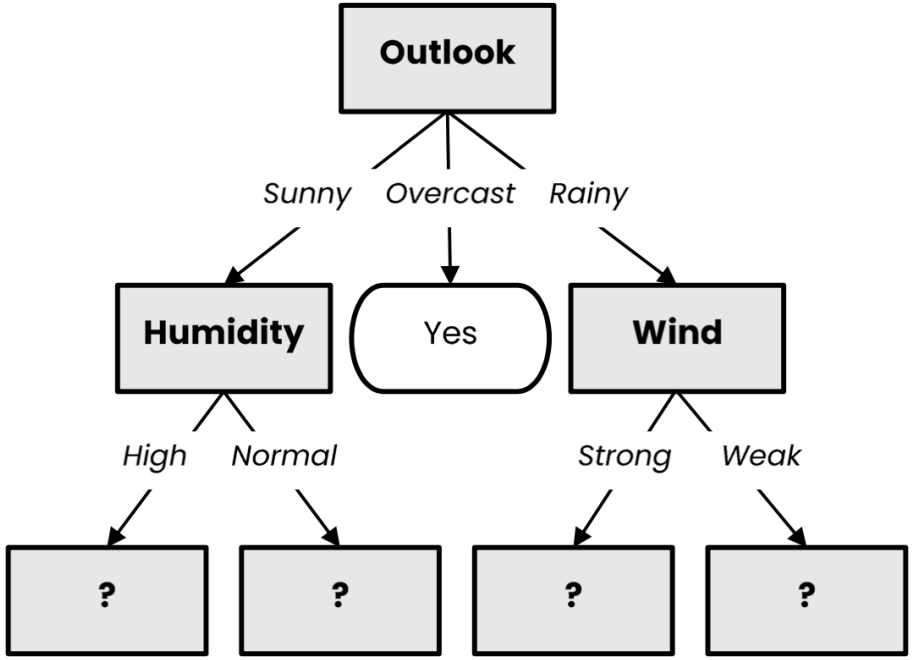
Outlook = *Rainy*

Wind = Strong

Temperature	Humidity	Play Tennis?
Cool	Normal	**No**
Mild	High	**No**

Wind = Weak

Temperature	Humidity	Play Tennis?
Mild	High	**Yes**
Cool	Normal	**Yes**
Mild	Normal	**Yes**



Decision Trees – How do we learn them?

- Fundamental Idea:
  - Choose an attribute and, based on its values, split the data into smaller sets
  - Recursively repeat this step until we can surely decide the label

Outlook = *Sunny*

Humidity = High

Temperature	Wind	Play Tennis?
Hot	Weak	**No**
Hot	Strong	**No**
Mild	Weak	**No**

Humidity = Normal

Temperature	Wind	Play Tennis?
Cool	Weak	**Yes**
Mild	Strong	**Yes**

Outlook = *Overcast*

Temperature	Humidity	Wind	Play Tennis?
Hot	High	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Mild	High	Strong	**Yes**
Hot	Normal	Weak	**Yes**

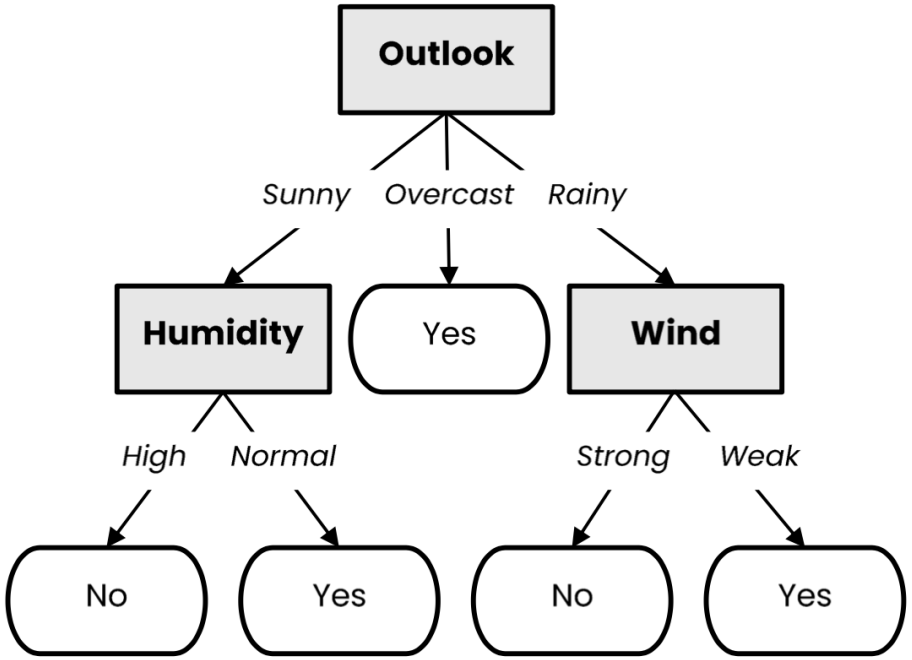
Outlook = *Rainy*

Wind = Strong

Temperature	Humidity	Play Tennis?
Cool	Normal	**No**
Mild	High	**No**

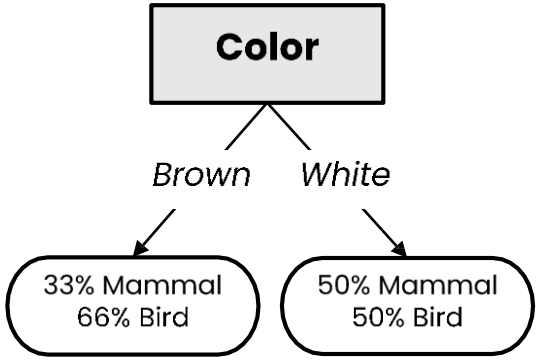
Wind = Weak

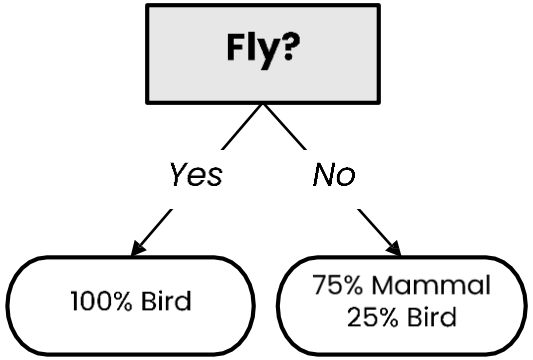
Temperature	Humidity	Play Tennis?
Mild	High	**Yes**
Cool	Normal	**Yes**
Mild	Normal	**Yes**



What is a good Attribute?

Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**

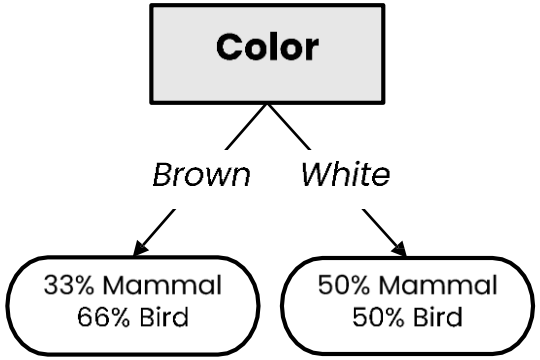


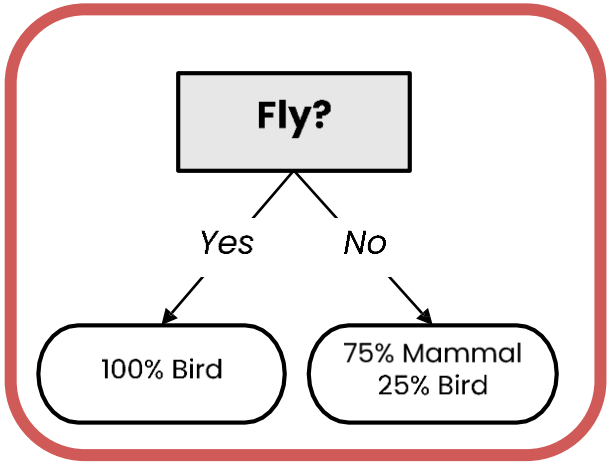


What is a good Attribute?

- Which attribute provides better splitting? And Why?
  - Because the resulting subsets are more pure
  - Knowing the value of this attribute gives us more information about the label
  - The entropy of the subsets is lower

Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**





What is a good Attribute?

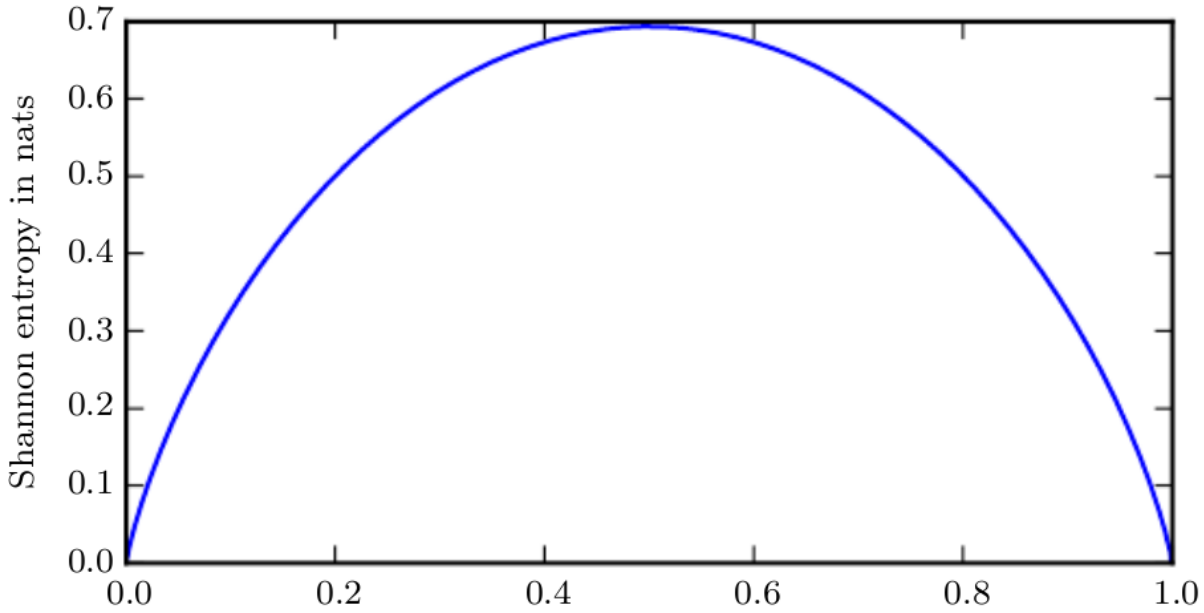
- We need a measure indicating the purity of a split / node
- It shall fulfill certain criteria
  - We should get the worst value, when all outcomes are equally likely, i.e., $\left(\frac{1}{2},\frac{1}{2}\right)$ for two classes, or $\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$ for three classes, etc.
  - It should be best, when the node is pure, e.g., $(1,0)$ or $(0,1,0)$ etc.

What is a good Attribute? -> Entropy

- Entropy H is a measure for the randomness of a distribution
- For a set of samples X with k classes, we have:

$$H(X) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

- Here, p_i Indicates the proportions of elements of class i



Just FYI: The Entropy is often used in Information theory. There is an extensive body of research involved in this field. But in very short, the intuition is that the occurrence of an extremely unlikely event is more informative than the occurrence of a very likely event
To quantify this intuition and need a function fulfilling:

- Likely events should have low information content
- Extreme case: Guaranteed events shall have no information content
- Less likely events should have higher information content.
- Independent events should have additive information.

Overall, the log function fulfills these requirements, and the expectation value over all events is then the entropy, as presented here.

Information Gain

- How do we use this to evaluate a node?
- We compute the information gain of a split
  - This is the reduction in entropy due to the splitting

- Let's have a look at the split at an attribute a

$$\text{gain}(X, a) = H(X) - \sum_{v \in \text{Values}(a)} \frac{|X_v|}{|X|} \cdot H(X_v)$$

- Here, X_v is the subset of X for which have the value v at attribute a

Information Gain

- How do we use this to evaluate a node?
- We compute the information gain of a split
  - This is the reduction in entropy due to the splitting
- Let's have a look at the split at an attribute a

"Old Entropy" of the node before the split

Proportion of the Elements with value v

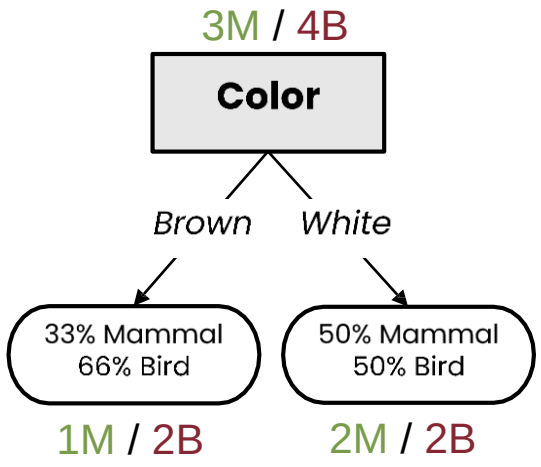
$$\text{gain}(X, a) = H(X) - \sum_{v \in \text{Values}(a)} \frac{|X_v|}{|X|} \cdot H(X_v)$$

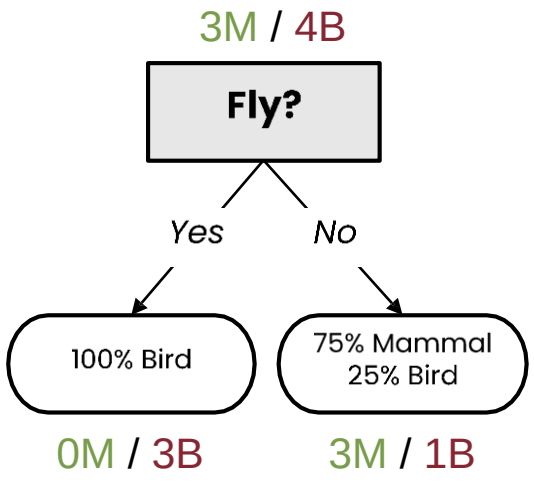
Entropy of the new nodes after the split

- Here, X_v is the subset of X for which have the value v at attribute a

Example

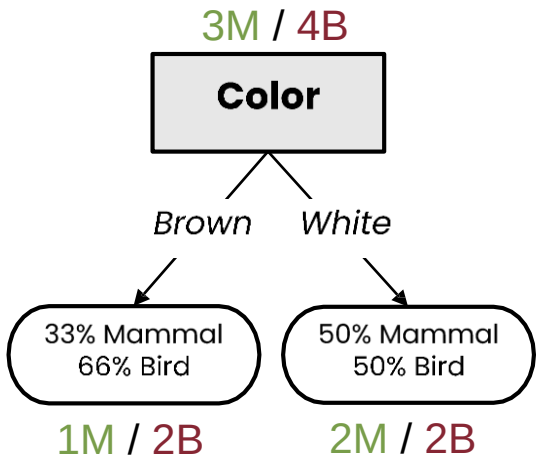
Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**





sdu.dk
#sdudk

Example

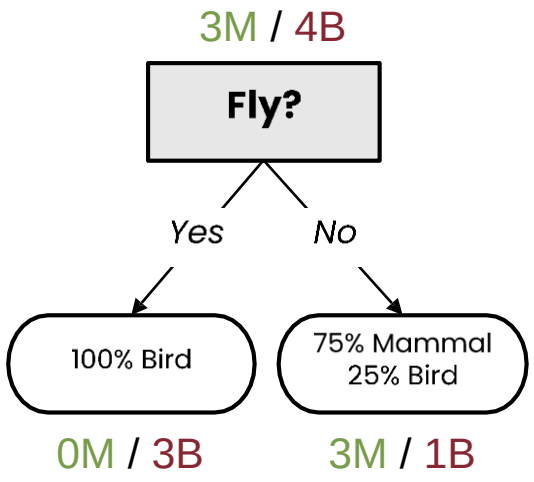


Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**

- Entropy before the split

$$H(X) = -p_B \log_2 p_M - p_B \log_2 p_B$$

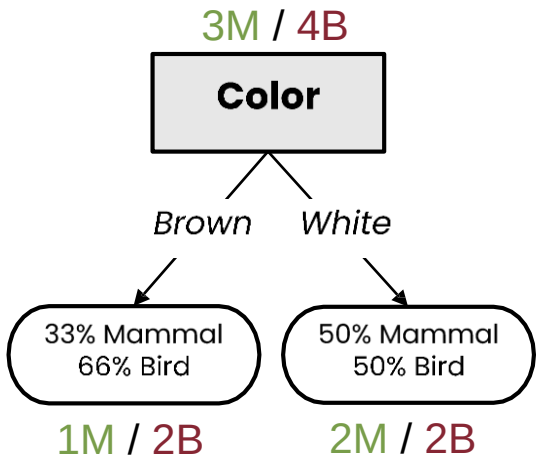
$$H(X) = -\frac{3}{7}\log_2 \frac{3}{7} - \frac{4}{7}\log_2 \frac{4}{7} = 0.985$$



sdu.dk
#sdudk

Example

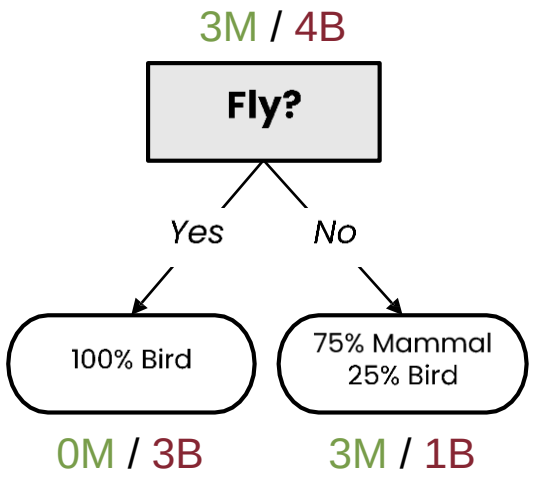
Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**



- Entropy before the split

$$H(X) = -p_B \log_2 p_M - p_B \log_2 p_B$$

$$H(X) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$



- Entropy Color c is brown b

$$H(X_{c=b}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$$

- Entropy Color c is white w

$$H(X_{c=b}) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

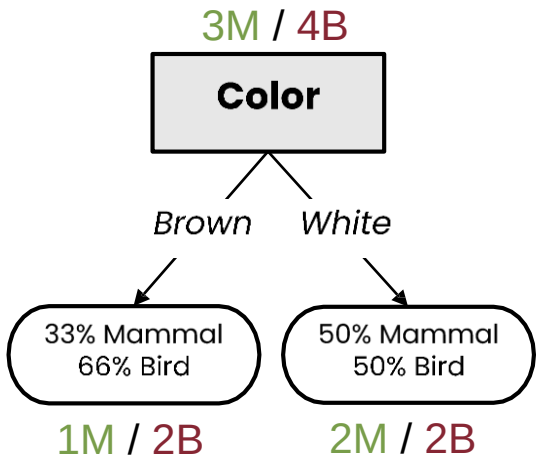
- Gain for split at color c:

$$\text{gain}(X, c) = 0.985 - \frac{3}{7}0.918 - \frac{4}{7}1 = 0.020$$

sdu.dk
#sdudk

Example

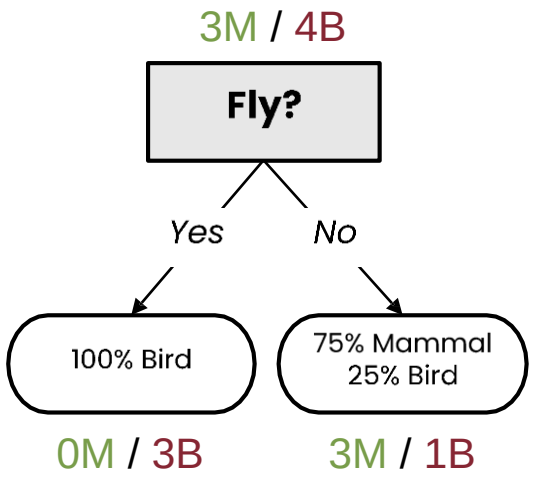
Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**



- Entropy before the split

$$H(X) = -p_B \log_2 p_M - p_B \log_2 p_B$$

$$H(X) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$



- Entropy Color c is brown b

$$H(X_{c=b}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$$

- Entropy Color c is white w

$$H(X_{c=b}) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

- Gain for split at color c:

$$\text{gain}(X, c) = 0.985 - \frac{3}{7}0.918 - \frac{4}{7}1 = 0.020$$

- Entropy Fly f is yes y

$$H(X_{f=y}) = -0 - 1\log_2 1 = 0$$

- Entropy Fly f is no n

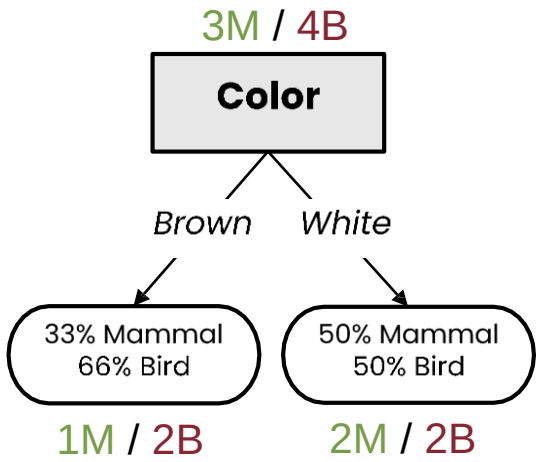
$$H(X_{f=n}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$$

- Gain for split at Fly f:

$$\text{gain}(X, f) = 0.985 - \frac{3}{7}0 - \frac{4}{7}0.811 = 0.521$$

Example

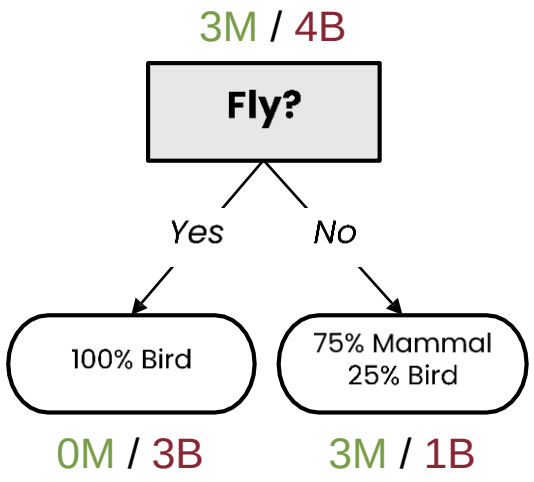
Does it fly?	Color	Class
No	Brown	**Mammal**
No	White	**Mammal**
Yes	Brown	**Bird**
Yes	White	**Bird**
No	White	**Mammal**
No	Brown	**Bird**
Yes	White	**Bird**



- Entropy before the split

$$H(X) = -p_B \log_2 p_M - p_B \log_2 p_B$$

$$H(X) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$



- Entropy Color c is brown b

$$H(X_{c=b}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$$

- Entropy Color c is white w

$$H(X_{c=b}) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

- Gain for split at color c:

$$\text{gain}(X, c) = 0.985 - \frac{3}{7}0.918 - \frac{4}{7}1 = 0.020$$

- Entropy Fly f is yes y

$$H(X_{f=y}) = -0 - 1\log_2 1 = 0$$

- Entropy Fly f is no n

$$H(X_{f=n}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$$

- Gain for split at Fly f:

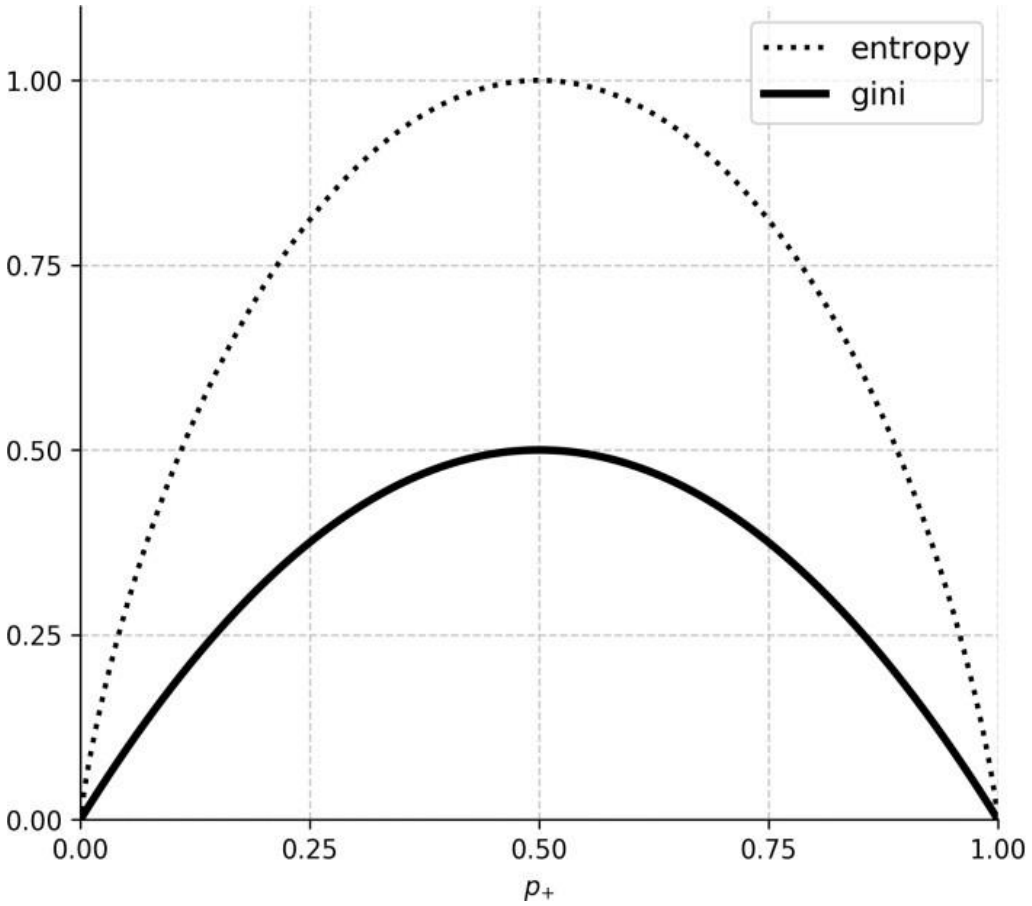
$$\boxed{\text{gain}(X, f) = 0.985 - \frac{3}{7}0 - \frac{4}{7}0.811 = 0.521}$$

Gini Impurity

- Gini impurity measures how often a randomly chosen example would be incorrectly labeled if it was randomly labeled according to the label distribution

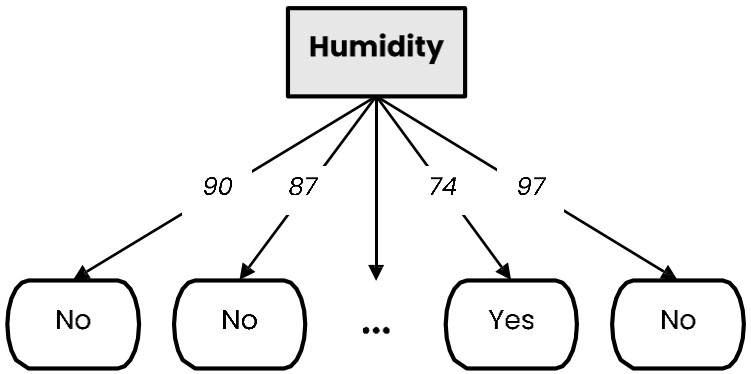
$$gini(X) = 1 - \sum_{i=1}^{k} p_i^2$$

- It is easier to compute
- Leaves the log out
- Essentially delivers the same results



Work with continuous variables

- Split at every possible numerical value infeasible
- Better solution
  - Sort according to the attribute
  - Compute the gain for every potential split



Humidity	Play Tennis?
90	No
87	No
93	Yes
89	Yes
79	Yes
59	No
77	Yes
91	No
68	Yes
80	Yes
72	Yes
96	Yes
74	Yes
97	No

Sort

Humidity	Play Tennis?
59	No
68	Yes
72	Yes
74	Yes
77	Yes
79	Yes
80	Yes
87	No
89	Yes
90	No
91	No
93	Yes
96	Yes
97	No

Mean of each consecutive pair

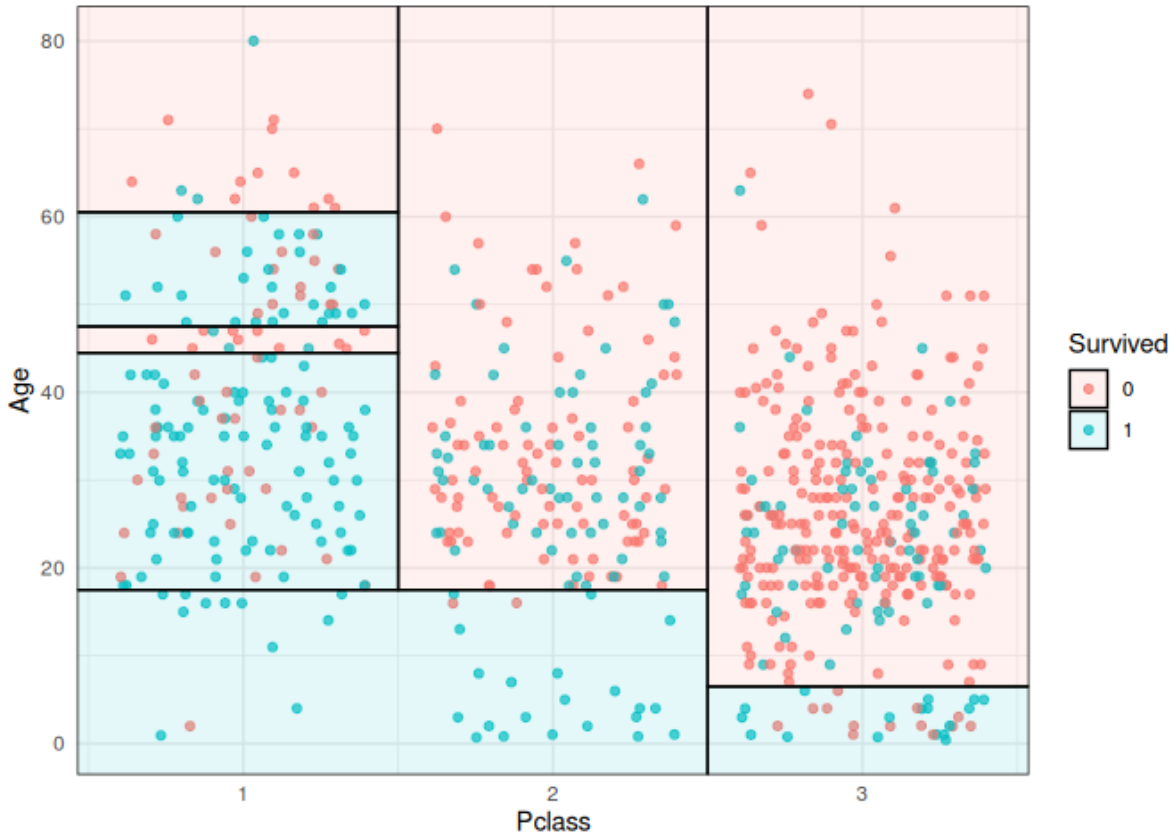
Candidate split values
63
70
73
75.5
78
79.5
83.5
88
89.5
90.5
92
94.5
96.5

Gain for every candidate

Information gain
0.113
0.01
0.0004
0.015
0.045
[illegible]
0.152
[illegible]
0.102
0.025
0.0004
0.01
0.113

Decision Boundary

- Decision Trees can learn complex decision boundaries
- They are perpendicular to the axis
- Each decision splits the data into parts of the axis



When to stop?

- We have a measure how to split based on the best attribute
- But when shall we stop?
- Decision tress can grow very very deep
- **They are prone to overfitting**

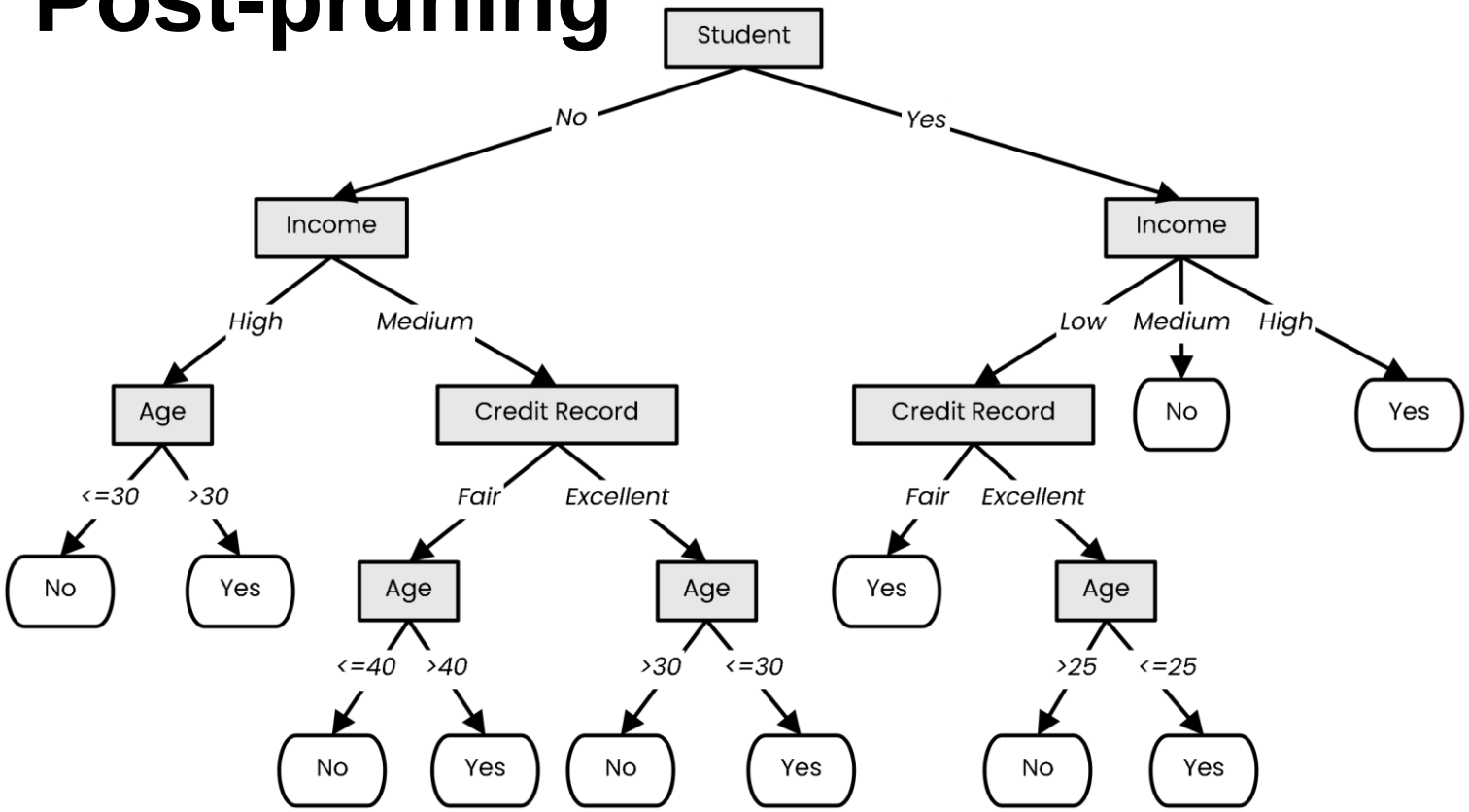
- **Tree-Pruning**
  - Pre-pruning: Stop the tree building algorithm before it fully classifies the data
  - Post-pruning: Build the complete tree, then replace some non-leaf nodes with leaf nodes if this improves validation error

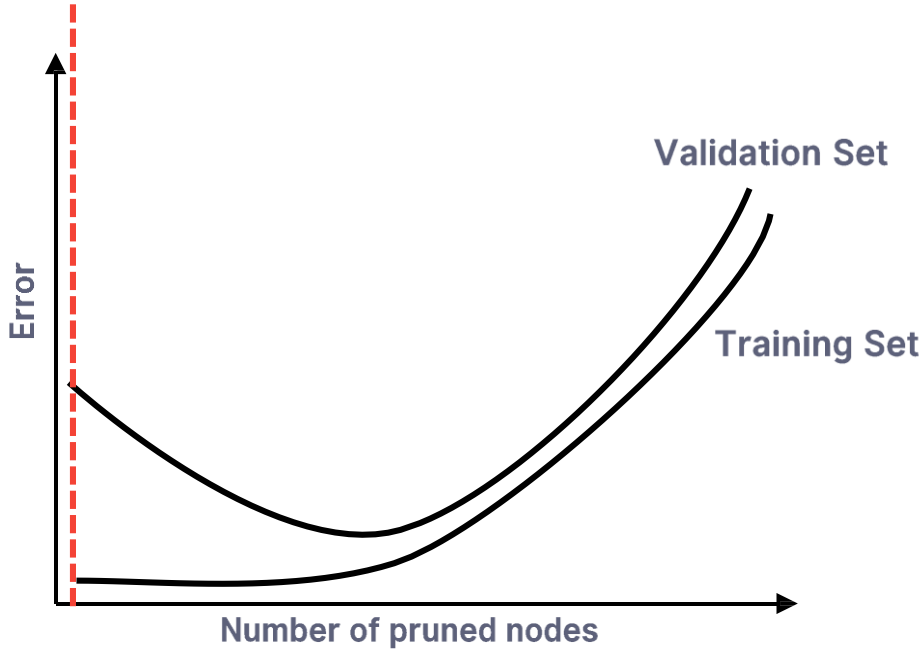
Pre-Pruning

- Pre-pruning implies early stopping:
  - If some condition is met, the current node will not be split, even if it is not 100% pure
  - It will become a leaf node with the label of the majority class in the current set
  - (the class distribution could be used as prediction confidence)

- Common stopping criteria include setting a threshold on:
  - Entropy (or Gini Impurity) of the current set
  - Number of samples in the current set
  - Gain of the best-splitting attribute
  - Depth of the tree

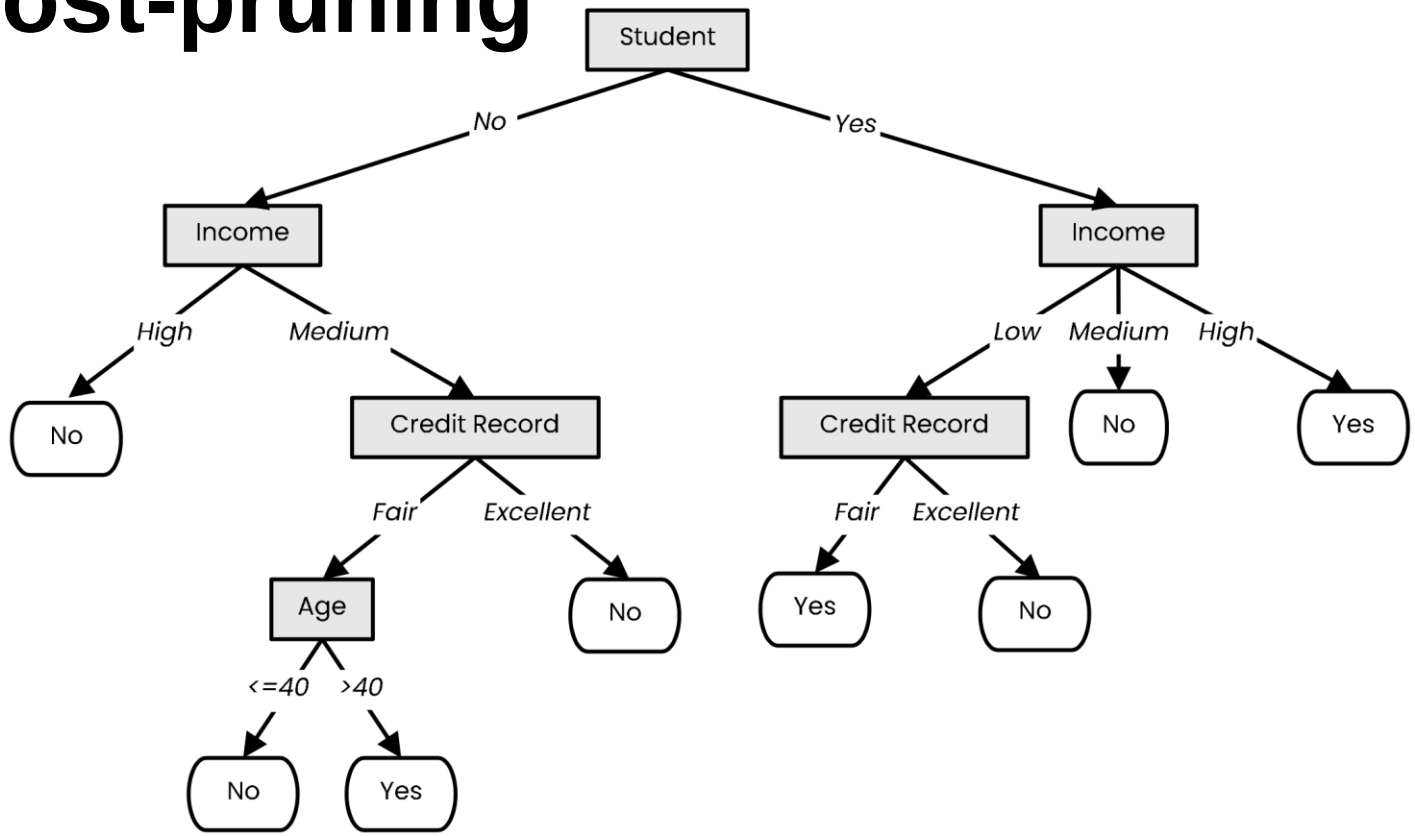
Post-pruning

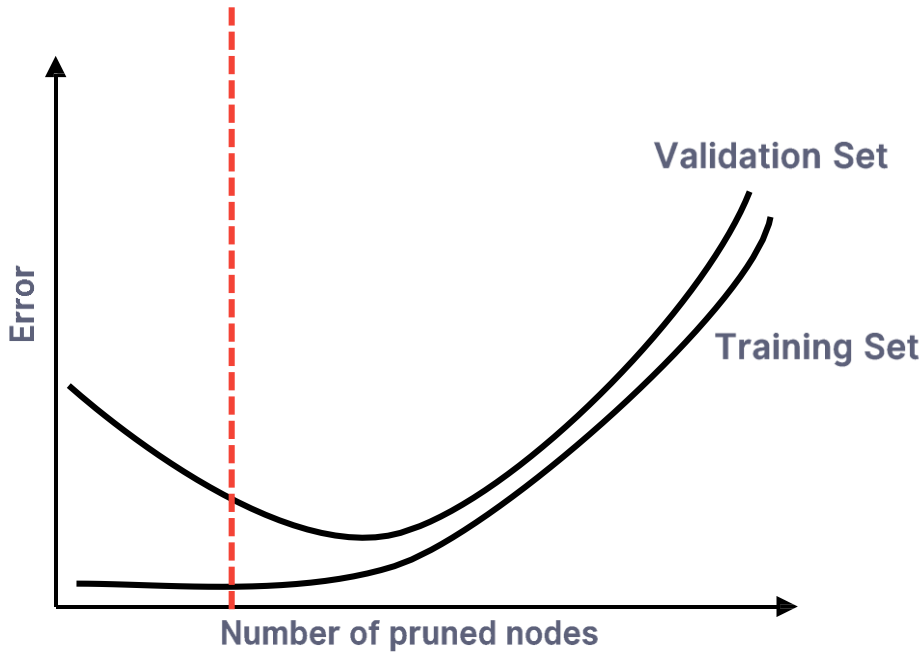




- Prune nodes in a bottom-up manner, if it decreases validation error

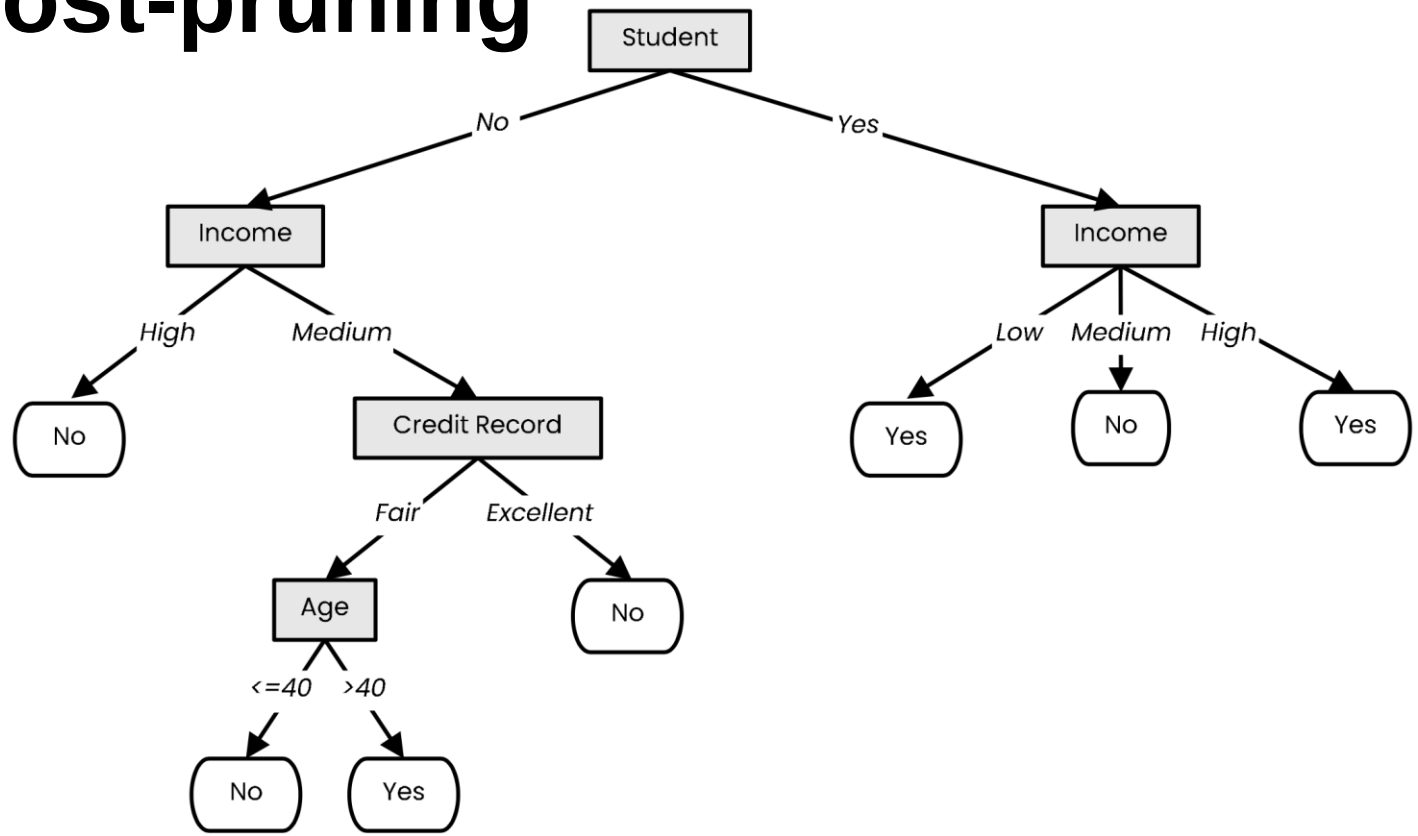
Post-pruning

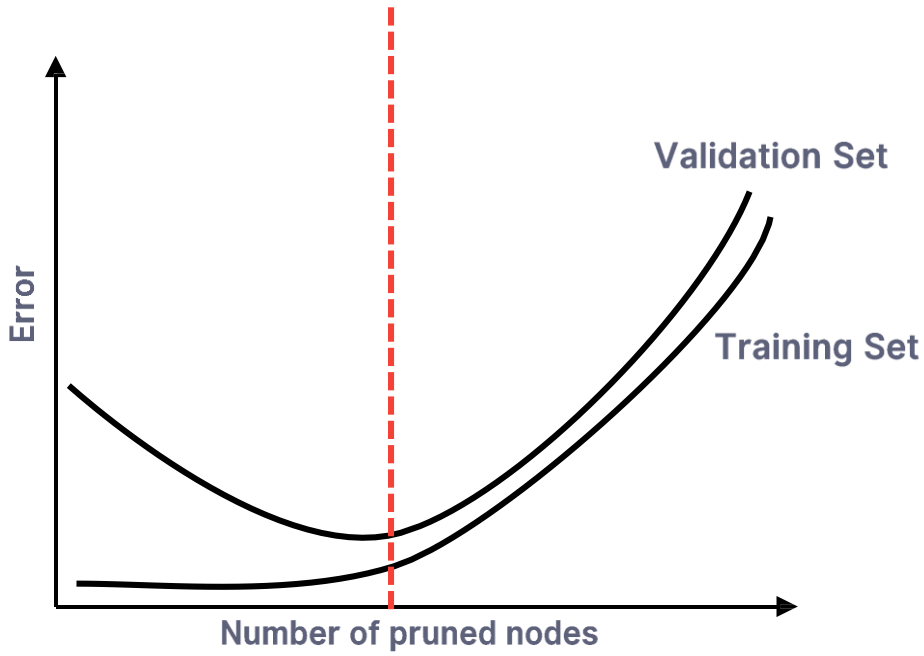




- Prune nodes in a bottom-up manner, if it decreases validation error

Post-pruning





- Prune nodes in a bottom-up manner, if it decreases validation error

An Example in R

- Let's look at a small example in R
- The data: Titanic Survivals
  - Passenger Class
  - Sex
  - Age
  - Number of Siblings on board
  - Number parents on board
  - Where did they embark
- After cleaning, removing NA's and converting data types to factors and numerical, we have the following data:
  - 1043 passengers, 59% died

```
> head(clean_titanic)
     pclass survived    sex age sibsp parch     fare embarked
57    Upper Survived   male  36     1     2 120.0000        S
774   Lower     Died   male  42     0     0   8.6625        S
920   Lower     Died   male  18     0     0   7.2292        C
430  Middle     Died   male  44     0     0  13.0000        S
1012  Lower     Died female  19     1     0  16.1000        S
476  Middle     Died female  26     1     1  26.0000        S
```

An Example in R

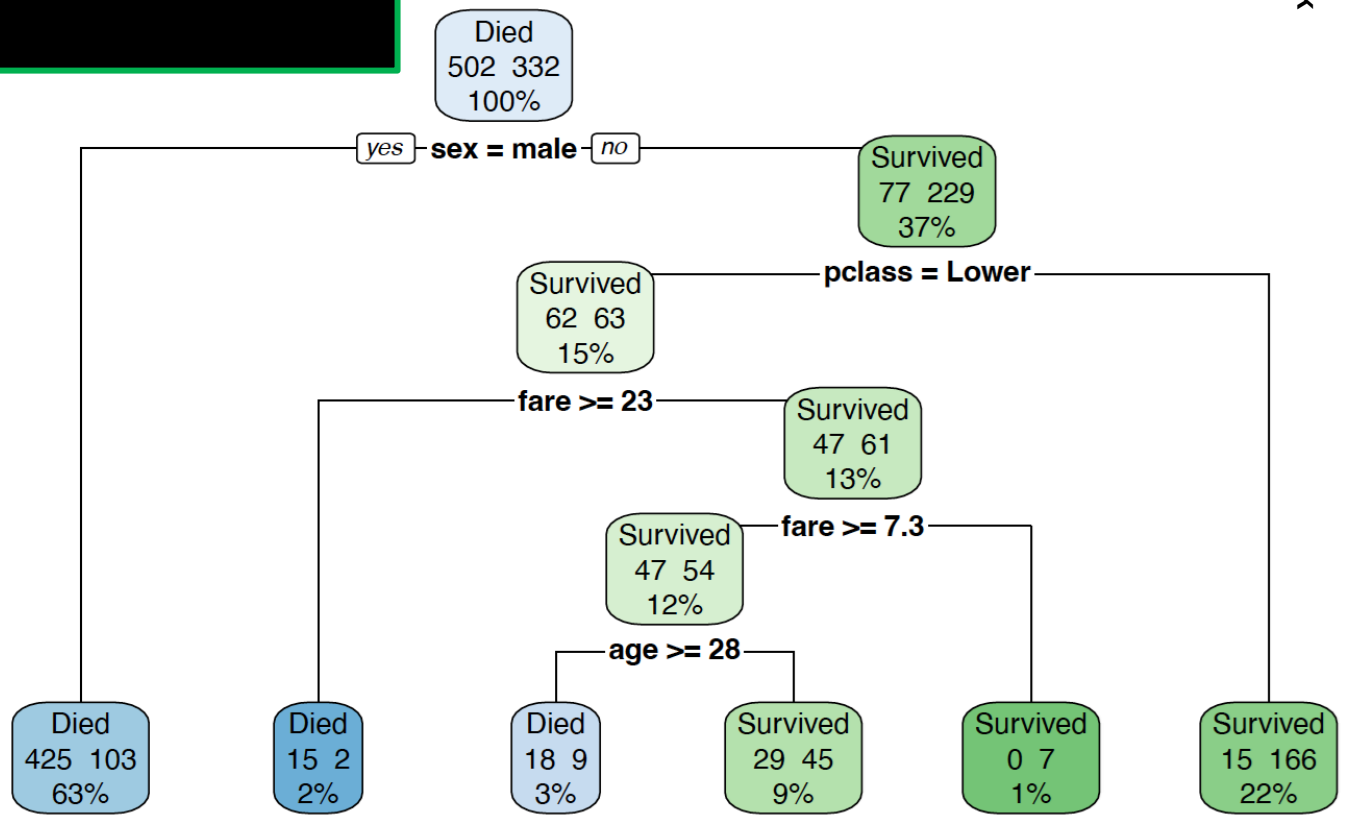
```
> library(rpart)
> library(rpart.plot)
>
> fit <- rpart(survived~., data = data_train, method = 'class', )
> summary(fit)
Call:
rpart(formula = survived ~ ., data = data_train, method = "class")
  n= 834

          CP nsplit rel error    xerror       xstd
1 0.45783133      0 1.0000000 1.0000000 0.04257942
2 0.01957831      1 0.5421687 0.5421687 0.03578525
3 0.01355422      3 0.5030120 0.5331325 0.03556708
4 0.01000000      5 0.4759036 0.5421687 0.03578525

Variable importance
     sex     fare   pclass    parch      age    sibsp embarked
      57       15       12        8        5        2        1
```

- We have split data into test and validation
- The package `rpart` contains decision tree learning
- Can be used exactly like other machine learning approaches
  - Requires a formula to be specified
  - The method ‘class’ refers to a classification problem (you can use trees also for regression)

An Example in R

```
> fit <- rpart(survived~., data = data_train, method = 'class', )
> summary(fit)
> rpart.plot(fit, extra=101)
```

- `rpart.plot(fit)` plots a representation of the tree
- The "`extra`" causes also the minority class to be printed (only optical sugar)



Evaluation

```
> predict_unseen <-predict(fit, data_test, type = 'class')
> table_mat <- table(data_test$survived, predict_unseen)
> table_mat
         predict_unseen
          Died Survived
  Died     106       10
  Survived  34       59
```

- We can use the function predict to check the quality of our model
- On the confusion Matrix, we see that we have:
  - 165 correctly classified instances
  - 44 incorrect classified instances
  - That corresponds to an accuracy of $\frac{165}{44+165} = 0.7895$
- When we do the same with the training data, we get an accuracy of 0.8106

Increasing the Model Complexity

```
> fit <- rpart(survived~., data = data_train, method =
'class', cp=0.00001, minsplit=2, maxdepth = 30)
```

- The complexity is mainly controlled by:
  - `cp` – that is the tree complexity, meaning that every split (and therefore increased complexity) demands a certain increase in performance. Default value is 0.01
  - `minsplit` – minimal number of elements in a node to be considered for splitting
  - `maxdepth` – limits the maximal depth of the tree

- In our case:
  - Training accuracy: 0.982 (yeah!)
  - Validation accuracy: 0.7656 (booo!)

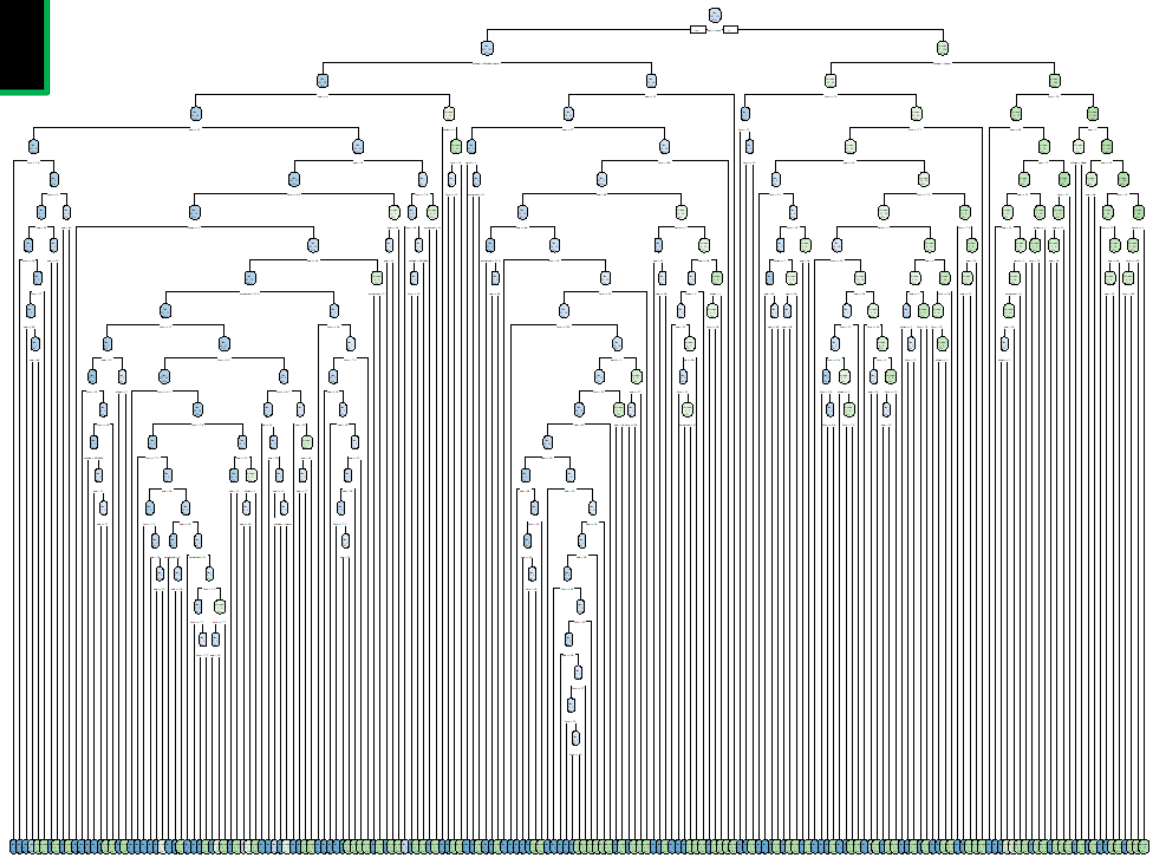
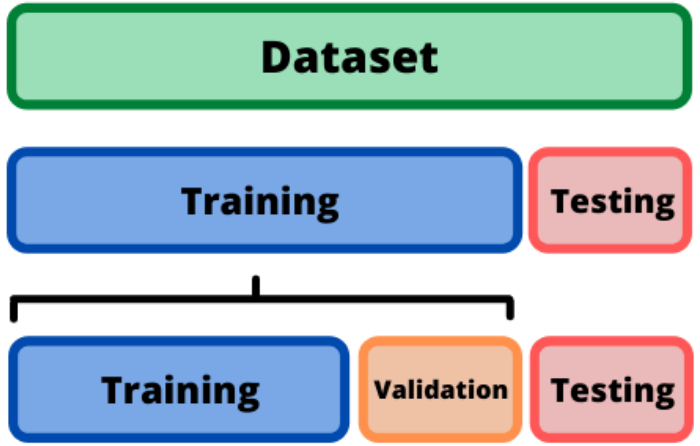


Table of Contents

- Decision Trees
- **Evaluation**
- Random Forests

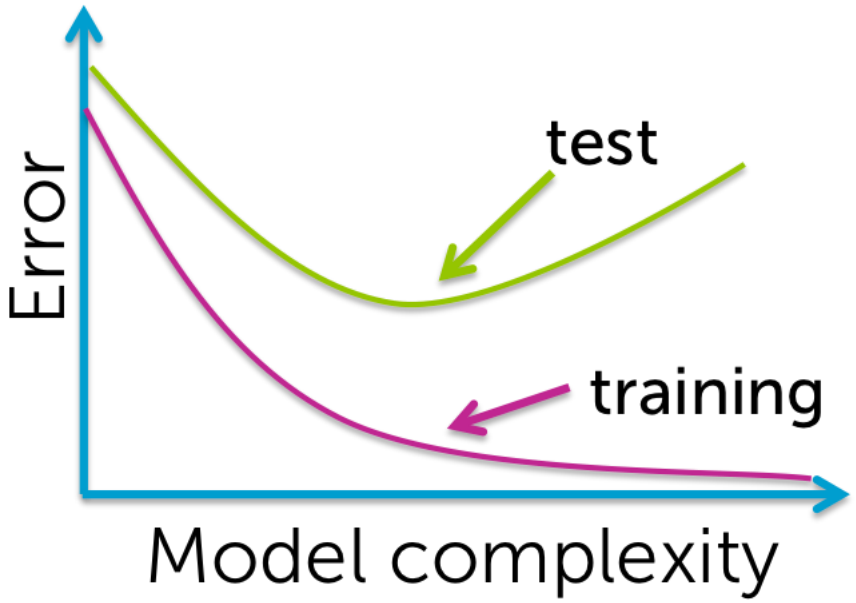
- The slides are based on the slides of Radu Ionescu, Prof. PhD. Faculty of Mathematics and Computer Science, University of Bucharest

From Last Lecture

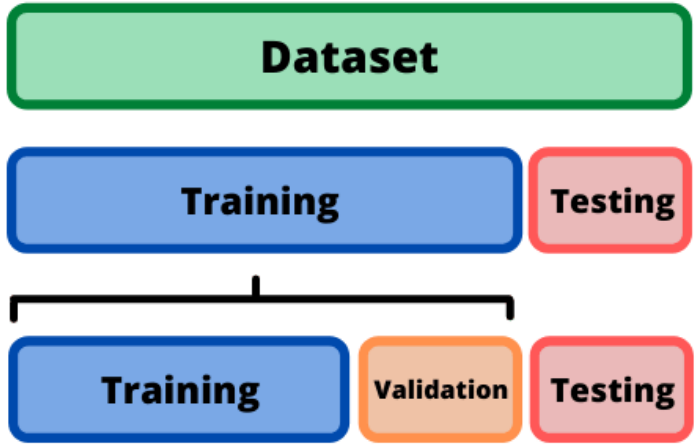


- Normally, we split our dataset into two parts
  - Training, validation and testing portion
  - Typical split: 70 / 20 / 10

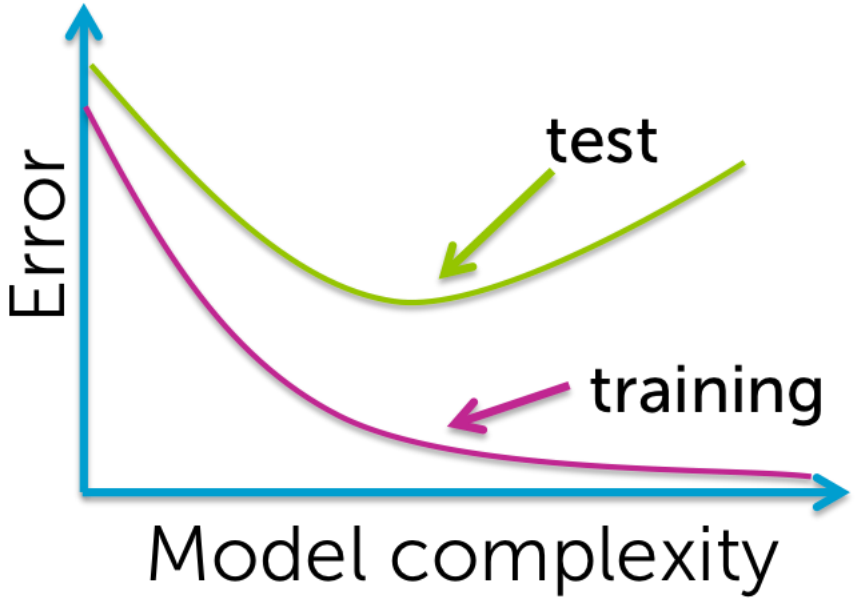
- **Central Idea**
  - We are mainly interested in the generalization error
  - That means, how well does our classifier perform on **unseen data**
  - Hard to assess, since we do not know the entirety of unseen data
  - The test set is an approximation of the error to expect



Problem with the Method

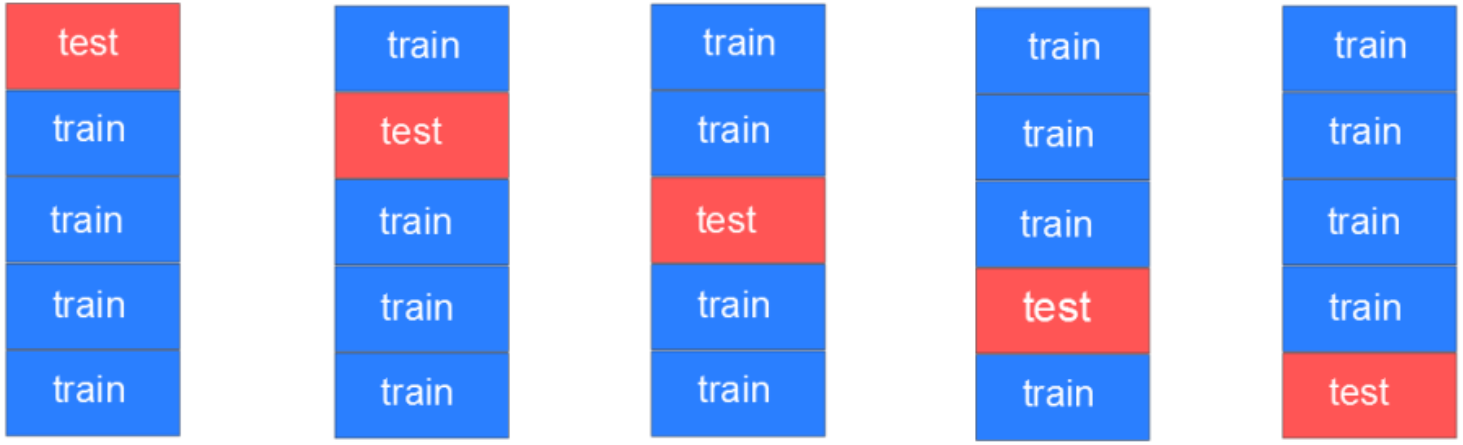


- Main concerns with this method:
  - The validation estimate of the test error can be **highly variable**, depending on precisely which observations are included in the training set and which observations are included in the validation set.
  - In the validation approach, only a **subset of the observations** - those that are included in the training set rather than in the validation set - are used to fit the model.
  - This suggests that the validation set error may tend to **overestimate** the test error for the model fit on the entire data set.



Cross-Validation

- When data set is too small, a fixed test set is problematic
- k-fold cross-validation:
  - All available data is partitioned into k groups
  - $k-1$ groups are used to train and evaluated on remaining group
  - Repeat for all k choices of held-out group
  - Performance scores from k runs are averaged



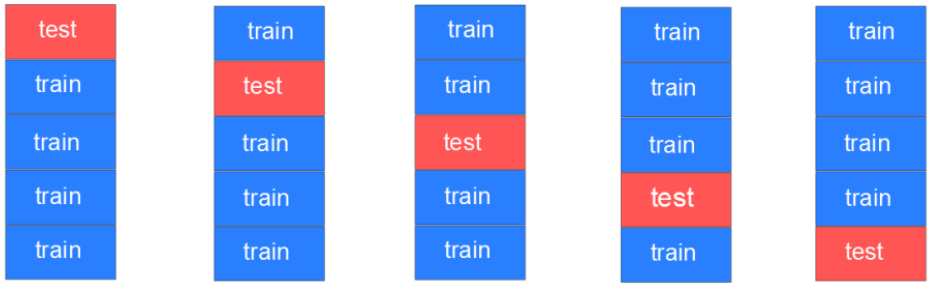
Cross-Validation

- The final cross-validation error is the average of the different runs

$$\mathrm{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{MSE}_k$$

- The number of folds is usually set to $K = 10$ or $K = 5$

Cross-Validaiton



- **Overall Procedure**:
  - Use CV to estimate the generalization error
  - Find optimal hyperparameters (setting the values for `minsplit`, `maxdepth` and `cp`)
  - After fixing the parameters, run the training on ALL data (we do not choose the best model of the CV runs)

- **Advantages**:
  - Less dependent on the goodness of the training/validation split
  - Better error estimate than using only a fixed split
- **Disadvantages**:
  - More costly to run

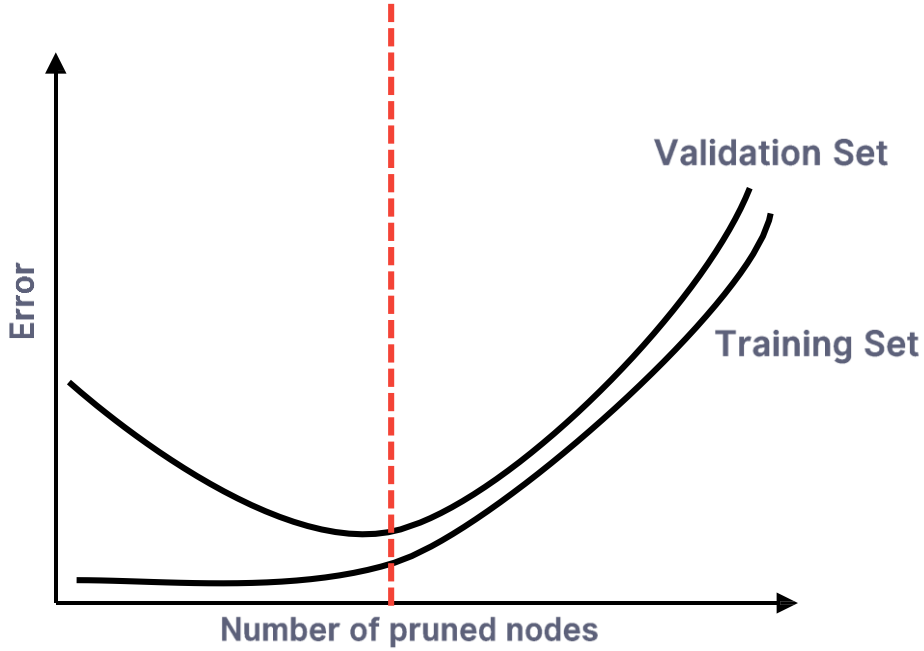


Table of Contents

- Decision Trees
- Evaluation
- **Random Forests**

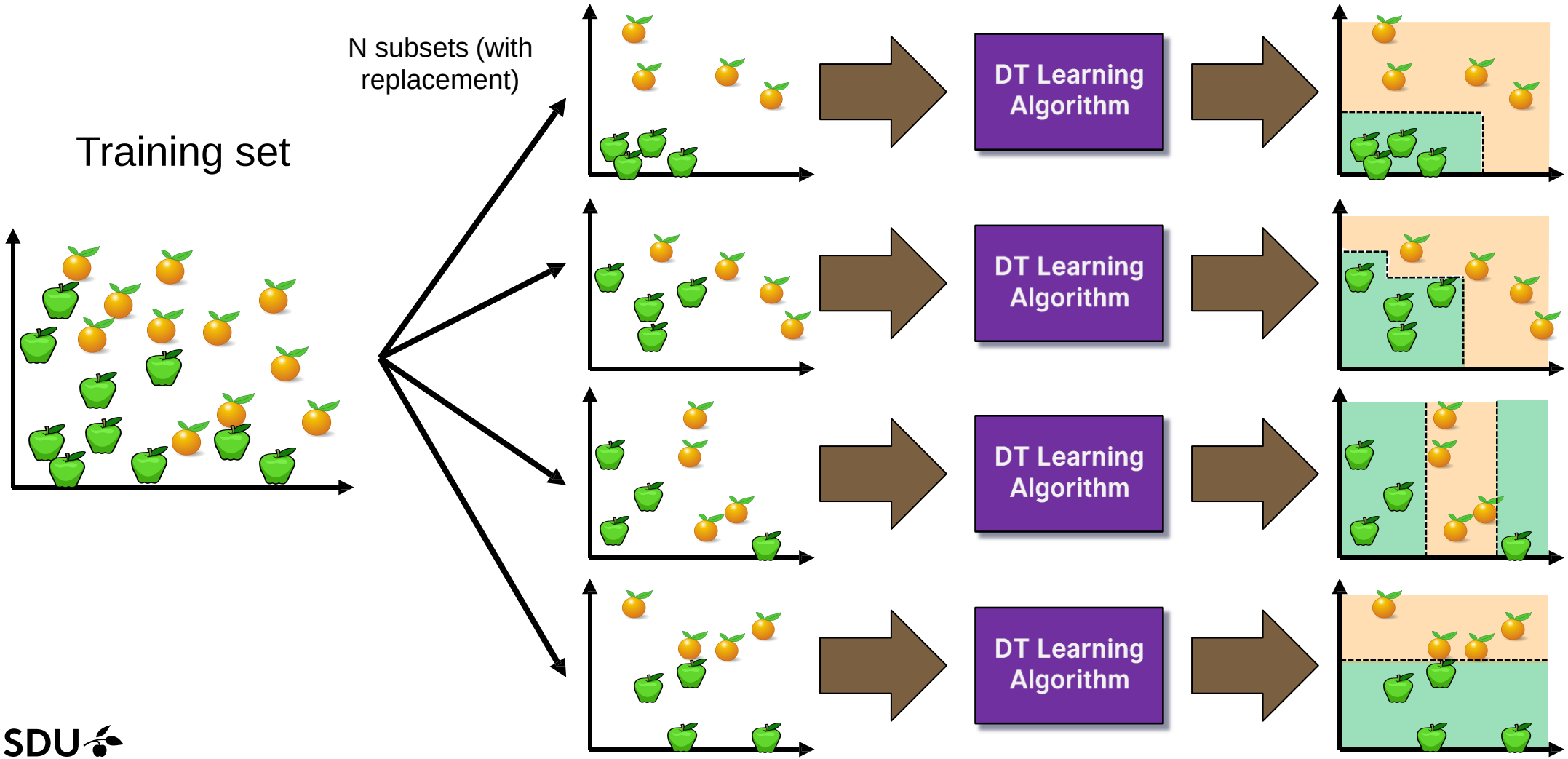
- The slides are based on the slides of Radu Ionescu, Prof. PhD. Faculty of Mathematics and Computer Science, University of Bucharest

Random Forest: Decision Trees on Steroids

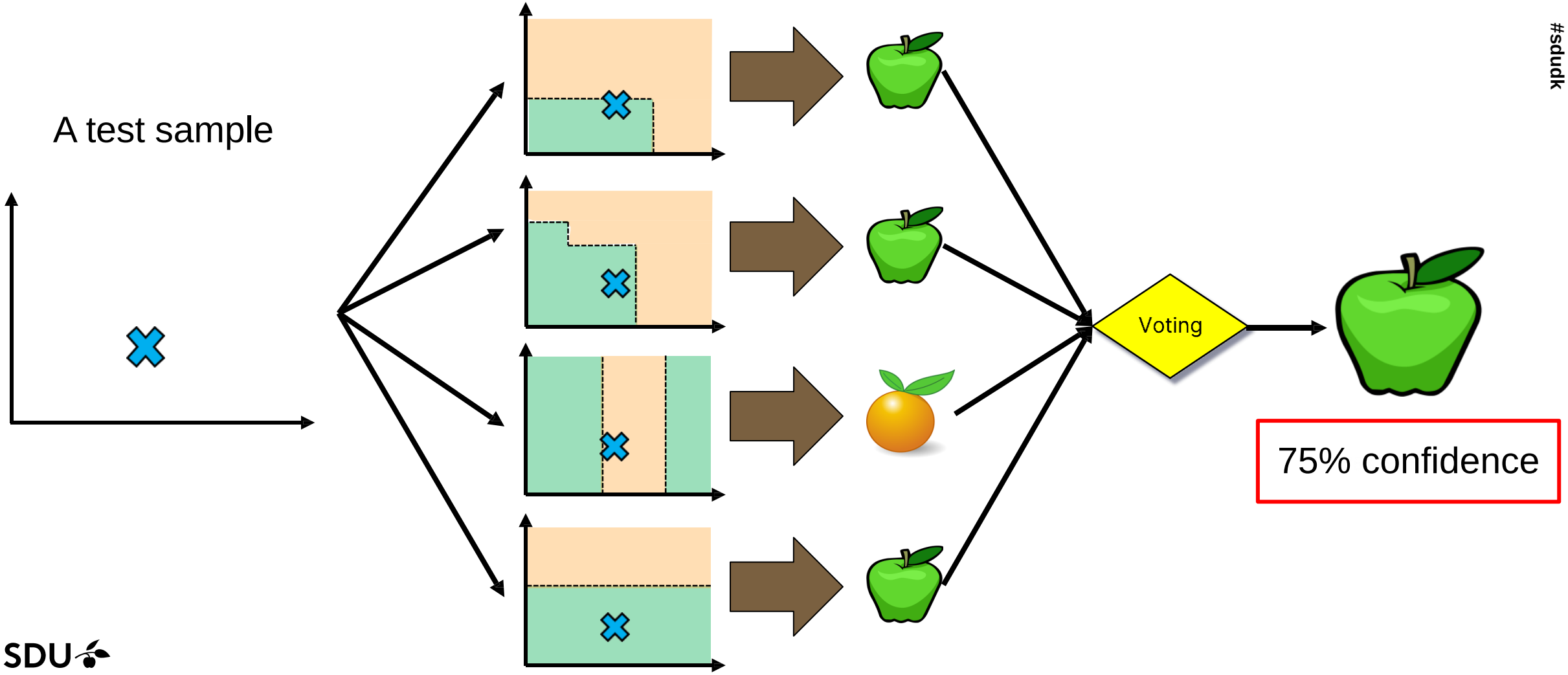
- What we learned about decision trees
  - More complex decision boundary
  - Prone to overfitting
  - Sensitive to the selected training / test split

- Random Forests:
  - Learn multiple trees
  - Diversity is ensured:
    - Random data sampling (Bagging)
    - Random subspaces (different subsets of features)
  - Idea:
    - Ensemble learning
    - Multiple classifiers make errors on different parts of the data
    - The majority gets it right -> better performance

Bagging at training time



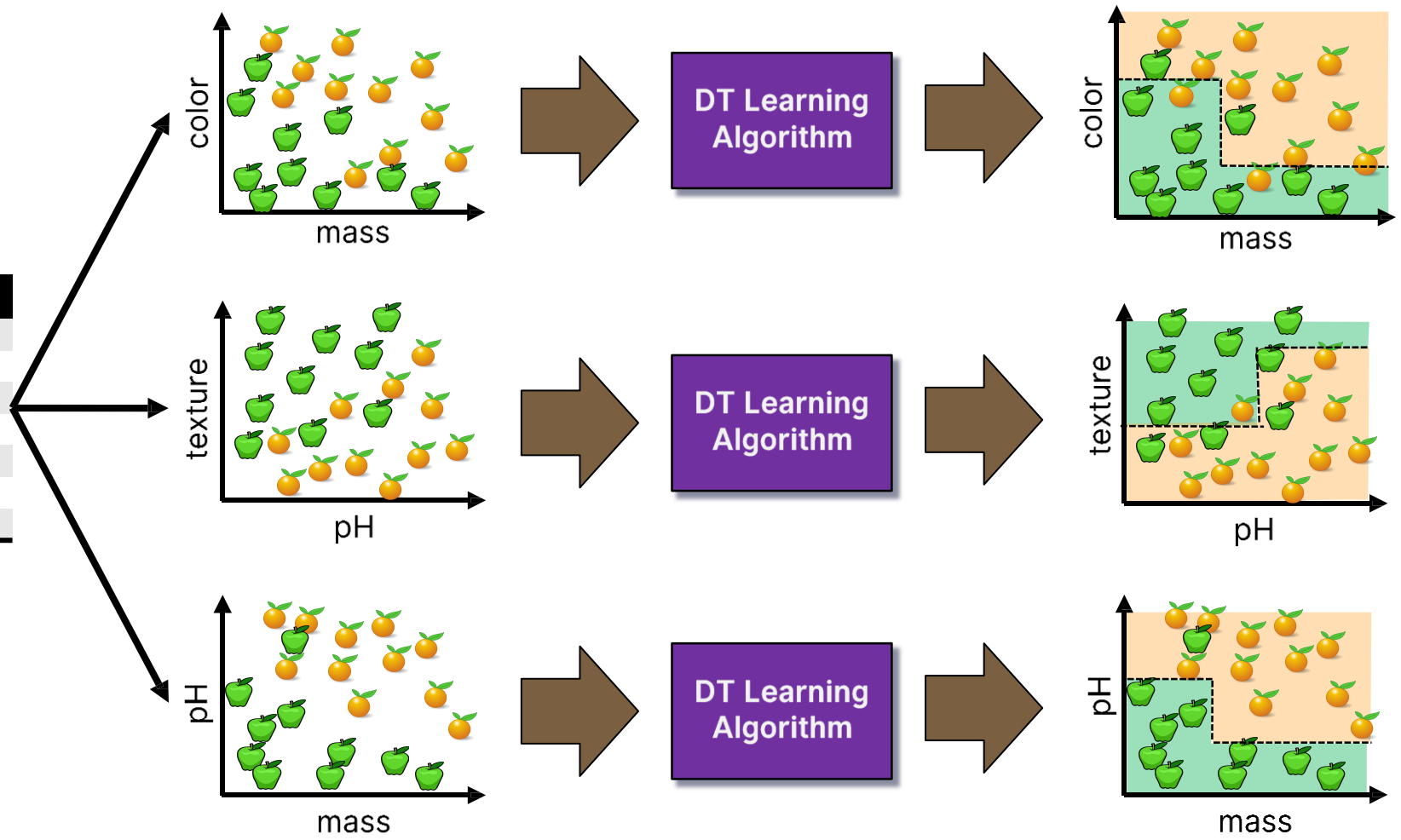
Bagging at inference time



Random Subspace Method at training time

Training data

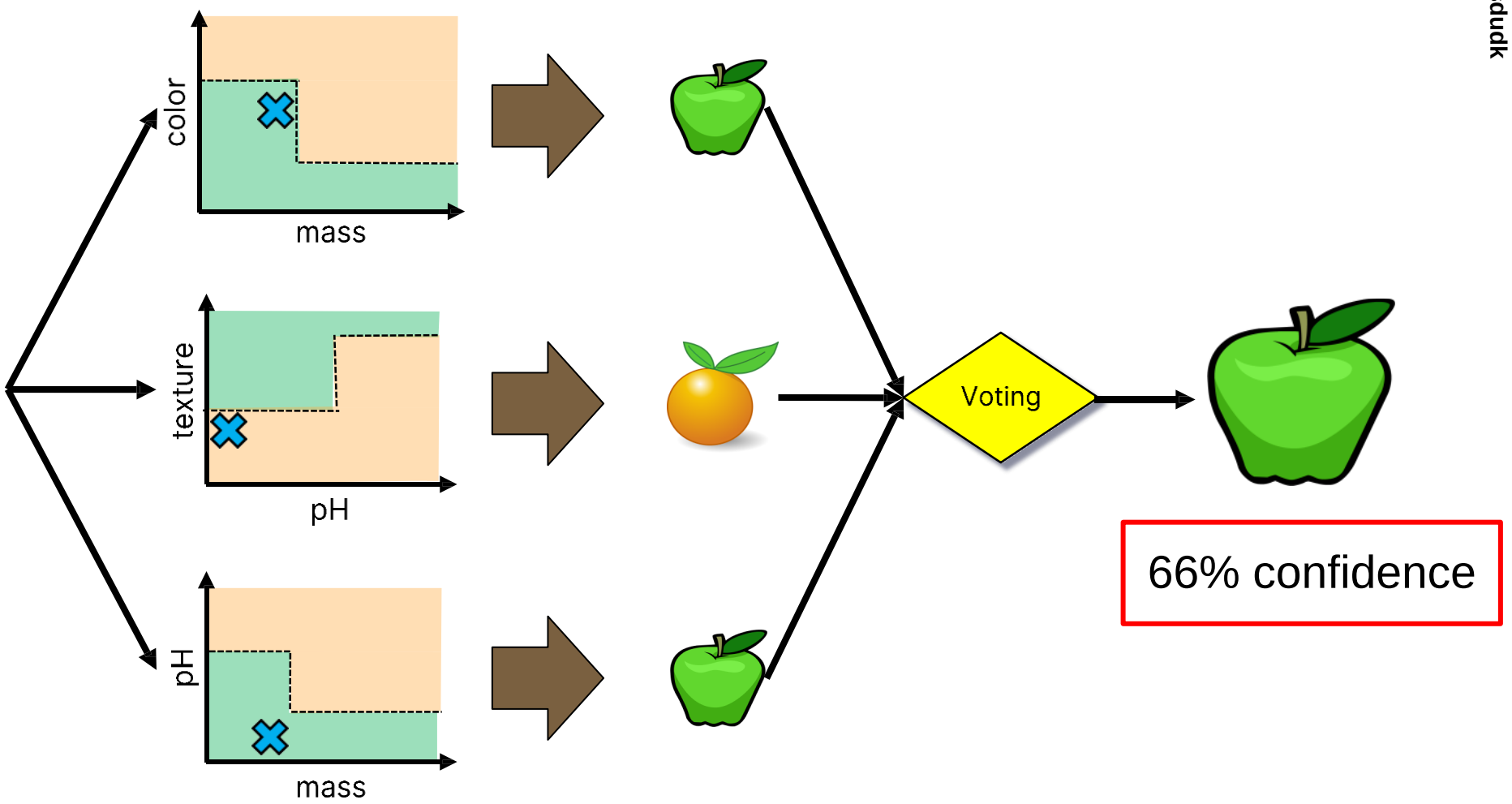
Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



Random Subspace Method at inference time

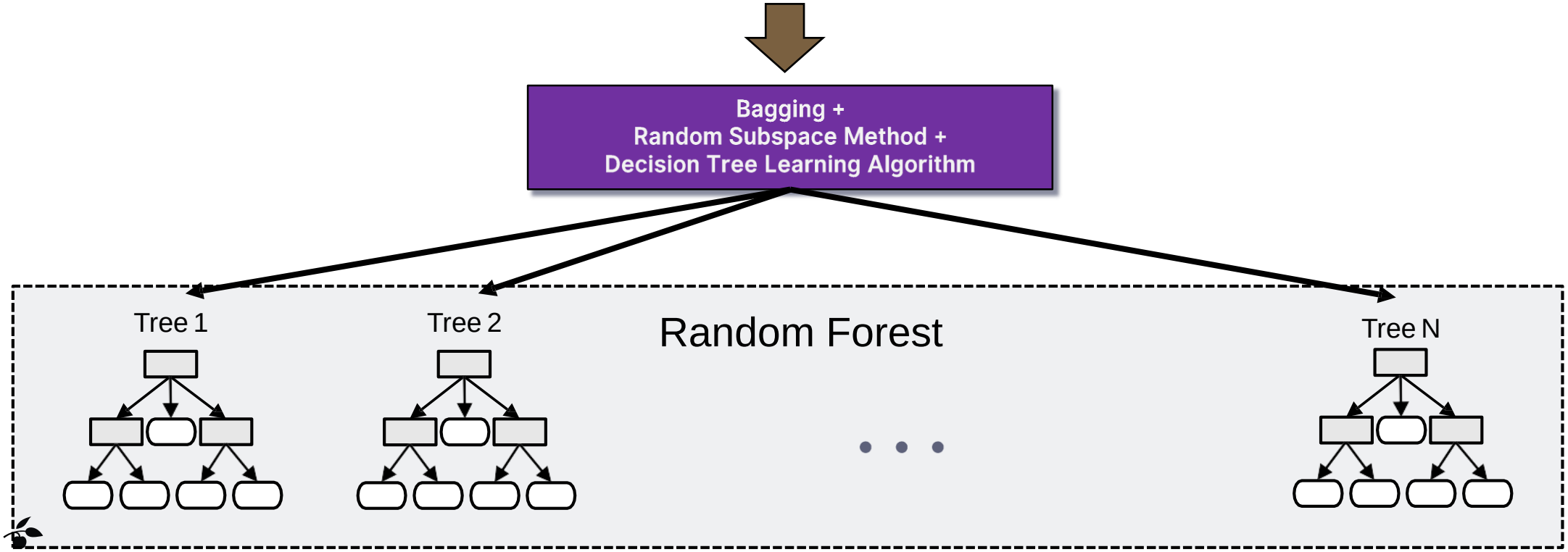
A test sample

87	Red	Smooth	3.1



Random Forests

Mass (g)	Color	Texture	pH	Label
84	Green	Smooth	3.5	**Apple**
121	Orange	Rough	3.9	**Orange**
85	Red	Smooth	3.3	**Apple**
101	Orange	Smooth	3.7	**Orange**
111	Green	Rough	3.5	**Apple**
...				
117	Red	Rough	3.4	**Orange**



Random Forest in R

```
> rf <- randomForest(survived~., data = data_train, ntree=1000)
> print(rf)

Call:
 randomForest(formula = survived ~ ., data = data_train)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 20.62%
Confusion matrix:
         Died Survived class.error
Died      446      56   0.1115538
Survived  116     216   0.3493976
```

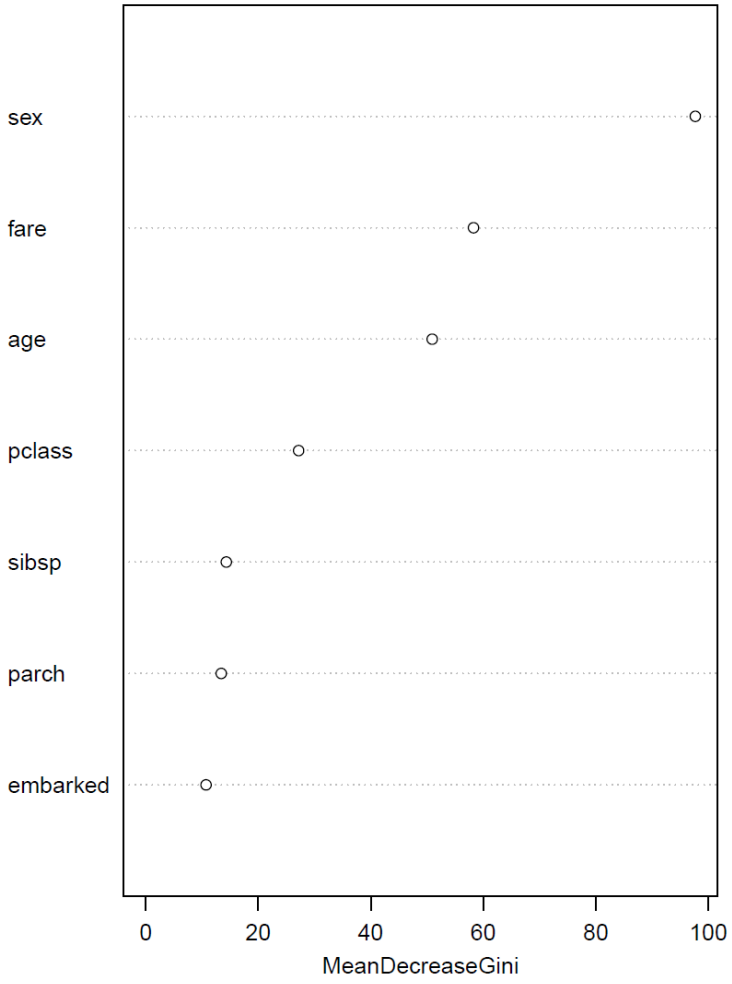
- Again, RF behaves like any other classification algorithm in R
- Interesting Aspects:
  - OOB estimate: That is the error on the samples left out in the bootstrap sample and are used for testing -> basically a built in cross-validation
  - The complexity is mainly controlled with the number of trees to grow (`ntree`)
- Performance: RandomForest gets to an accuracy of 82% on the dataset

Variable Importance

```
> rf <- randomForest(survived~., data = data_train, ntree=1000, importance=TRUE)
> importance(rf)
             Died    Survived MeanDecreaseAccuracy MeanDecreaseGini
pclass   22.96586  33.2179552             40.43434         27.19060
sex      92.86630 106.7077743            119.07691         97.69581
age      25.30986  12.0543726             29.35003         50.93103
sibsp    23.81408  -1.1984172             22.07722         14.33351
parch    17.17242   4.0625309             18.19980         13.43554
fare     18.75535  22.3907098             31.89824         58.27091
embarked 16.34189  -0.9305677             13.48261         10.73340
> varImpPlot(rf)
```

- `importance()` generates a table with the variable importance statistic
  - If you call `randomForest()` with additional parameter '`importance=TRUE`', then more statistics are computed
- The functions `varImpPlot()` plots the importance (next slide)

Variable Importance



Summary

- Ensemble Learning methods combine multiple learning algorithms to obtain performance improvements over its components
- Random Forests utilize:
  - Bagging (resampling of the data with replacement)
  - Random Subspace Method (randomly select features available for the tree)
- They rectify the overfitting problem of decision trees!

- Variable Importance
  - The interpretation is no longer that straight forward
  - We can look at the gain each variable gave for each tree
  - Allows to estimate the overall performance of the different variables