

# Decision Trees, Random Forests, Cross Validation

ECE/CS 498 DS U/G

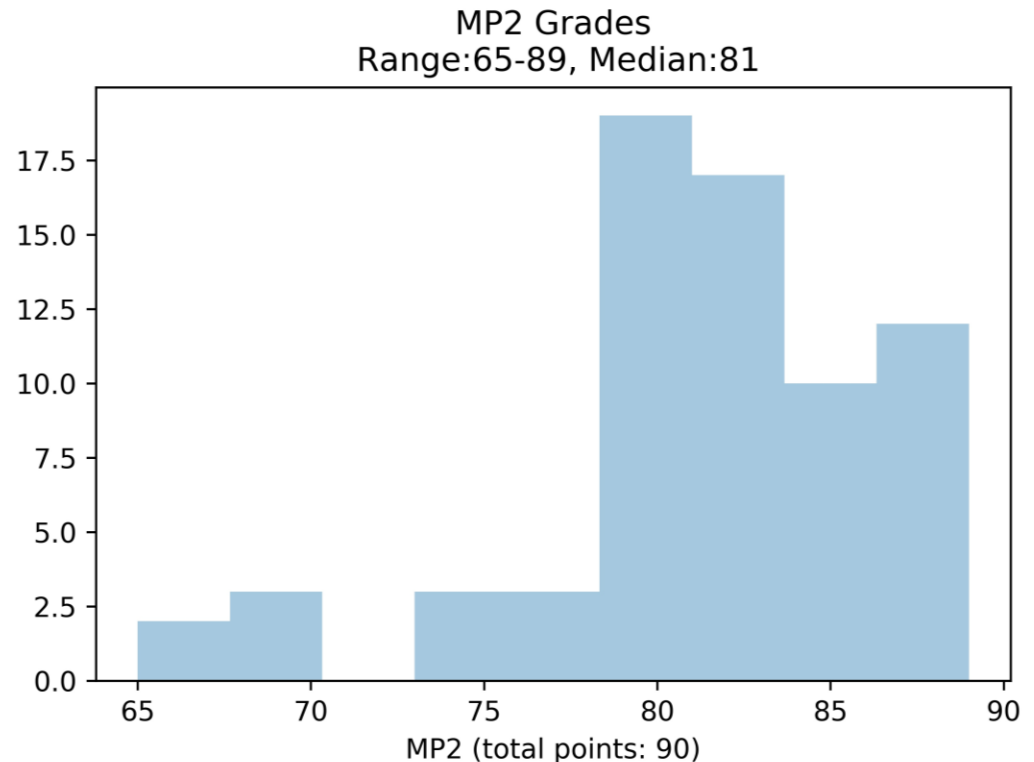
Lecture 22

Ravi K. Iyer

Dept. of Electrical and Computer Engineering  
University of Illinois at Urbana Champaign

# Announcements

- HW4 on HMM and FG due on Wednesday, April 24
- ICA 6 on Wednesday, April 24
- **Final Exam details:**
  - **Date: May 8, 2019**
  - **Time: 1900-2200 hrs**
  - **Venue: Siebel 1404**
- MP2 grades will be released today

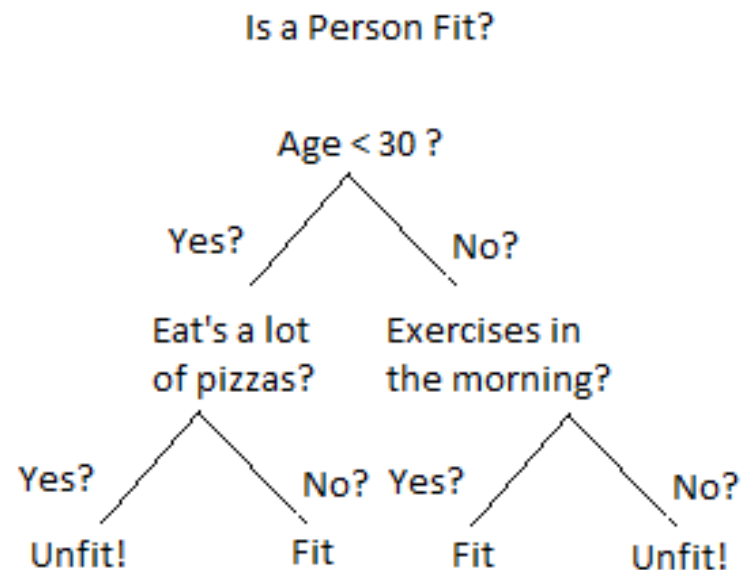


# MP2 General Feedback

- What does a p-value of 0.05 represent in our context? Common mistakes:
  - 5% chance two groups are not different
  - when null hypothesis is true, 5% chance getting this statistics<sup>2</sup>.
- Identifying corresponding clusters
  - Cannot direct compare 2d cluster mean: the 2d representation of baseline and metformin cells used 2 different projection matrix, thus they are not in the same coordinate systems
  - Cannot use baseline prediction of cell  $i$  to make a decision for metformin treated cell  $i$  - they are different cells! (384 cells in total, this was made very clear in the document)
- Many people forgot to provide reasoning for choosing the number of clusters

# Decision Trees: Motivating Example

- Intuitive way of making a decision based on series of simple rules
- Decision trees are a formal way of representing this intuition
- Example: Based on the above model, is a person aged 40 who does not exercise in the morning fit?
  - Unfit



Decision Tree

Image Source: <https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html>

# Decision Trees

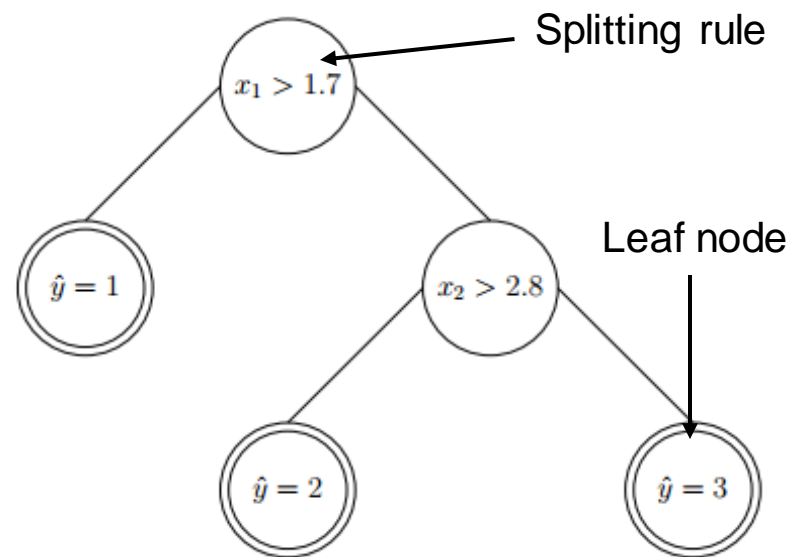
- Supervised learning method
- A decision tree maps input  $x \in R^2$  to output  $y$  using binary decision rules:
  - Each node in the tree has a **splitting rule**
  - Each leaf node is associated with an output value (outputs can repeat)

- Each splitting rule is of the form

$$h(x) = \mathbb{I}\{x_j > t\}$$

for some dimension  $j$  of  $x$  and  $t \in R$

- Using these transition rules, a path to a leaf node gives the prediction

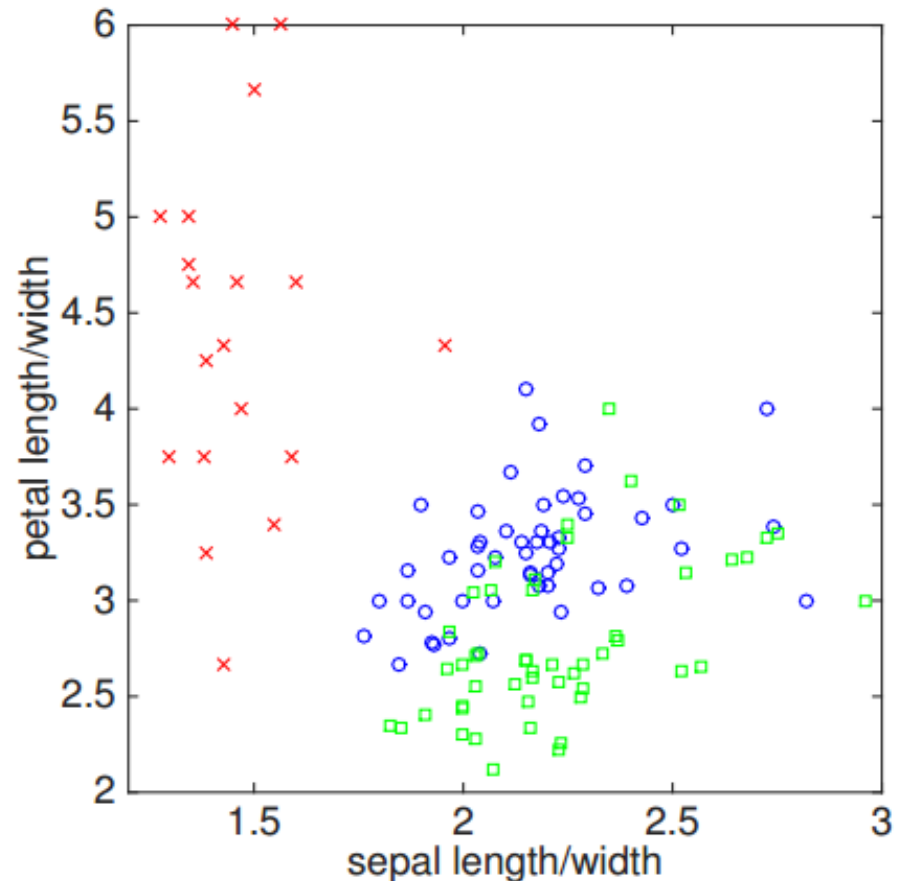


Example decision tree with  $x \in R^2$  and  $y \in \{1,2,3\}$ .

# Decision Tree Example

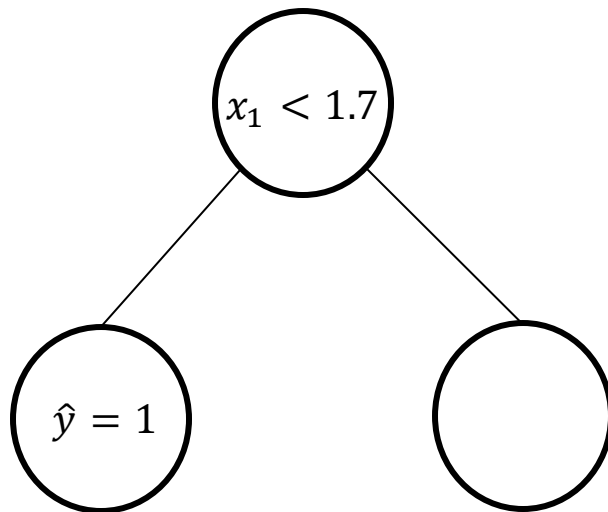
Classifying iris flowers (virginica, setosa, versicolor) using petal and sepal measurements

- $x \in R^2, y \in \{1,2,3\}$
- $x_1$  = ratio of sepal length to width
- $x_2$  = ratio of petal length to width

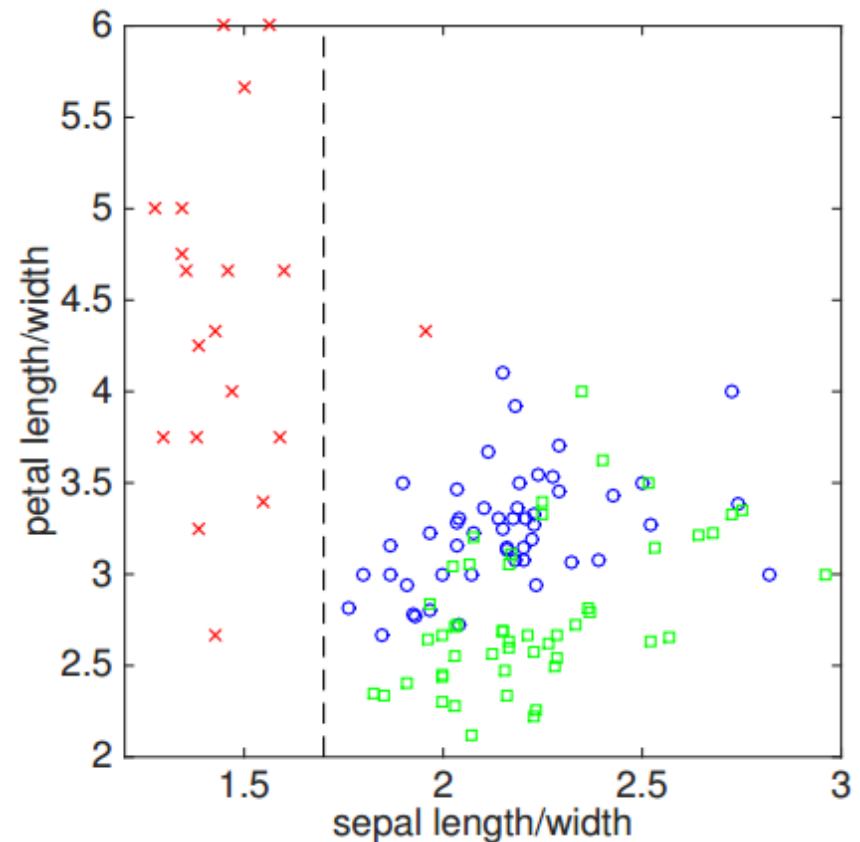


# Decision Tree Example

- The splitting function  $x_1 < 1.7$  separates the red class of flowers from the blue and green classes

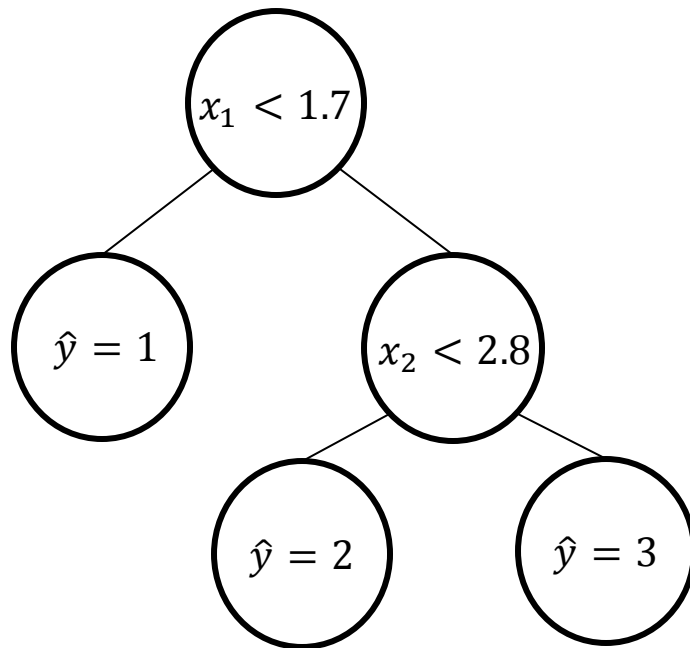


Decision tree

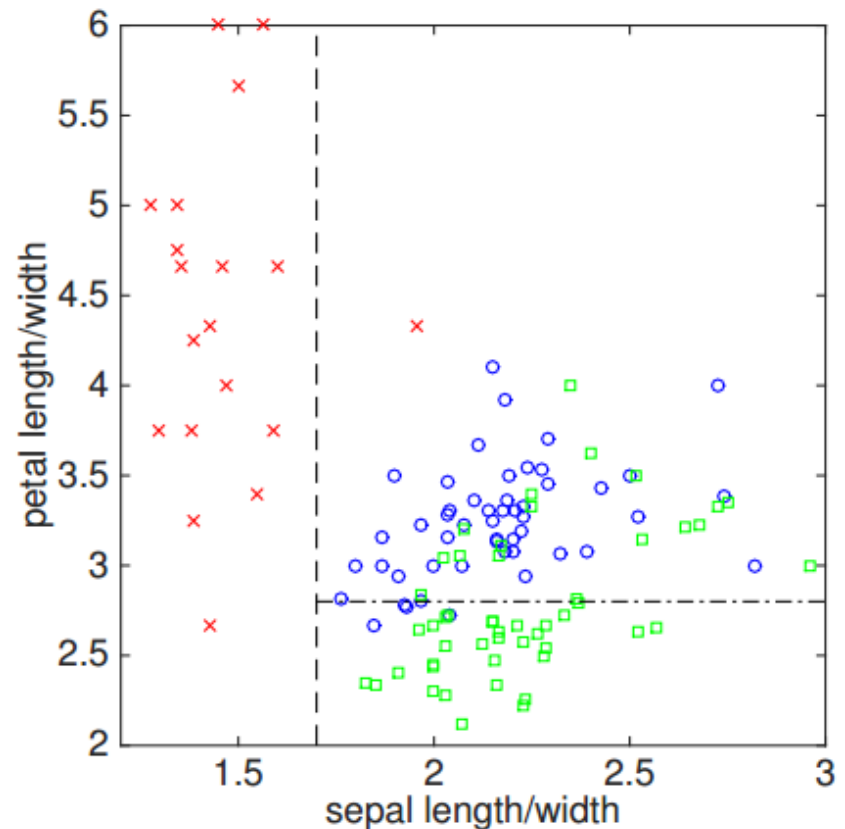


# Decision Tree Example

- The splitting function  $x_2 < 2.8$  separates the green class of flowers from the blue class



Decision tree





# Decision Tree: King Example

The king of TreeLand loved fooling locals by disguising himself on the streets. This was annoying because whenever someone unintentionally treated the king badly, he would be punished. People were quite vexed, and wise man John stepped up to solve the problem.

John assumed features of the king that distinguish him in public.

- Comes out of the castle (Castle)
- Walks slowly (Slow)
- Eat 5 or more meals daily (Greedy)
- Has a golden tooth (Gold\_Tooth)

# King Example Data Set

- John collected the following dataset over the next few days which includes data of the king as well as other people

Castle	Slow	Greedy	Gold_Tooth	Is_King
Y	Y	N	N	N
N	N	Y	Y	Y
N	Y	N	Y	Y
Y	N	Y	Y	Y
Y	Y	N	Y	Y
N	N	Y	Y	N
N	Y	N	Y	N
Y	N	Y	N	N
N	Y	N	Y	N
N	N	Y	Y	N

- Can a decision tree be used for the classification of Is\_King using the features Castle, Slow, Greedy and Gold\_Tooth?

# How to decide the splitting function for the decision tree?

Castle	Is_King
Y	N
N	Y
N	Y
Y	Y
Y	Y
N	N
N	N
Y	N
N	N
N	N

Slow	Is_King
Y	N
N	Y
Y	Y
N	Y
Y	Y
N	N
Y	N
N	N
Y	N
N	N

Greedy	Is_King
N	N
Y	Y
N	Y
Y	Y
N	Y
Y	N
N	N
Y	N
N	N
Y	N

Gold_Tooth	Is_King
N	N
Y	Y
Y	Y
Y	Y
Y	Y
Y	N
Y	N
N	N
Y	N
Y	N

- Use the features that are most predictive of the outcome for splitting rule
- Gold\_Tooth and Castle are more predictive of Is\_King compared to Greedy and Slow
  - Can use them for splitting rule
- Intuition of “more predictive” captured in Information Gain (IG) criteria

$$IG(Y, F) = H(Y) - H(Y|F)$$



Entropy of Y

Green: If the attribute matched the corresponding label  
 Red: If the attribute did not match the corresponding label

# Information Gain

$$H(Y) = - \sum_{y \in Y} p_y \log_2(p_y)$$

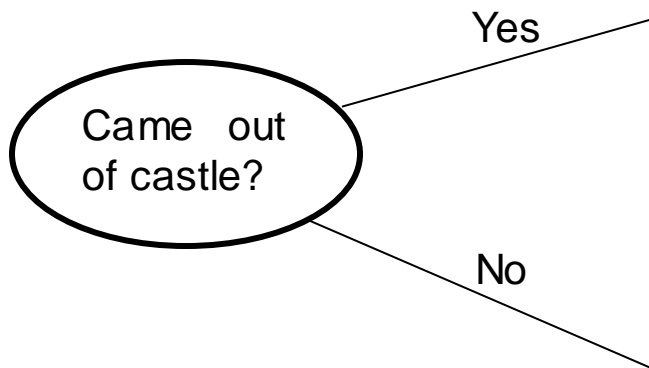
- Entropy measures uncertainty
  - Example:  $Y = \{0,1\}$ ,  $p_0 = 0.5$ ,  $H(Y) = 1$
  - Example:  $Y = \{0,1\}$ ,  $p_0 = 0$ ,  $H(Y) = 0$
- Information gain measures the difference in predictability before and after providing information about the feature **F**

<b>F</b>	<b>H(Y)</b>	<b>H(Y F)</b>	<b>IG(Y;F)=H(Y)-H(Y F)</b>
“Very” Predictive	High	Low	High
Not “Very” Predictive	High	High	Low

Qualitative effect of F on IG

# Using Information Gain to pick splitting rule

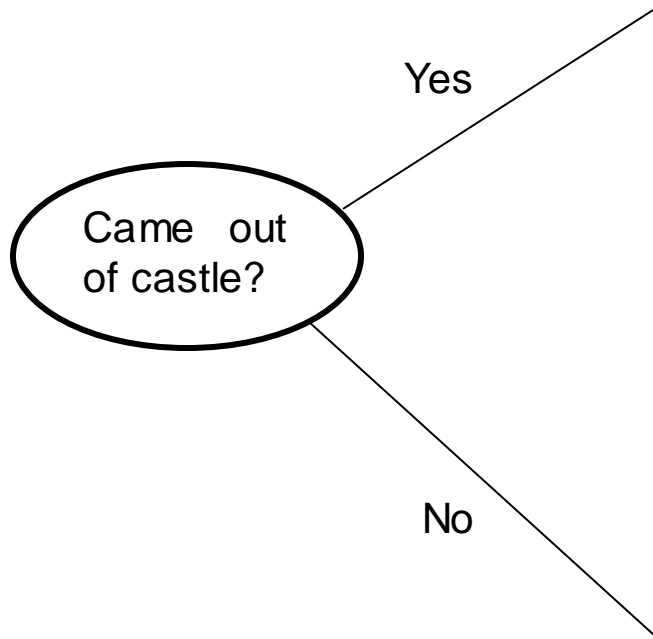
- Splitting rule: Pick the feature with highest information gain
- Information Gain for Castle and Gold\_Tooth is equal and is greater than IG of Slow and Greedy
- John picks Castle as the first feature which gives the following decision tree and data split



Castle	Slow	Greedy	Gold_Tooth	Is_King
Y	Y	N	N	N
Y	N	Y	Y	Y
Y	Y	N	Y	Y
Y	N	Y	N	N

Castle	Slow	Greedy	Gold_Tooth	Is_King
N	N	Y	Y	Y
N	Y	N	Y	Y
N	N	Y	Y	N
N	Y	N	Y	N
N	Y	N	Y	N
N	N	Y	Y	N

# Continue splitting...



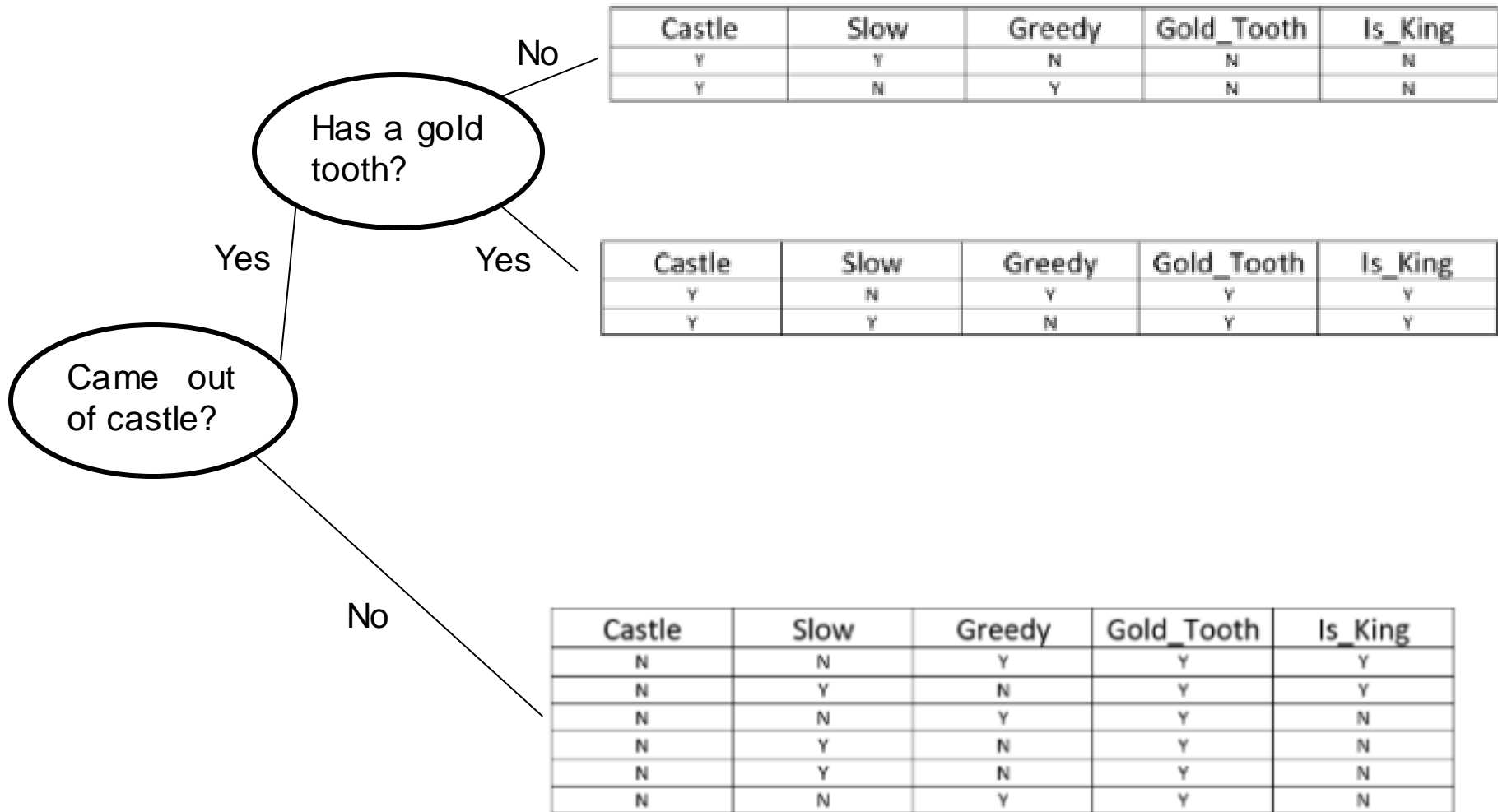
Castle	Slow	Greedy	Gold_Tooth	Is_King
Y	Y	N	N	N
Y	N	Y	Y	Y
Y	Y	N	Y	Y
Y	N	Y	N	N

- 2/4 cases can be classified correctly (50%)
- This node can be split further
- IG is maximum for Gold\_Tooth for this subset of data

Castle	Slow	Greedy	Gold_Tooth	Is_King
N	N	Y	Y	Y
N	Y	N	Y	Y
N	N	Y	Y	N
N	Y	N	Y	N
N	Y	N	Y	N
N	N	Y	Y	N

- 4/6 cases can be classified correctly, so this node is not split further

# King Example: Final decision tree



# Decision Trees Algorithm

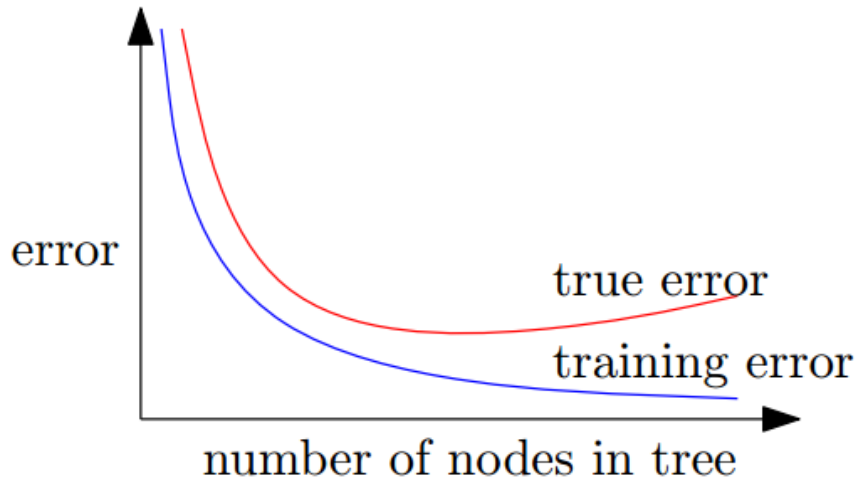
The basic method for learning trees is with a top-down greedy algorithm

1. Start with a single leaf node containing all data
2. Loop through the following steps:
  - Pick the leaf to split that reduces uncertainty the most
  - Pick the splitting rule for the chosen leaf node using the following
    - i. For each predictor, all possible splits of the predictor values are considered
    - ii. For each predictor, the best split is considered. Criteria for best could be classification error, information gain, Gini Index etc
    - iii. With the best split of each predictor determined, pick the best predictor in that group
3. Continue splitting nodes (increasing depth of tree) until stopping criteria is reached (not discussed here)

Label/response of the leaf is majority of the data assigned to it



# Bootstrap Aggregation



- Challenge: Decision Trees are prone to overfitting
  - Training error decreases as nodes increase
  - Testing error decreases and then increases due to overfitting

- Solution: Use multiple decision trees, each trained with a subset of the training data
- Termed as “Bagging”: **B**ootstrap **a**ggregation

Sample dataset with replacement. Take up to the number of observations in the original data.

Aggregate decision from multiple trees

# Bootstrap Aggregation

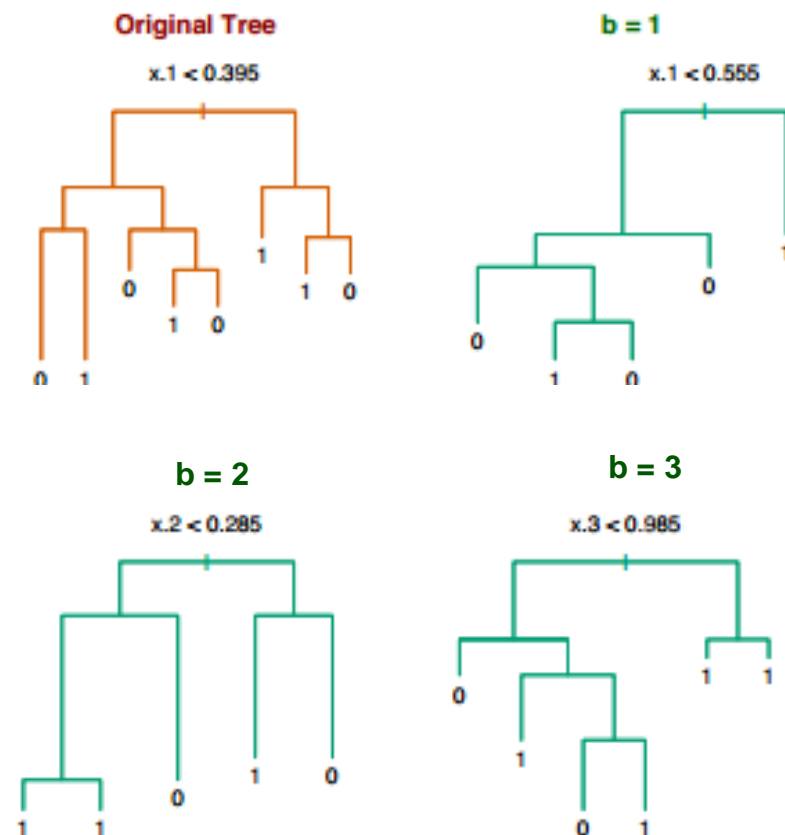
Consider a training dataset of size  $n$

Algorithm:

For  $b = 1, \dots, B$ :

1. Draw a bootstrap sample  $\mathcal{B}_b$  of the same size  $n$  from the training data (sample data with replacement)
2. Train a decision tree  $f_b$  with  $\mathcal{B}_b$

For a new point  $x_0$ , compute the decision from all  $f_b$ 's and pick the majority decision



# Random Forests

- Challenge: Trees obtained by bagging are correlated which decreases its benefits
- Solution – at each split, only consider a random subset of dimensions to choose the splitting rule; Random Forests

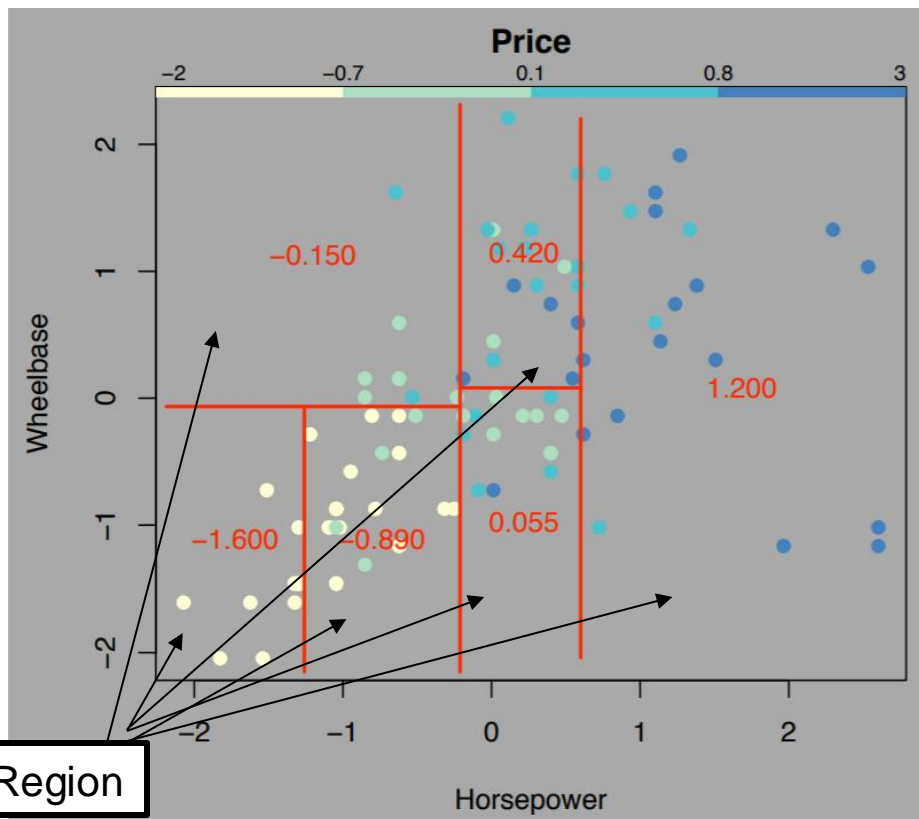
Consider training data of size  $n$  for data of dimension  $d$ , and a positive integer  $m < d$  (often  $m \approx \sqrt{d}$ )

Algorithm:

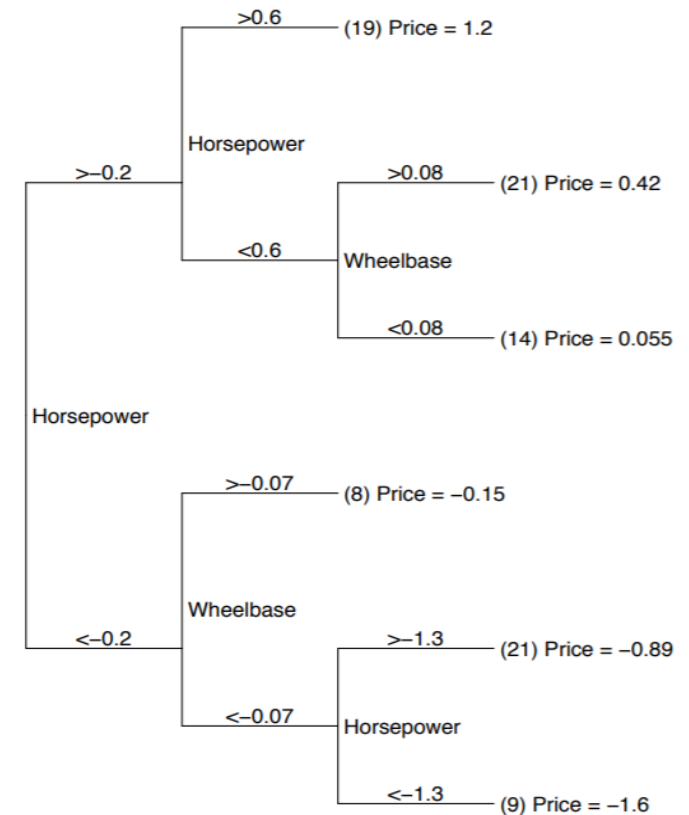
1. Draw a bootstrap sample (sample data with replacement)  $\mathcal{B}_b$  of the same size  $n$  from the training data
2. Train a tree  $f_b$  on  $\mathcal{B}_b$ , where each split is computed as follows:
  - i. Randomly select  $m$  dimensions of  $x \in R^d$ , newly chosen for each  $b$
  - ii. Make the best split restricted to the subset of dimensions

# Decision Trees for Regression

- Example: Predict the price of a car given its Horsepower and Wheelbase
- For a new point in a given region, the tree assigns value equal to the average of the samples within that region



Partition of feature space due to decision tree



Decision Tree for regression

# Advantages of Random Forests

- Applicable to both regression and classification problems
- Handle categorical predictors naturally
- Computationally simple and quick to fit, even for large problems
- No formal distribution assumptions (non-parametric)
- Can handle highly non-linear interactions and classification boundaries
- Provides variable/feature importance
- Handles missing values
- Offers some interpretability; though not as interpretable as Decision Trees

# Summary of similarities and differences

	<b>Decision Tree</b>	<b>Bootstrap Aggregation</b>	<b>Random Forests</b>
<b>Number of trees</b>	Single	Multiple	Multiple
<b>Dataset</b>	Entire data	Random subset of data for each tree	Random subset of data for each tree
<b>Dimensions used for picking splitting rule</b>	All dimensions considered at each node	All dimensions considered at each node	Random subset of dimensions considered at each node
<b>Output for classification task</b>	Decision of tree	Majority decision from all the trees	Majority decision from all the trees

# Validation:

## Binary Classification Performance

You are solving a Binary Classification Problem, Predict if person has cancer (1) or not (0). You have learnt a model using your training dataset.

**How do you test your model?**

Explanation:

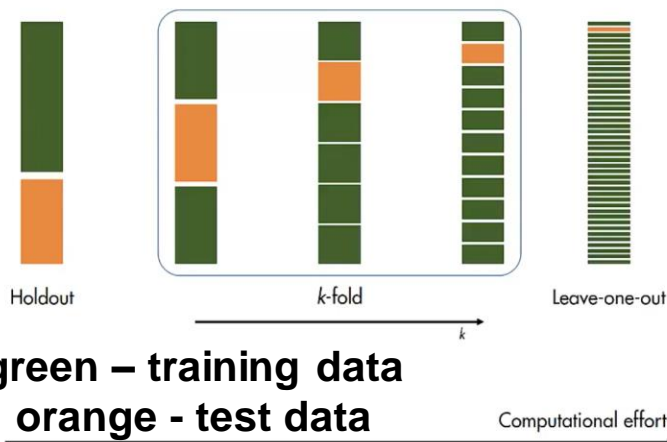
- 1) **Holdout**: Divide dataset into 2 parts, learn model on training data and validate on the test data
- 2) **K-fold**: Divide data into K equal parts, run the following K-times: (hold out one part of the dataset, train on the remaining, test on the holdout). Every data-sample is part of the training data K-1 times and part of the test data 1 time.
- 3) **Leave-one-out**: For every data sample, leave it out, train the model on the remaining data and predict for the left out data sample.

**Cross Validation:** For a given training iteration, divide the dataset into 2 parts:

- training set (for training the model)
- test set (for validating the model)

Different ways of doing this:

- (1) Holdout
- (2) K-fold
- (3) Leave-one-out

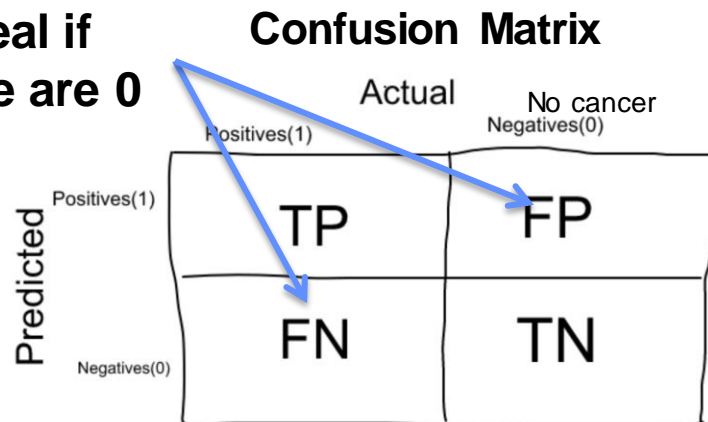


# What does it mean to **validate** on the test data?

Ideal if  
these are 0

**Confusion Matrix**

		Actual Positives(1)	No cancer Negatives(0)
Predicted Positives(1)		TP	FP
Negatives(0)		FN	TN



Depending on the **cost of making a mistake**, minimize the number of FN or FP

## Common issues

**Precision** = 1 if FP=0, but FN may be high

**Sensitivity** = 1 if predict everything as positive

**Specificity** = 1 if predict everything as negative

How to strike a **balance**? Use F1-score!

## Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

(Recall)

$$\text{Specificity} = \frac{TN}{TN + FP}$$

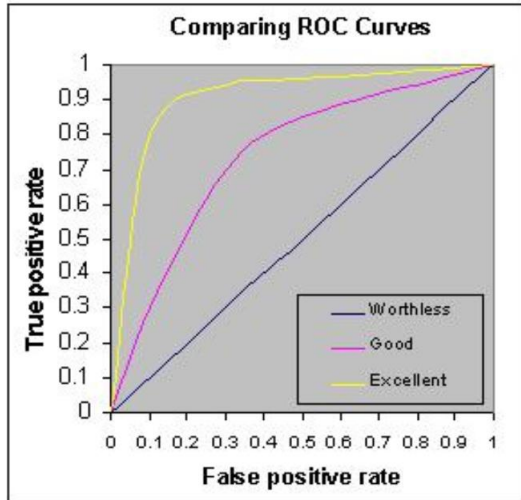
$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

**Harmonic mean** of Precision and Recall  
(when large disparity between the 2 values,  
the score is closer to the smaller value)



# ROC and AUC

Receiver Operating Characteristic curve: A plot of **False Positive Rate** Vs **True Positive Rate**



$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}$$

## Area Under Curve (AUC):

A good classifier would have the largest area under the ROC curve.

The random predictor would have an ROC of 0.5 (the curve of the 'worthless' model in fig.)

## AUC value interpretation –

If a pair of data samples are drawn independently, one each from the positive and negative sets, AUC gives the probability that the classifier will predict a lower score for the negative sample as compared to the positive sample

An ML model has parameters and for different values of the parameters, the model gives different FPR and TPR.

An **ROC (Receiver Operating Characteristic) curve** can be plotted for these different settings. A **good setting** of the parameters i.e. a good classifier, would have a **high TPR and low FPR**.

# References

- [https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm#giniimp](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#giniimp)
- Adele Cutler lecture slides (Utah State University)
- John Paisley lecture slides (Columbia University)
- Anthony Anh Quoc Doan lecture slides (California State University Long Beach)
- <https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-decision-trees-adb2165ccab7>