Random Forests and Cross Validation

Lecture 26:
Random Forests and Cross Validation

ECE/CS 498 DS

Professor Ravi K. Iyer

Department of Electrical and Computer Engineering

University of Illinois

Announcements

- MP 3 final submission due **Wed 4/29 @ 11:59 PM** via Compass
- HW 5 will be released tonight, due **Mon 5/4 @ 11:59 PM** via Compass
  - Covers SVM, neural networks, and random forest
- Discussion section on **Fri 5/1** will be additional practice with neural networks
- Final Exam will **be Friday 5/15 from 8-11 AM**
  - Please plan to have a webcam available during the exam
  - More details about logistics to come soon

Course Grades

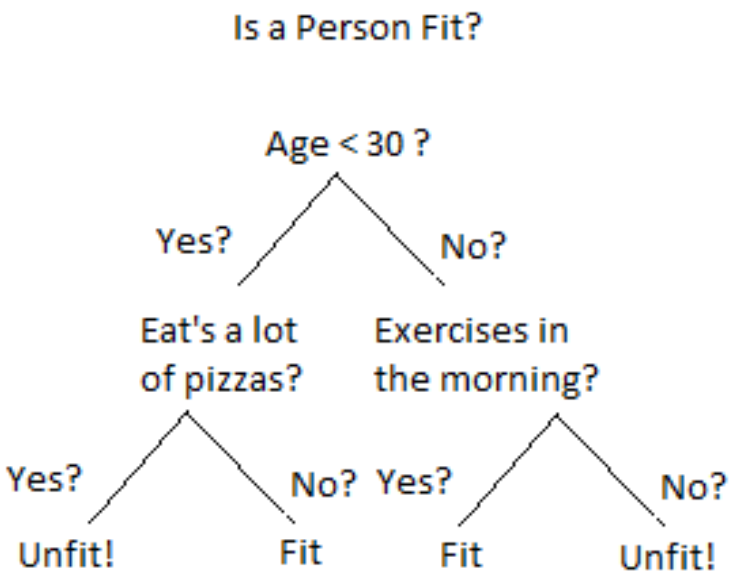
- Most of the course grades have been released already
  - **HW** 0-2 (HW 3 was optional)
  - **ICA** 1-5
  - **Quiz** 1-6
  - **MP 1** Checkpoint and Final Submission
  - **MP 2** Checkpoints 0.5, 1, 1.5
  - **MP 3** Checkpoint 1.5
  - **Final Project** Proposal, Progress Report 1, Progress Report 2
- Current Assignments still being graded:
  - **MP 2** Final Submission
  - **MP 3** Checkpoint 1
  - **HW 4**
  - **ICA 6**
- Histograms will be posted

Decision Trees and Random Forest

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

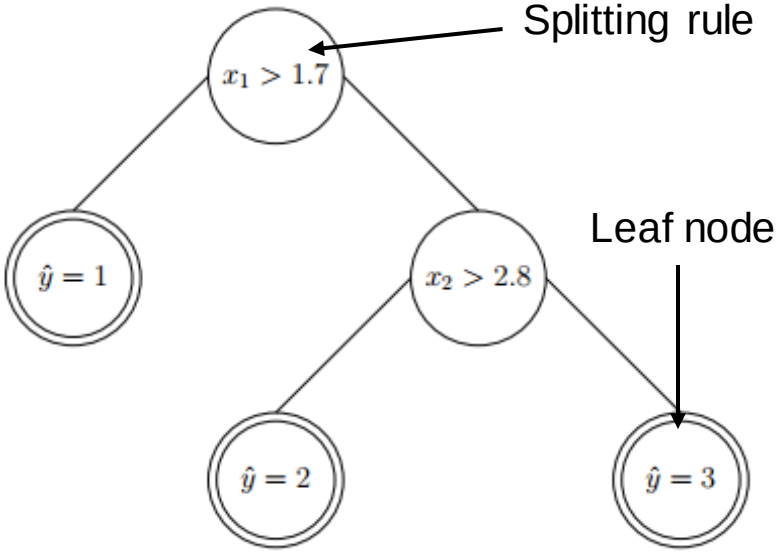
- Supervised learning method
- A decision tree maps input $x \in R^2$ to output y using binary decision rules:
  - Each node is the tree has a **splitting rule**
  - Each leaf node is associated with an output value (outputs can repeat)

- 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

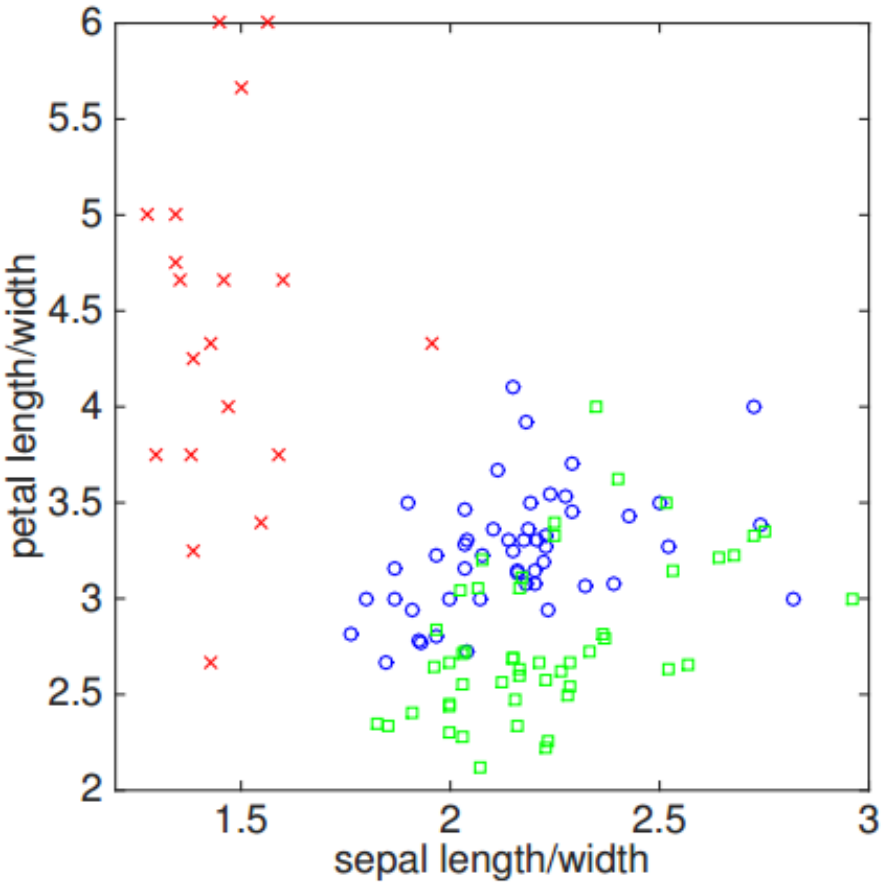


Example decision tree with $x \in R^2$ and $y \in \{1,2,3\}$.

Decision Tree Example

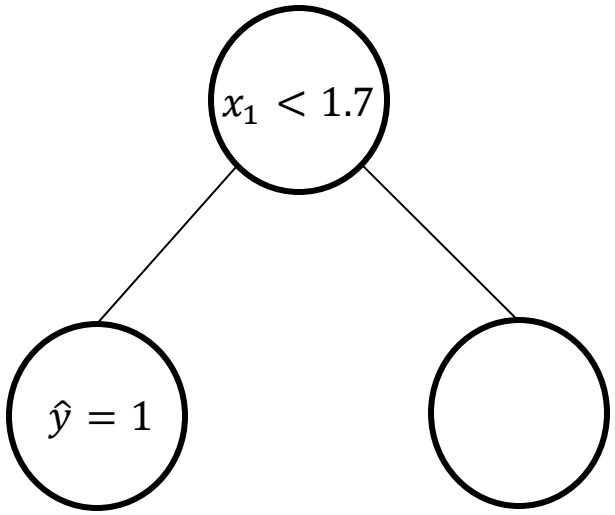
Classifying iris flowers (virginica, setosa, versicolor) using petal and sepal measurements

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

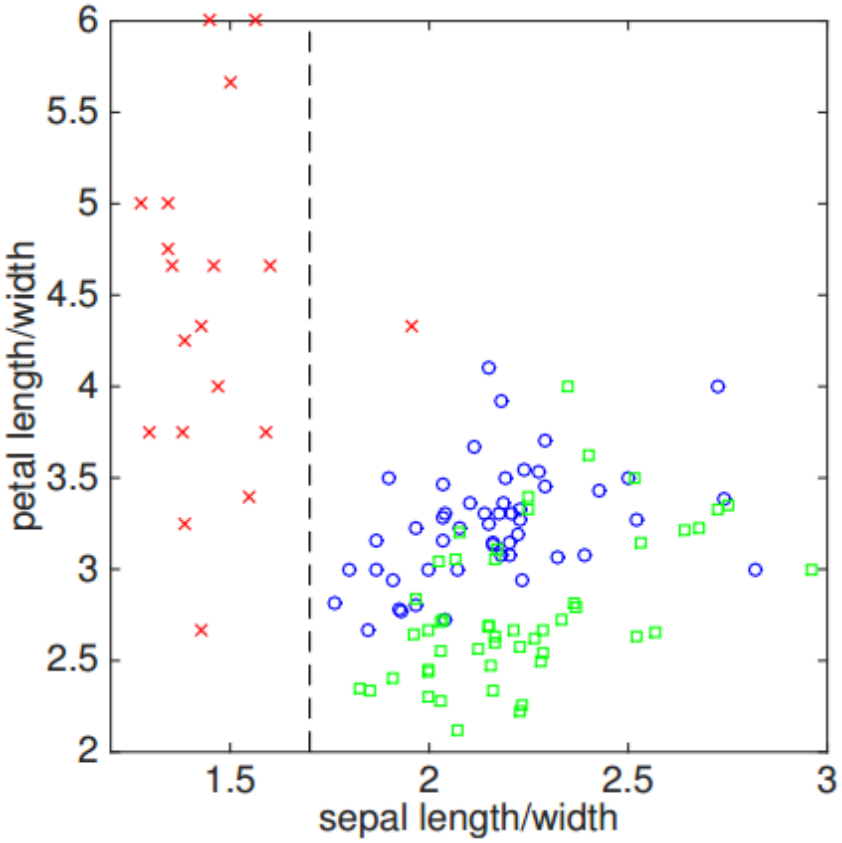


Decision Tree Example

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

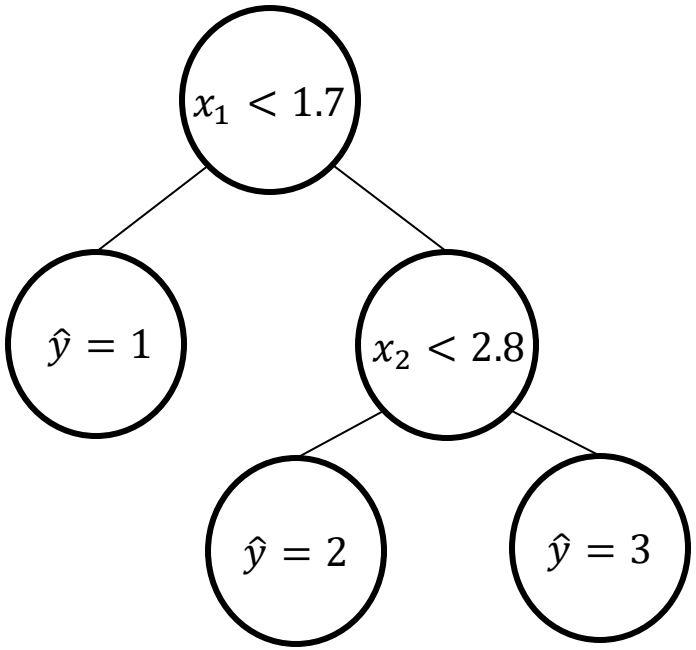


Decision tree

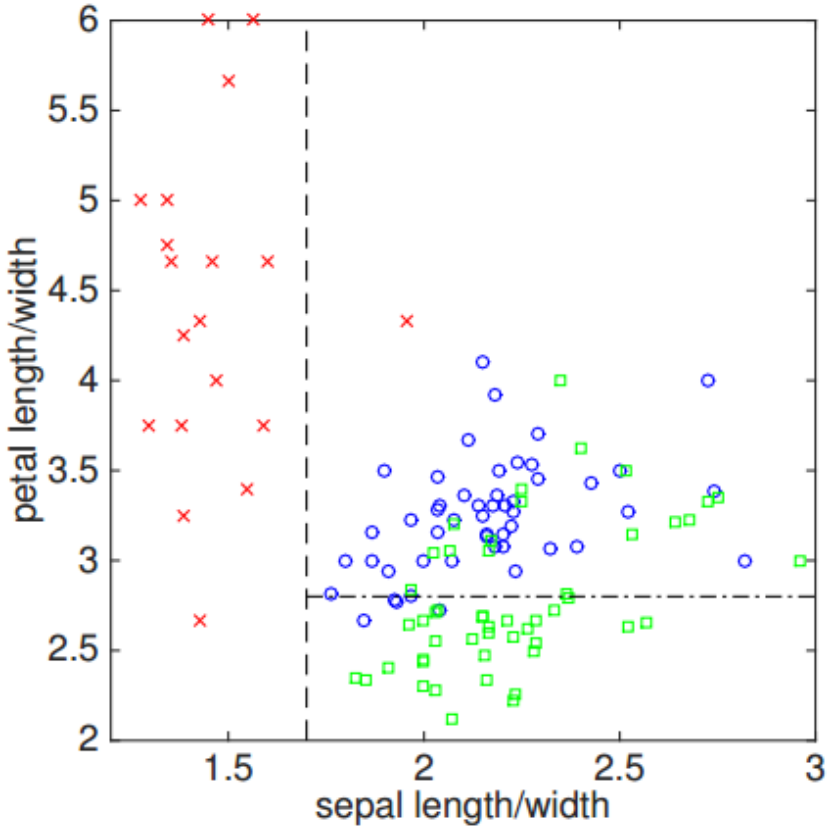


Decision Tree Example

- 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

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

- 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

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: $Y = \{0,1\}, p_0 = 0, H(Y) = 0$

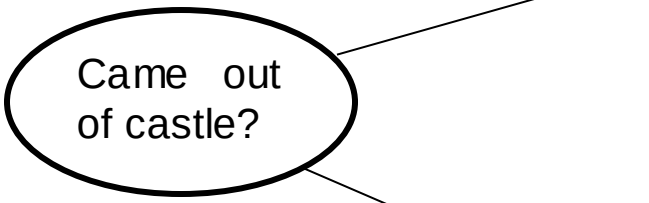
- Information gain measures the difference in predictability before and after providing information about the feature **F**

F	H(Y)	H(Y|F)	IG(Y;F)=H(Y)-H(Y|F)
"Very" Predictive	High	Low	High
Not "Very" Predictive	High	High	Low

Qualitative effect of F on IG

Using Information Gain to pick splitting rule

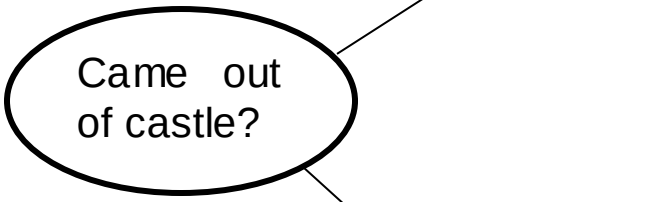
- Splitting rule: Pick the feature with highest information gain
- Information Gain for Castle and Gold_Tooth is equal and is greater than IG of Slow and Greedy
- John picks Castle as the first feature which gives the following decision tree and data split



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…



Yes

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

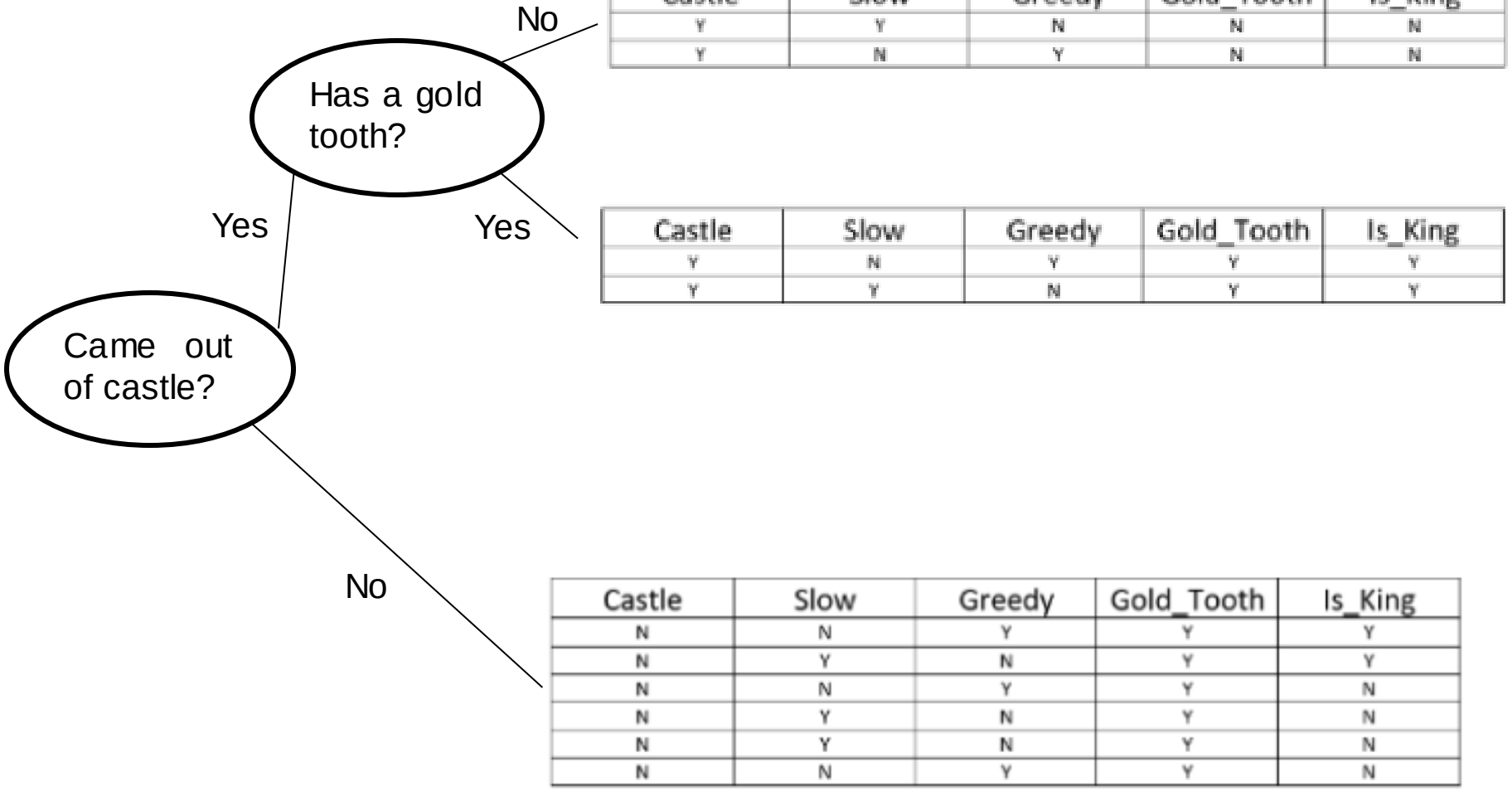
- 2/4 cases can be classified correctly (50%)
- This node can be split further
- IG is maximum for Gold_Tooth for this subset of data

No

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 Exampe: Final decision tree



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

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

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

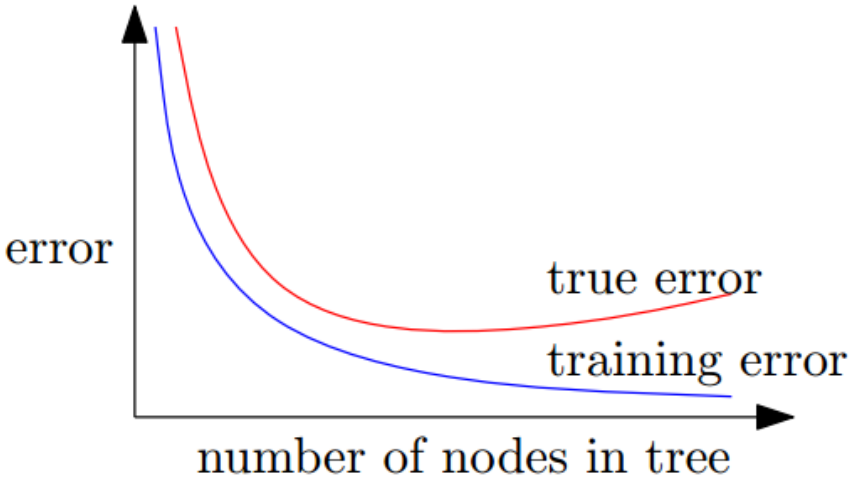
Decision Trees Algorithm

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 **agg**regation

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

Aggregate decision from multiple trees

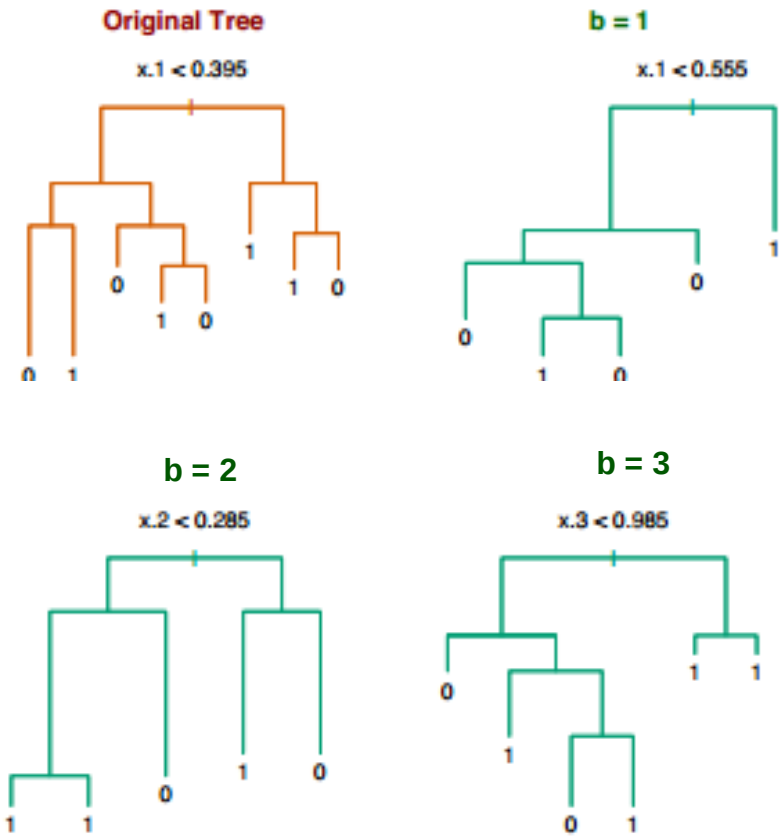
Bootstrap Aggregation

Consider a training dataset of size n

Algorithm:
For $b = 1, \ldots, B$:

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

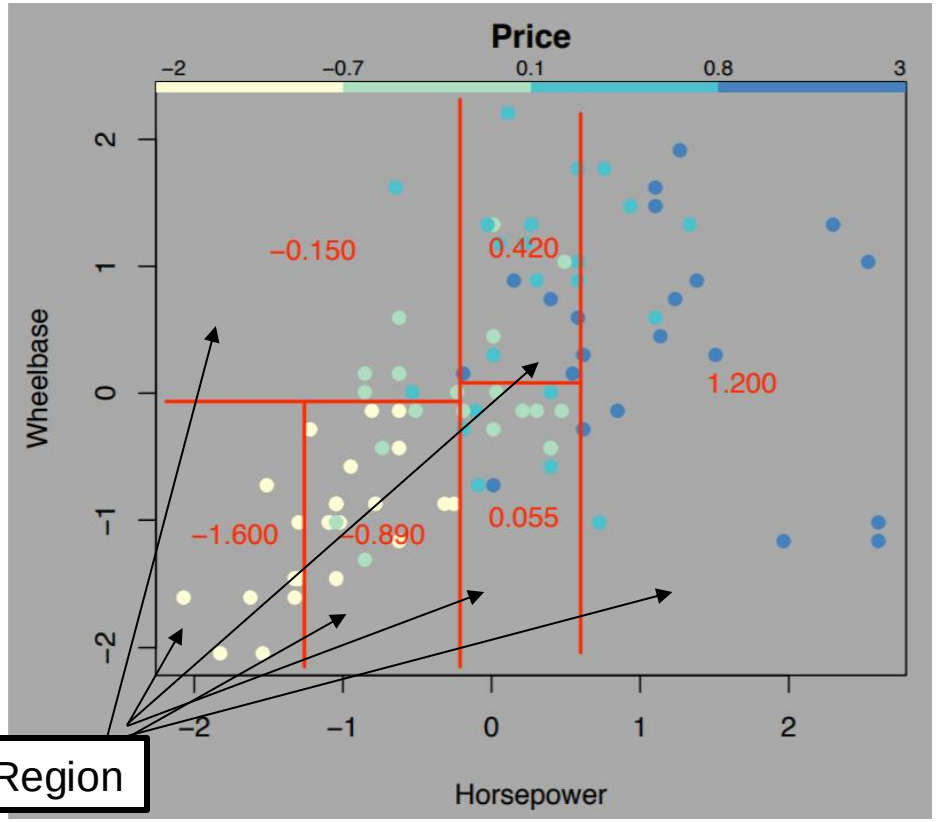
Consider training data of size n for data of dimension d, and a positive integer $m < d$ (often $m \approx \sqrt{d}$)

Algorithm:

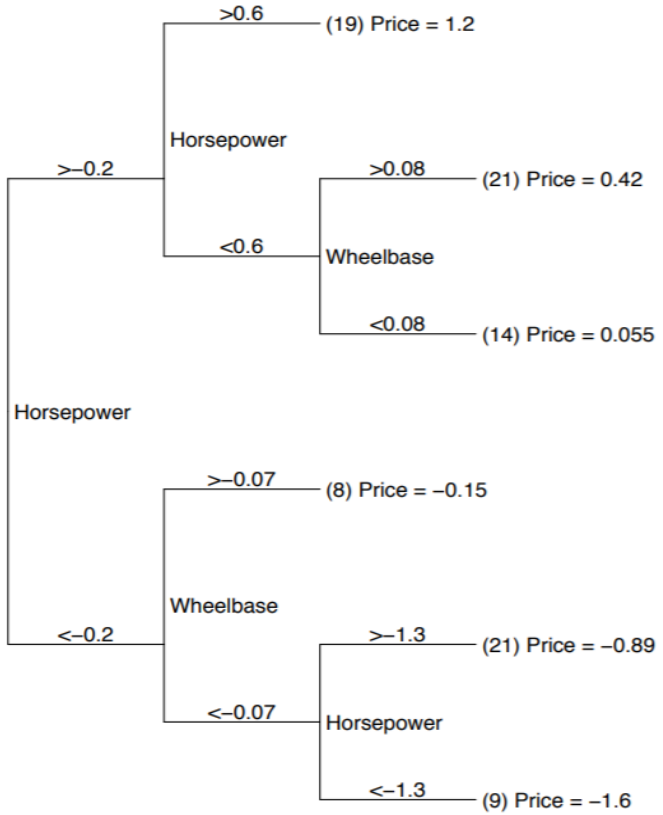
1. Draw a bootstrap sample (sample data with replacement) $\mathcal{B}_b$ of the same size n from the training data
2. Train a tree f_b on $\mathcal{B}_b$, where each split is computed as follows:
   i. Randomly select m dimensions of $x \in R^d$, newly chosen for each b
   ii. Make the best split restricted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

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

References

- 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

Cross Validation

Validation:
Binary Classification Performance

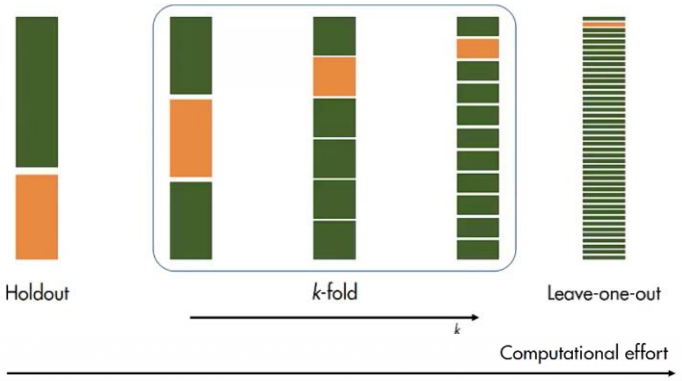
You are solving a Binary Classification Problem, Predict if a person has cancer (1) or not (0). You have learned a model using your training dataset.
How do you test your model?

Explanation:

1) **Holdout**: Randomly 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 of the K parts 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.
4) **Monte Carlo Cross Validation**: In each of multiple trials, randomly divide the dataset into training and testing sets (of prespecified sizes). Always retrain the model on the trial's training set and validate on the trial's testing set.

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
(4) Monte Carlo

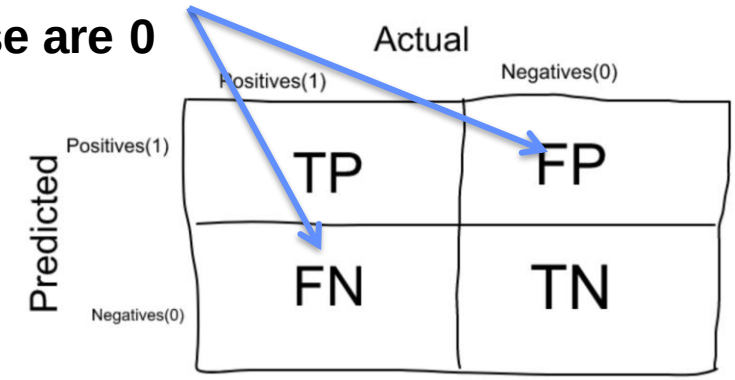


green – training data
orange - test data

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

Ideal if these are 0

Confusion Matrix



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

Metrics

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

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

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN}$$

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

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

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

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

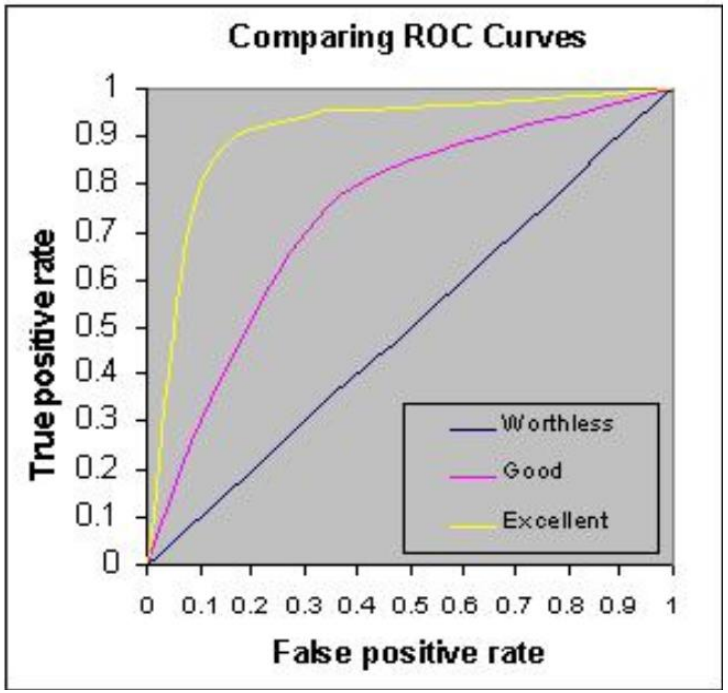
How to strike a **balance**? Use F1-score!

Why not Always Use Accuracy?

- Suppose you need to perform binary classification, where the classes are either positive or negative
- If the class distribution is imbalanced, another metric called AUC may be a better performance evaluation criterion
  - Let's say 90% of the test labels are positive
  - A very simple model that only predicts positive labels (and doesn't use the training data at all) would achieve 90% accuracy
    - If the training data is also just as skewed, then even a more complex model may end up predicting positive most of the time
  - This high accuracy is misleading since it depends solely on the distribution of the test dataset
  - By capturing results after varying the decision threshold τ in the range $[0, 1]$, AUC measures discriminative performance of the model
    - Thus, AUC is generally preferred for evaluating binary classifiers with imbalanced datasets
    - The very simple model would report a poor AUC score

ROC and AUC

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

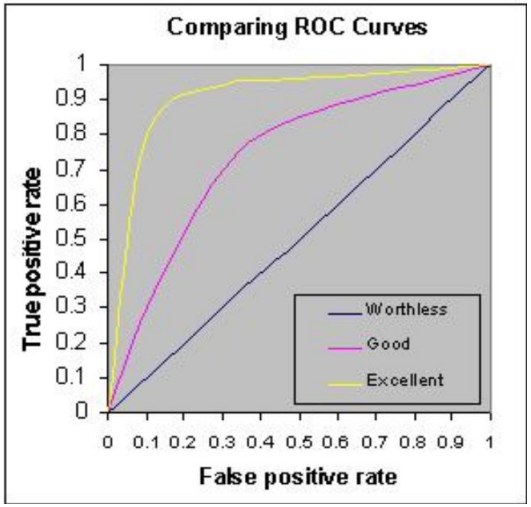


$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{TN + FP}$$

- Typically, a binary classifier has a decision boundary of $\tau = 0.5$
  - If $P(sample\ has\ positive\ label) \geq \tau$, then predict a positive label for the sample. Else, predict a negative label.
- An ROC curve is traced by tracking (FPR, TPR) for a variety of classification thresholds τ in the range $[0, 1]$
  - $\tau = 0 \Rightarrow$ Only positive labels predicted $\Rightarrow$ (FPR, TPR) = (1,1)
  - $\tau = 1 \Rightarrow$ Only negative labels predicted $\Rightarrow$ (FPR, TPR) = (0,0)

ROC and AUC

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



$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{TN + FP}$$

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**.

Area Under Curve (AUC) in [0,1]:
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