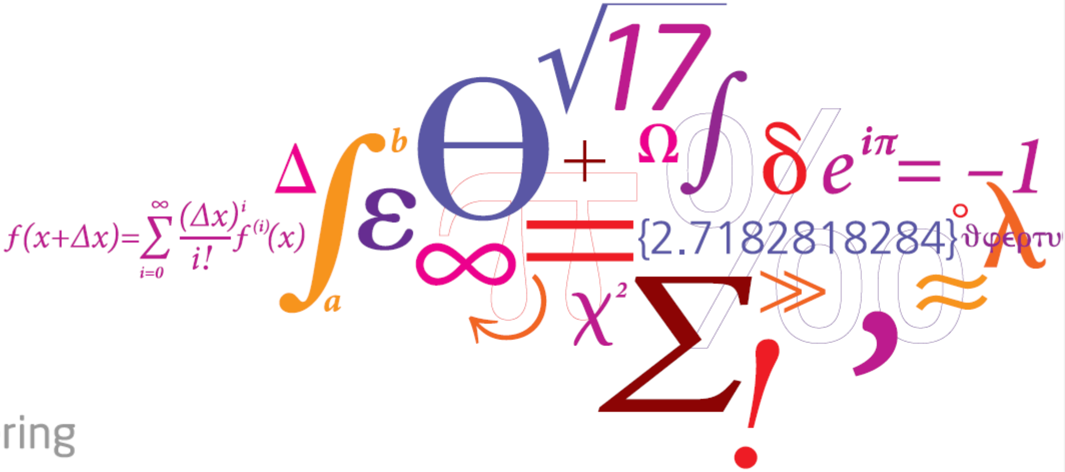
Introduction to Business Analytics

Lecture 4 – Decision Trees and Ensembling



DTU Management Engineering
Department of Management Engineering

Outline

- Decision Trees
    - What is a decision tree classifier
- Ensemble methods
    - Bagging
    - Random Forest
    - Gradient Boosting
- Performance Metrics
    - Threshold Metrics
    - Ranking Metrics
    - Probability Metrics

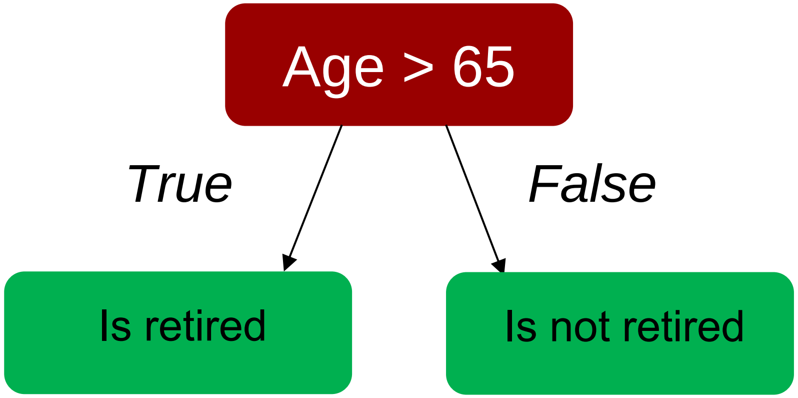
Learning Objectvies

- Be able to explain how a decision tree is built from the raw data
- Explain the main limitations of Decision Trees and some ways to overcome them
- Discuss the difference between decision trees with bagging, random forest, and gradient boosting
- Distinguish the difference type of performance metrics used to evaluate a classifier
- Apply different classifiers to real data, and rank their performances using ROC and Precision-Recall curve

Decision Trees

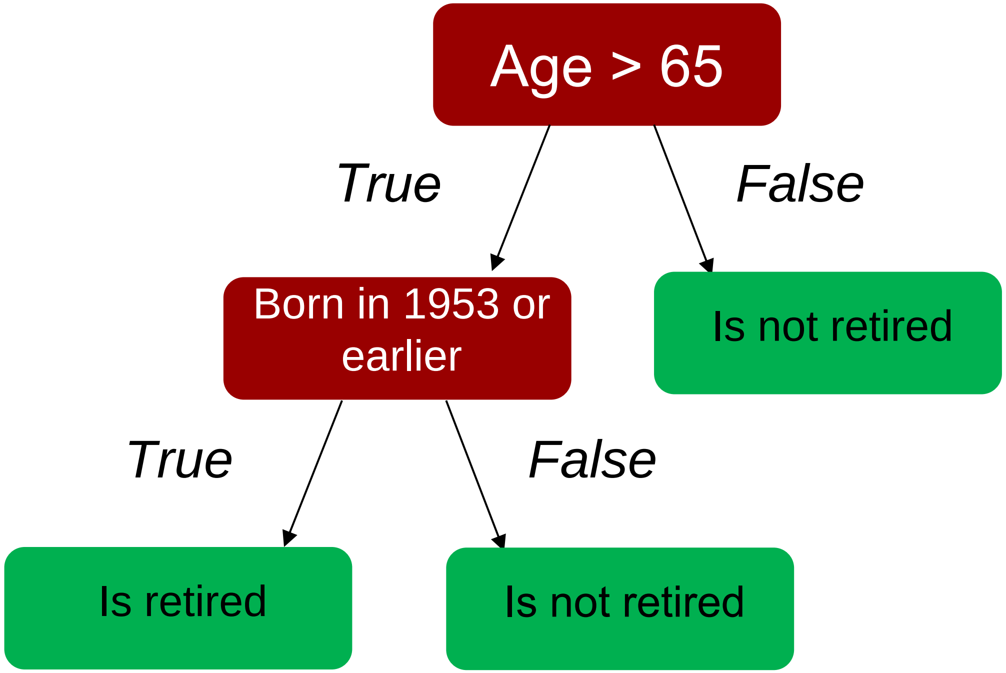
Decision Trees

- A decision tree splits the feature space into distinct non-overlapping regions
- A simple example of decision tree is



Decision Trees

- A decision tree splits or segments the feature (predictor) space into distinct non-overlapping regions
- A simple example of decision tree is



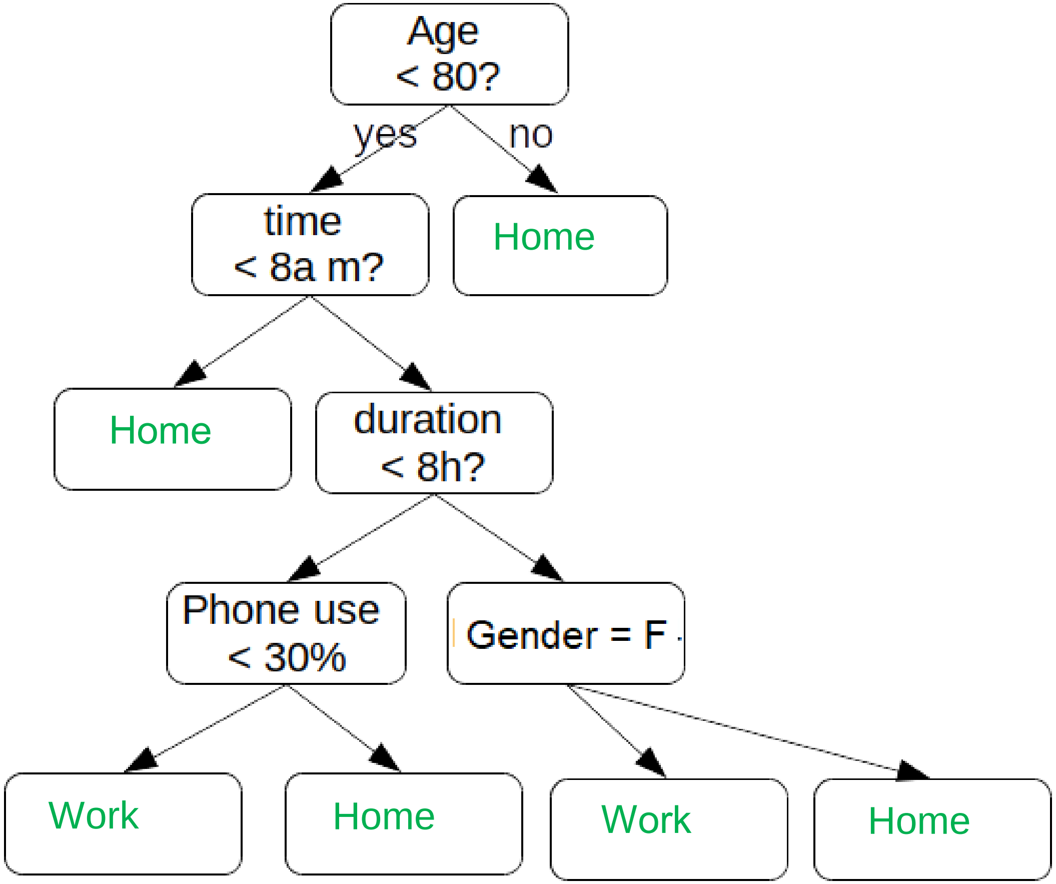
- However, more complex trees can explain more about the problem
- If the decision tree predicts a number (instead of a category) then it is called a Decision Tree Regressor (more in the next lectures)

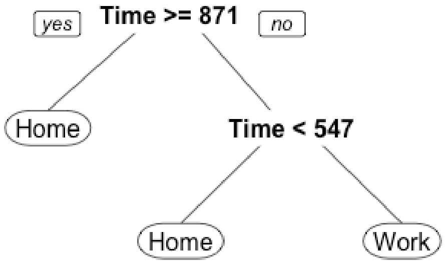
Decision Trees

- These are two example of more complex decision trees. They try to classify activities into ‘home’ or ‘work’

- Decision Trees can combine numerical data (e.g., Age, phone use) and yes/no data (Gender)

- Numerical Data can compare multiple times Decision compares multiple times





Decision Trees

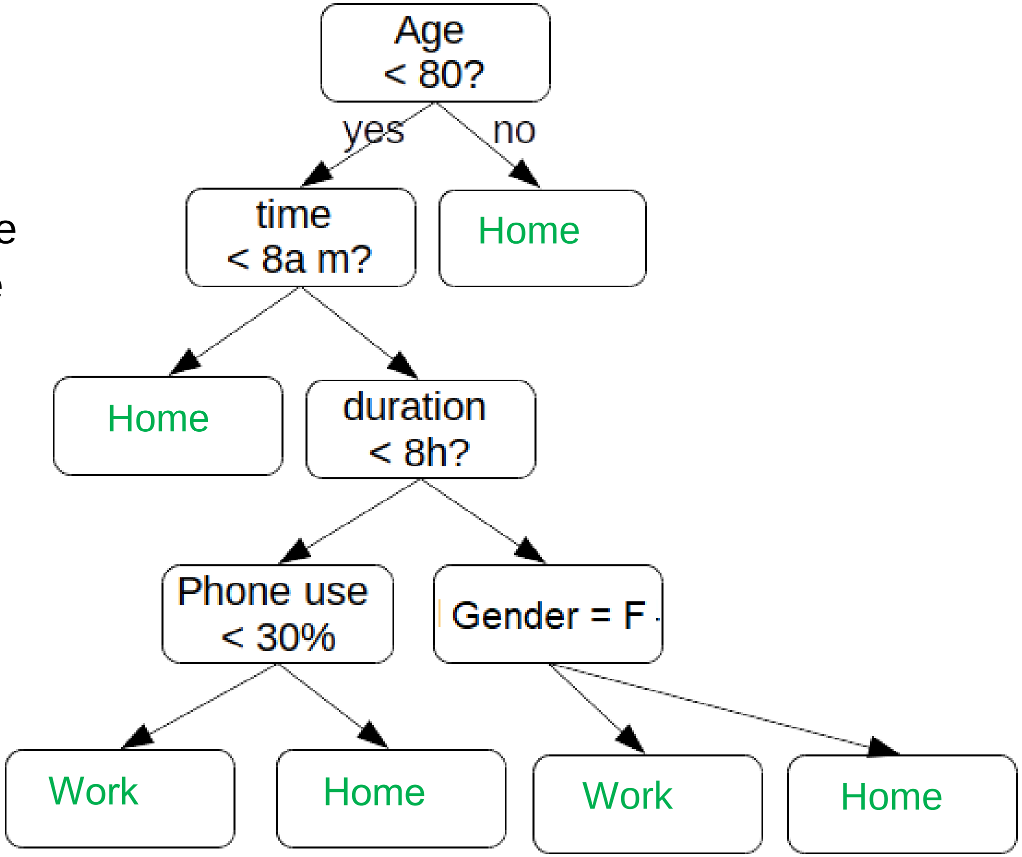
- *Grow a tree:* While many possibilities exist, we mostly use a top down approach and recursive binary splitting.

- *Top down:* We start from the top.

- *Recursive:* At each level we split our data into two using the feature that best separates the data

- Start at the top of the tree, select the best predictor and cut-point at which to split

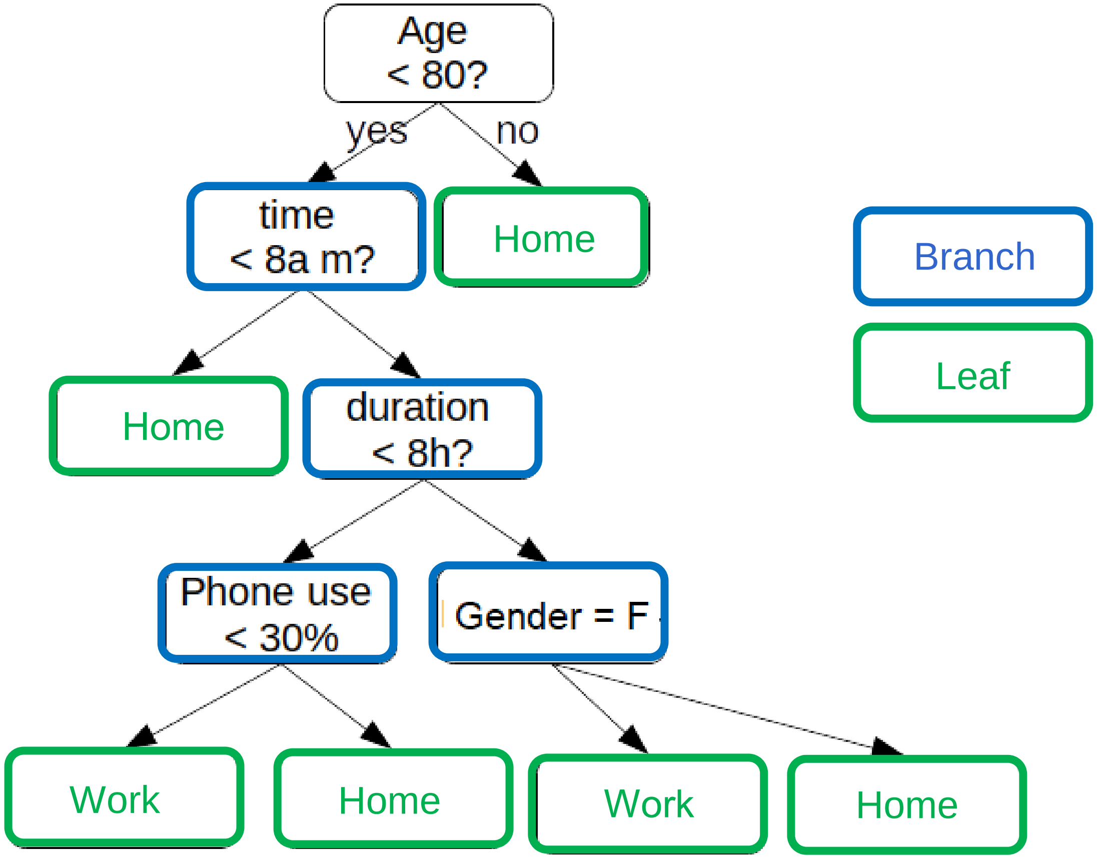
- Repeat the process until a stopping criterion is reached: for example, no region (leaf node) contains more than 10 observations



Decision Trees

- *Grow a tree:* While many possibilities exist, we mostly use a top down approach and recursive binary splitting.

- Usually, left is yes (or True) and right is no (or False)
- The top ‘node’ of the tree is called **Root Node**
- All the intermediate nodes are called **Internal Nodes**, or **branches**.
- The ending Node, is called **Leaf Node**, or **Leaf**.



Decision Trees: Building a Decision Tree

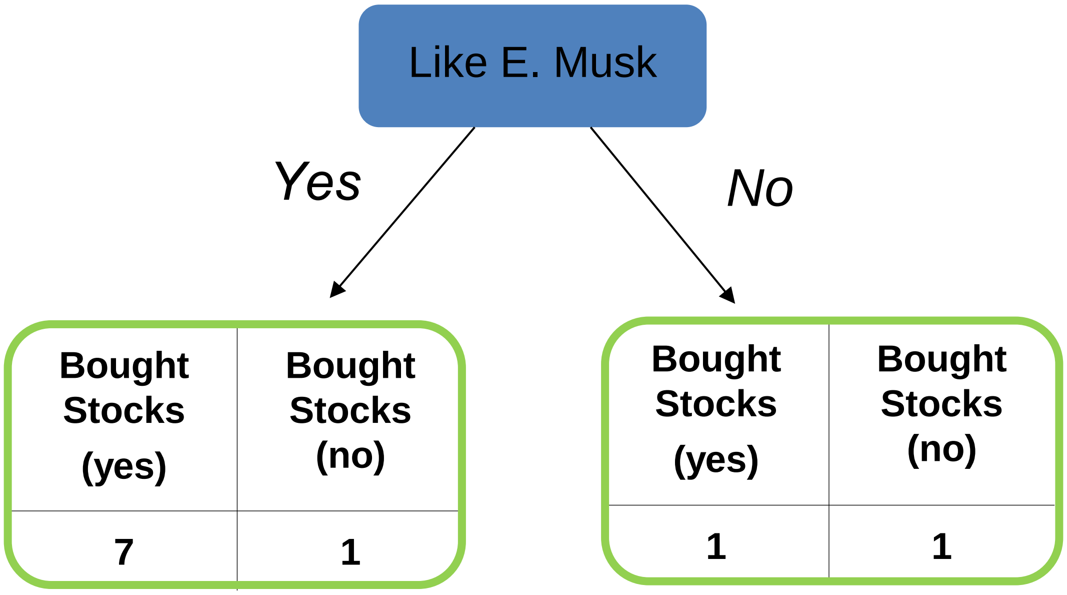
- *Grow a tree:* Let's build a Decision Tree from some data.
- We will use this data to build a classification tree to predict whether a certain person will buy or not Tesla Stocks
- To start, we need which feature should be the Root of our tree

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
Yes	No	19	Yes
No	Yes	55	Yes
Yes	Yes	33	Yes
Yes	Yes	40	Yes
Yes	Yes	55	Yes
Yes	Yes	56	Yes
No	No	10	No
No	Yes	19	No
Yes	Yes	26	Yes
Yes	Yes	29	Yes

Decision Trees: Building a Decision Tree

- We start by building a very simple decision tree to see how well ‘Likes Elon Musk’ predicts the probability to buy Tesla Stocks

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
Yes	No	19	Yes
No	Yes	55	Yes
Yes	Yes	33	Yes
Yes	Yes	40	Yes
Yes	Yes	55	Yes
Yes	Yes	56	Yes
No	No	10	No
No	Yes	19	No
Yes	Yes	26	Yes
Yes	Yes	29	Yes



Like E. Musk

Yes

Bought Stocks (yes)	Bought Stocks (no)
7	1

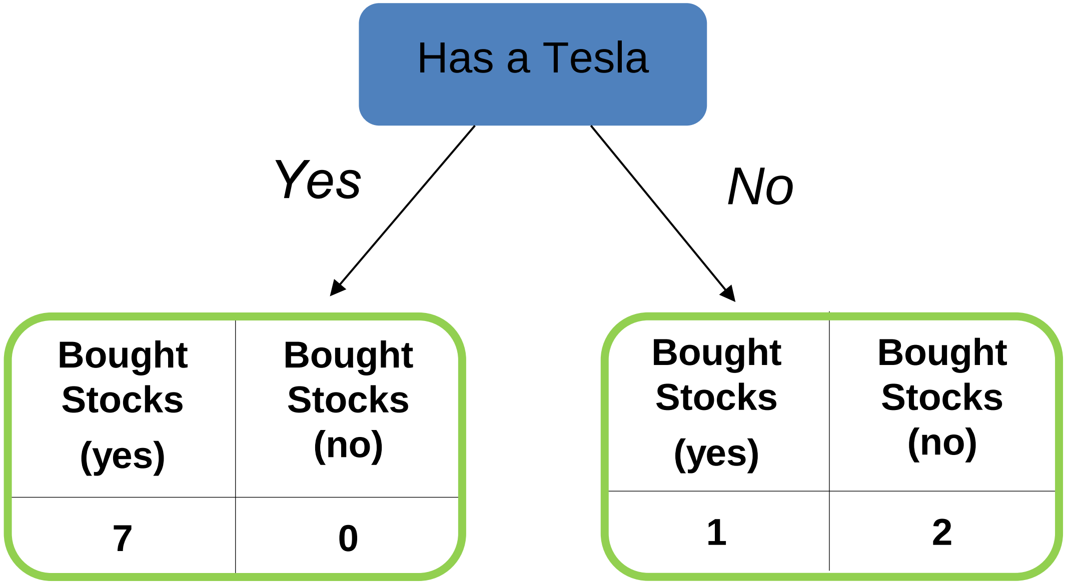
No

Bought Stocks (yes)	Bought Stocks (no)
1	1

Decision Trees: Building a Decision Tree

- We start by building a very simple decision tree to see how well ‘Likes Elon Musk’ predicts the probability to buy Tesla Stocks
- Now, we can do the same thing for ‘Has a Tesla’

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
Yes	No	19	Yes
No	Yes	55	Yes
Yes	Yes	33	Yes
Yes	Yes	40	Yes
Yes	Yes	55	Yes
Yes	Yes	56	Yes
No	No	10	No
No	Yes	19	No
Yes	Yes	26	Yes
Yes	Yes	29	Yes



Has a Tesla

Yes

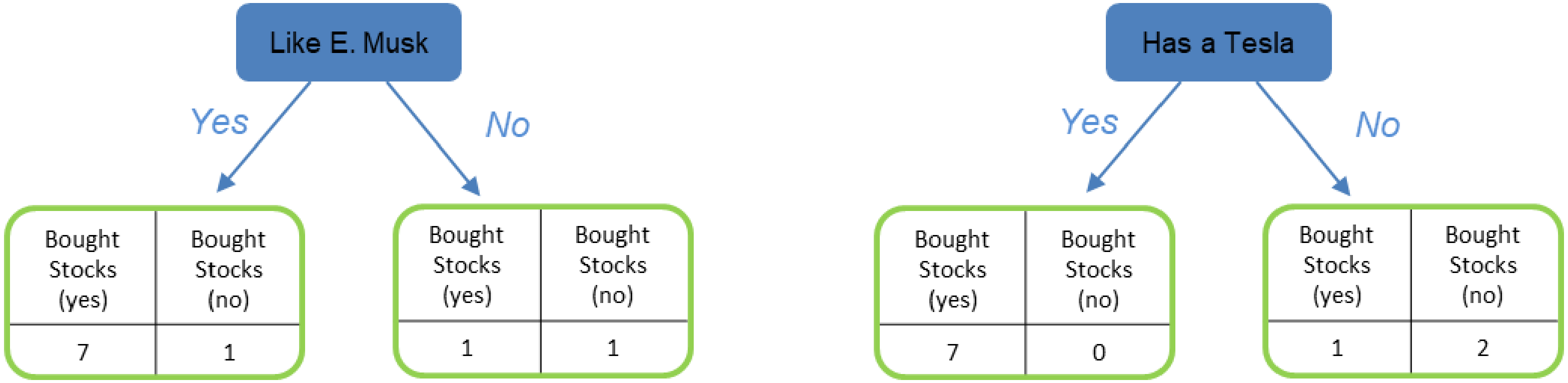
No

Bought Stocks (yes)	Bought Stocks (no)
7	0

Bought Stocks (yes)	Bought Stocks (no)
1	2

Decision Trees: Building a Decision Tree

- Neither of the two trees can perfectly classify whether a certain user will buy or not Tesla stocks
- All leaves that contain mix of yes/no, are called **Impure**
- A leaf that classify all the observations correctly, is called **Pure**



Bought Stocks (yes)	Bought Stocks (no)
7	1

Bought Stocks (yes)	Bought Stocks (no)
1	1

Bought Stocks (yes)	Bought Stocks (no)
7	0

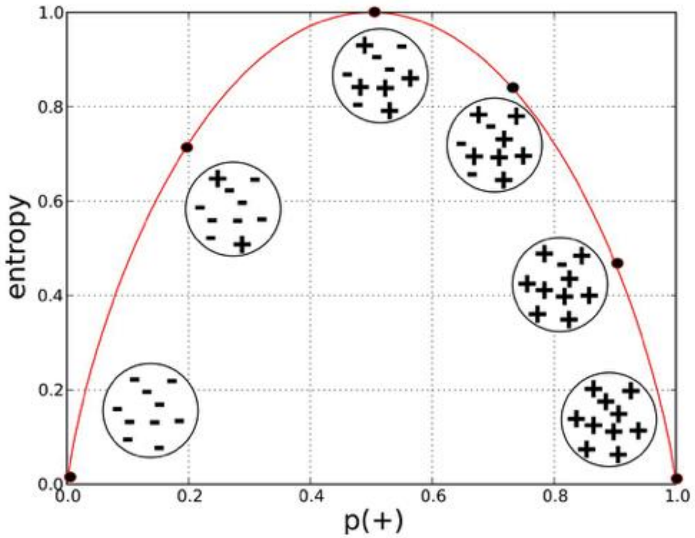
Bought Stocks (yes)	Bought Stocks (no)
1	2

- Hence, it is important to quantify the impurity of a certain leaf
  - A **splitting rule** is a way to measure the impurity and decide which feature should be used to split the data
  - Several indicators can be used, including **Entropy**, **Information Gain**, or **Gini Impurity**

Decision Trees: Building a Decision Tree

- **Entropy** is a common splitting rule, and measures the 'purity' of a leaf, and it is computed as:

$$E = \sum_{k=1}^{K} -p_k log_2(p_k)$$



- p_k is the probability/frequency of a certain class k.
- For example, for the feature 'Bought Tesla Stocks'

$$E = -\frac{8}{10}log_2\left(\frac{8}{10}\right) - \frac{2}{10}log_2\left(\frac{2}{10}\right) = 0{,}72$$

- Then the entropy follows a U shape, and it is maximum at when $p_k = 0.5$ (and 0 when the sample is 'pure')

- Hence, one common splitting rule is to minimize Entropy in each leaf

Source figure: https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision Trees: Building a Decision Tree

- **Gini index** is another common splitting rule (default in Sklearn)
- The Gini Index is composed of two parts. The actual index is computed as

$$Gini = 1 - \sum_{k=1}^{K} {p_k}^2$$

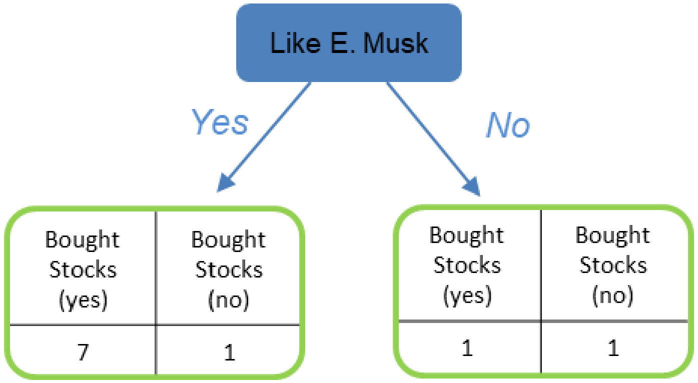
- The overall Gini index of a tree can be computed as the weighted average of the Gini index of each leaf/

$$Gini_{Tree} = \sum_{l=1}^{L} p_l \, Gini_l$$

- Where p_l is the frequency/probability of a certain leaf and $Gini_l$ is the Gini index for a specific leaf.

Decision Trees: Building a Decision Tree

- For the categorical features, we can compute it directly compute the Gini Impurity for each lea.



Bought Stocks (yes)	Bought Stocks (no)
7	1

Bought Stocks (yes)	Bought Stocks (no)
1	1

$$Gini_{Like\ Elon\_Yes} = 1 - \left(\tfrac{7}{7+1}\right)^2 - \left(\tfrac{1}{7+1}\right)^2 = 0.2188$$

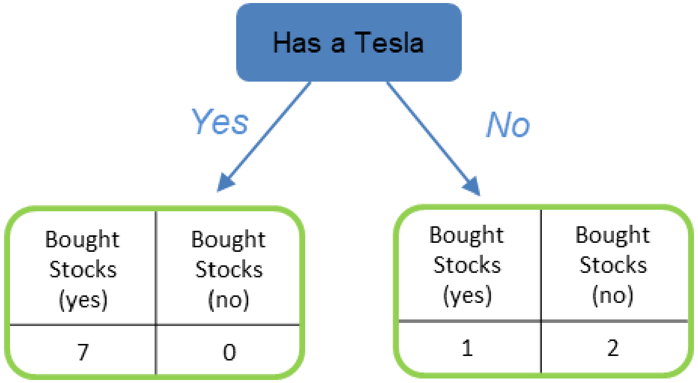
$$Gini_{Like\ Elon\_No} = 1 - (0.5)^2 - (0.5)^2 = 0.5$$

- And then compute the overall Gini Index as:

$$Gini\_index_{Like\ Elon} = \frac{8}{10}(0.21) + \frac{2}{10}(0.25) = 0.5$$

Decision Trees: Building a Decision Tree

- We can do the same for



Bought Stocks (yes)	Bought Stocks (no)
7	0

Bought Stocks (yes)	Bought Stocks (no)
1	2

$$Gini_{Like\ Elon\_Yes} = 1 - \left(\frac{7}{7}\right)^2 - \left(\frac{0}{7}\right)^2 = 0$$

$$Gini_{Like\ Elon\_No} = 1 - \left(\frac{1}{1+2}\right)^2 - \left(\frac{2}{1+2}\right)^2 = 0.44$$

- And then compute the overall Gini Index as:

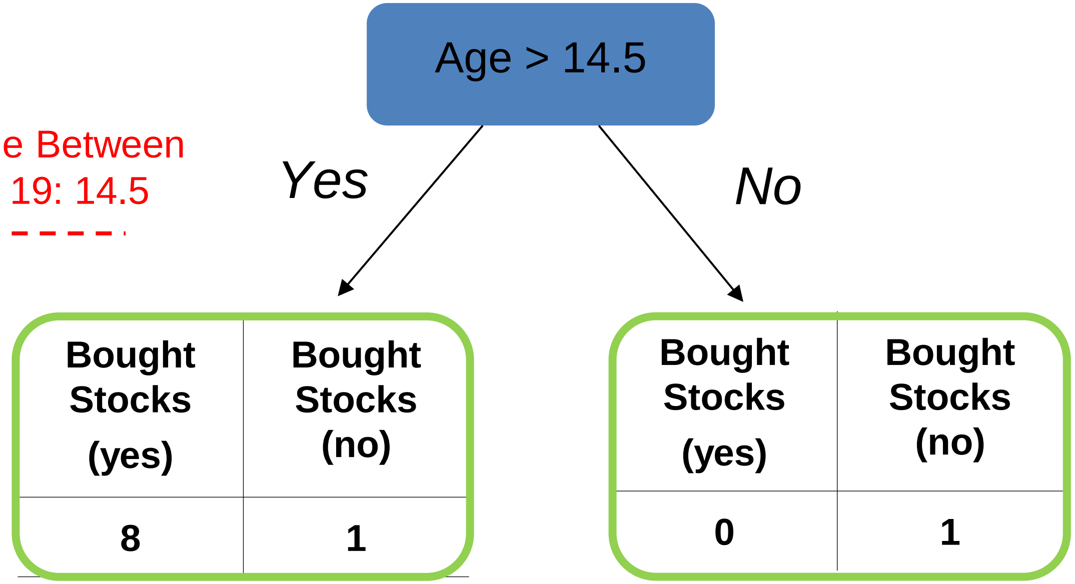
$$Gini\_index_{Like\ Elon} = \frac{7}{10}(0) + \frac{3}{10}(0.44) = 0.133$$

Decision Trees: Building a Decision Tree

- For continuous features, it works the same way but we need to use thresholds.
- We sort the data in ascending order, and then compute the Average between the first and the second observation
- The Average is used as threshold to split the data
- The Gini index is computed as before

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
No	No	10	No
Yes	No	19	Yes
No	Yes	19	No
Yes	Yes	26	Yes
Yes	Yes	29	Yes
Yes	Yes	33	Yes
Yes	Yes	40	Yes
No	Yes	55	Yes
Yes	Yes	55	Yes
Yes	Yes	56	Yes

Average Between 10 and 19: 14.5



Age > 14.5

Yes

Bought Stocks (yes)	Bought Stocks (no)
8	1

No

Bought Stocks (yes)	Bought Stocks (no)
0	1

$$Gini_{Age>14\_Yes} = 1 - \left(\frac{8}{8+1}\right)^2 - \left(\frac{1}{8+1}\right)^2 = 0.19$$

$$Gini_{Age>14\_No} = 1 - (0)^2 - (1)^2 = 0$$

Decision Trees: Building a Decision Tree

- Then the overall Gini Index is:

$$Gini\_index_{Age>14} = \frac{9}{10}(0.19) + \frac{1}{10}(0) = 0.17$$

- We can compute the Gini Index for all different thresholds
- The threshold with the lowest Gini Index is the best one, hence we should use either Age 19 or 22.5 as thresholds.

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
No	No	10	No
Yes	No	19	Yes
No	Yes	19	No
Yes	Yes	26	Yes
Yes	Yes	29	Yes
Yes	Yes	33	Yes
Yes	Yes	40	Yes
No	Yes	55	Yes
Yes	Yes	55	Yes
Yes	Yes	56	Yes

Gini Index

Age > 14.5 → 0.17

Age > 19 → 0.13

Age > 22.5 → 0.13

Age > 27.5 → 0.20

Age > 31 → 0.24

Age > 36.5 → 0.27

Age > 47.5 → 0.29

Age > 55 → 0.31

Age > 55.5 → 0.31

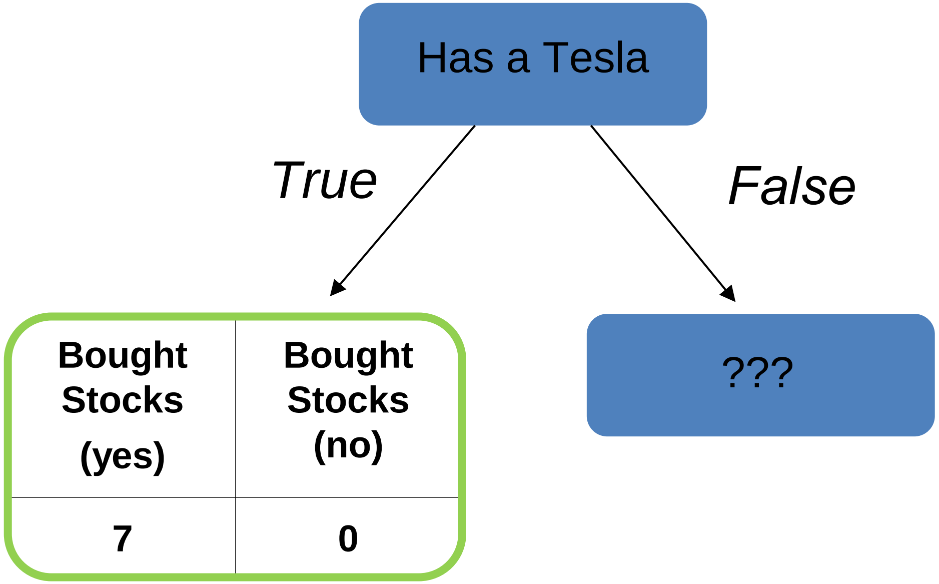
Decision Trees: Building a Decision Tree

- We can now compare the Gini index of the different threes. The root node, is the one that performs best.

$$Gini\_index_{Like\ Elon} = 0.27 \qquad Gini\_index_{Has\ a\ Tesla} = 0.13$$

$$Gini\_index_{Age>19} = 0.13$$

- The root node is ‘Has a Tesla’



Bought Stocks (yes)	Bought Stocks (no)
7	0

Decision Trees: Building a Decision Tree

- We repeat the same procedure and compute the Gini Index for each feature on the reduced data set (only users who do not have a Tesla).

Likes Elon Musk	Age	Bought Tesla Stocks
No	10	No
No	19	Yes
Yes	19	No
Yes	26	Yes
Yes	29	Yes
Yes	33	Yes
Yes	40	Yes
Yes	55	Yes
Yes	55	Yes
Yes	56	Yes

$$Gini\_index_{Like\ Elon\_Yes} = 0.33$$

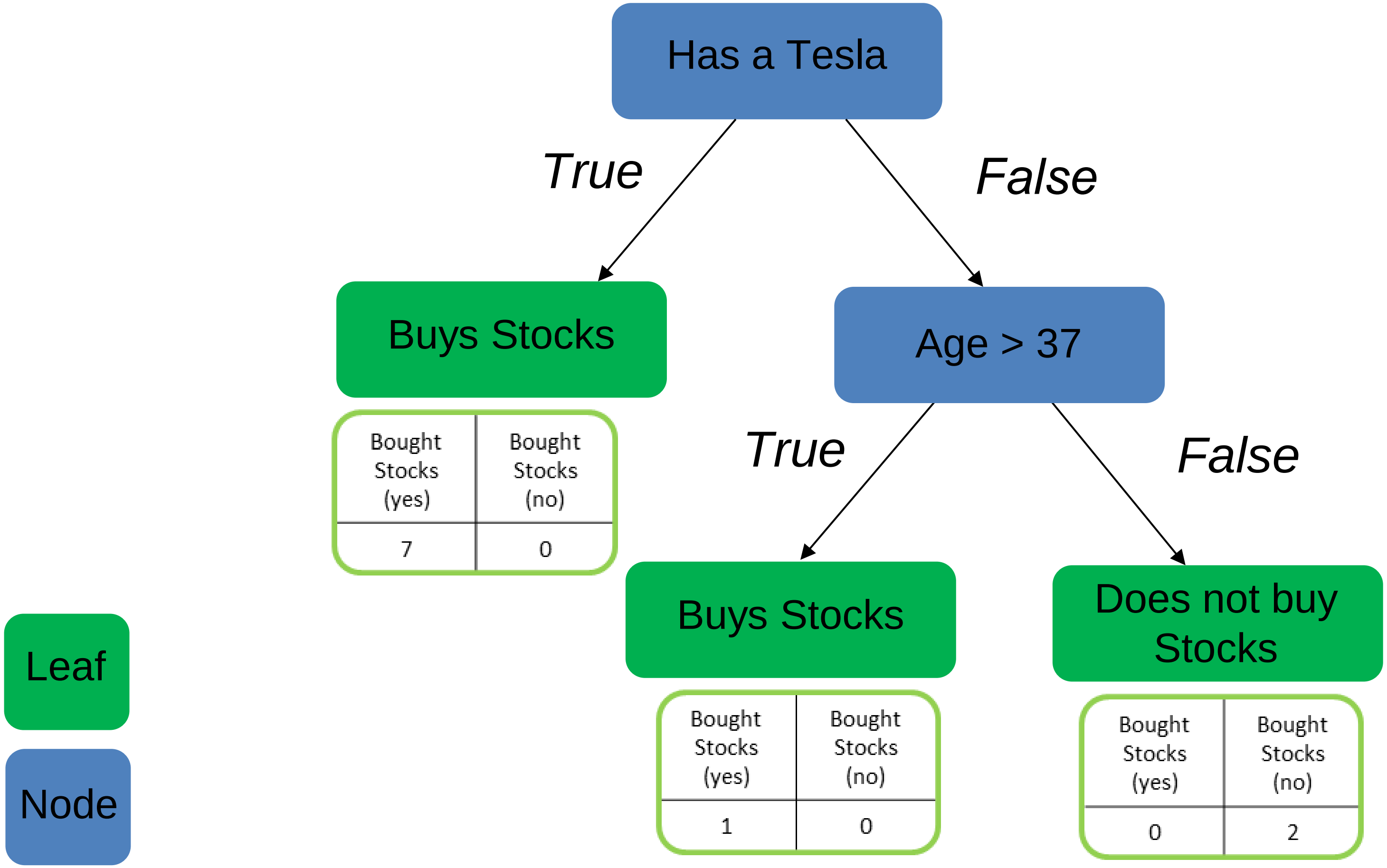
$$Gini\_index_{Age>14} = 0.33$$

$$Gini\_index_{Age>37} = 0$$

- Which lead to choosing Age>37 as the next variable

Decision Trees: Building a Decision Tree

- Since all the leaves are pure, we can stop terminate the procedure
- The Tree is complete.



Has a Tesla

True

False

Buys Stocks

Bought Stocks (yes)	Bought Stocks (no)
7	0

Age > 37

True

False

Buys Stocks

Bought Stocks (yes)	Bought Stocks (no)
1	0

Does not buy Stocks

Bought Stocks (yes)	Bought Stocks (no)
0	2

Leaf

Node

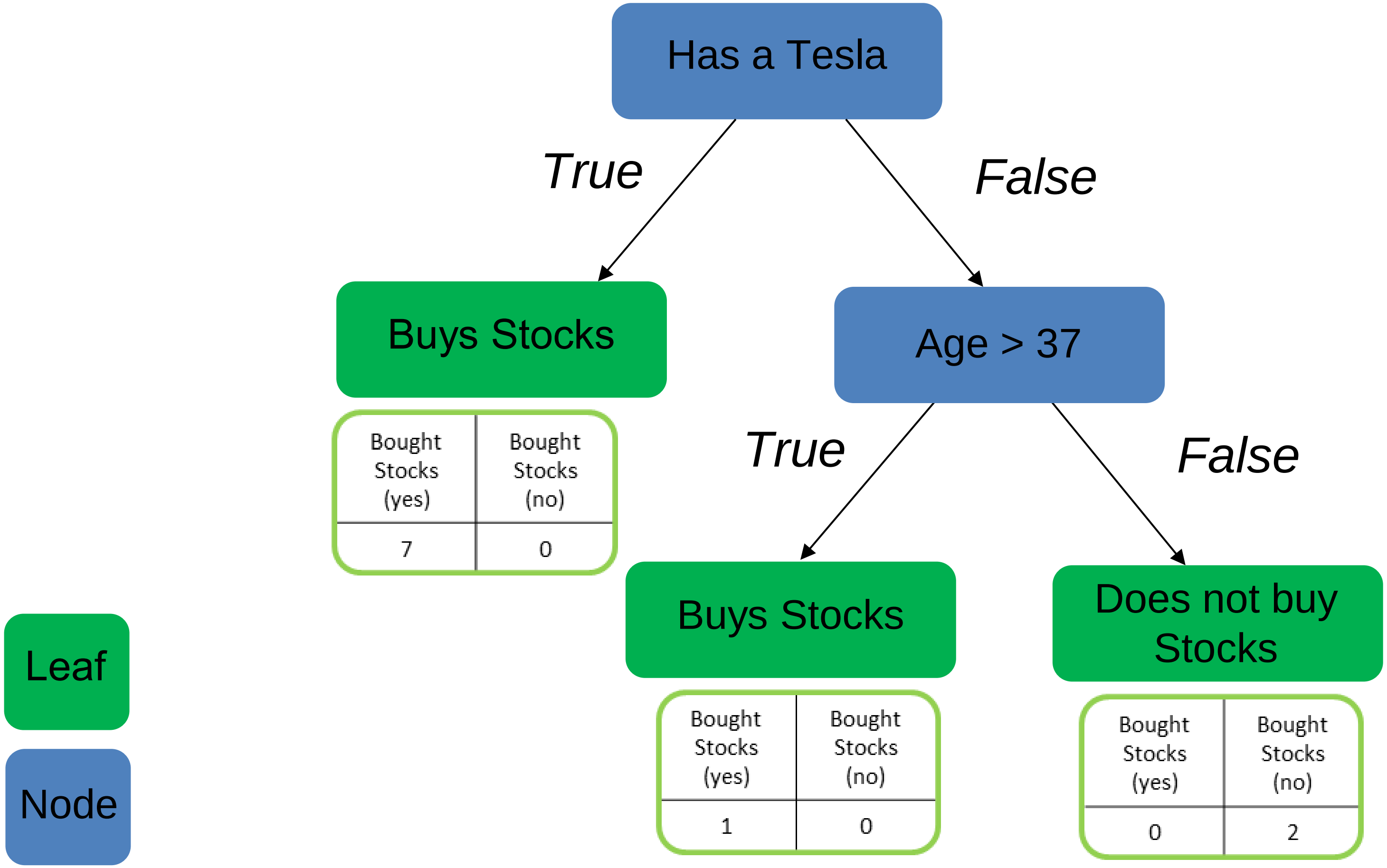
Playtime

- Open the "4. DecisionTree.ipynb" notebook
- Do Part 1
- Estimated duration: MAX 20 min

Ensemble methods

Problems with decision trees

- The problem with decision trees is that they fit too well the data
- Can you point out at least two problems with this Classifier?



Has a Tesla

True

Buys Stocks

Bought Stocks (yes)	Bought Stocks (no)
7	0

False

Age > 37

True

Buys Stocks

Bought Stocks (yes)	Bought Stocks (no)
1	0

False

Does not buy Stocks

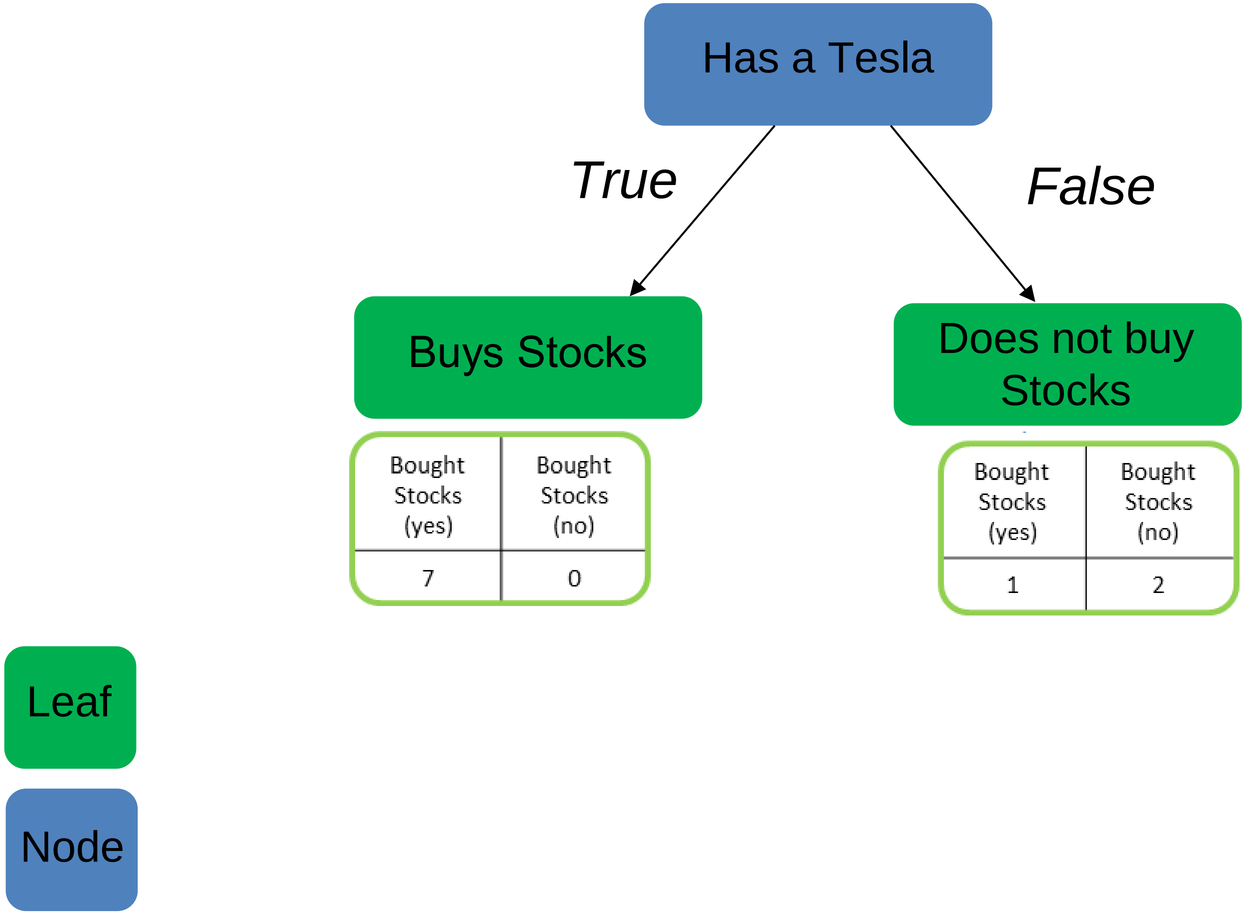
Bought Stocks (yes)	Bought Stocks (no)
0	2

🟩 Leaf

🟦 Node

Problems with decision trees

- The problem with decision trees is that they fit too well the data
- Can you point out at least two problems with this Classifier?

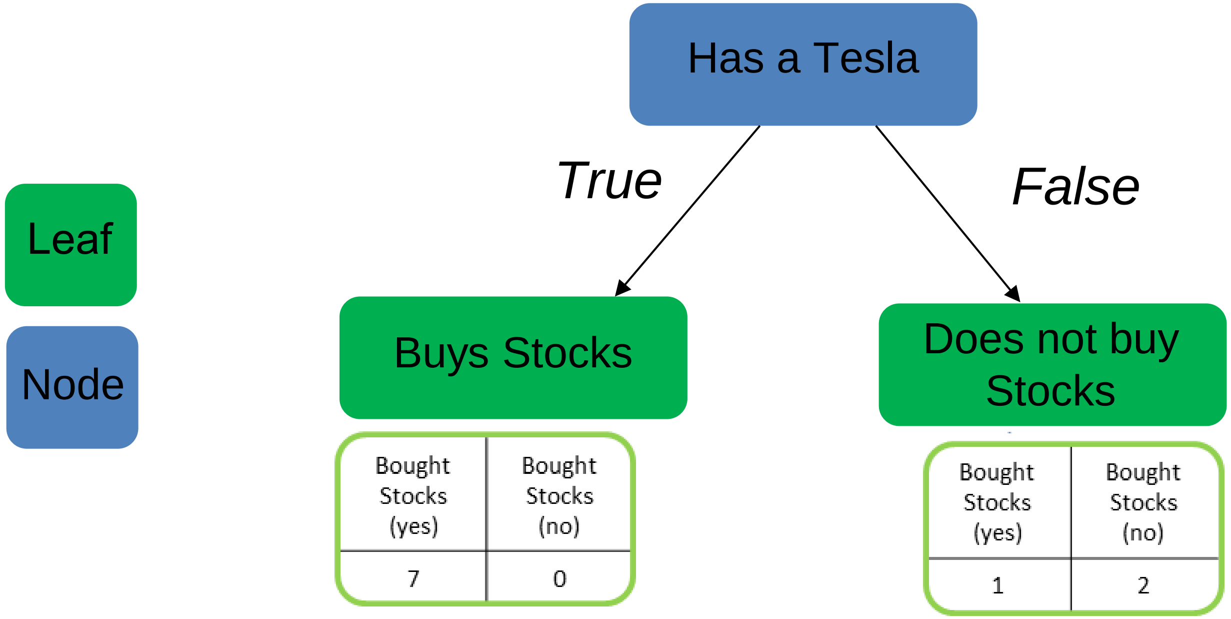


Bought Stocks (yes)	Bought Stocks (no)
7	0

Bought Stocks (yes)	Bought Stocks (no)
1	2

Problems with decision trees

- The problem with decision trees is that they fit too well the data
- Decision Trees will prioritize the larger class if the data in imbalanced
- We may have leaves created to classify a single data point (overfitting)
- The simplest solution is to set a limit to how much the three may grow, or how many observations should be in one leaf.



Leaf

Node

Has a Tesla

True

False

Buys Stocks

Bought Stocks (yes)	Bought Stocks (no)
7	0

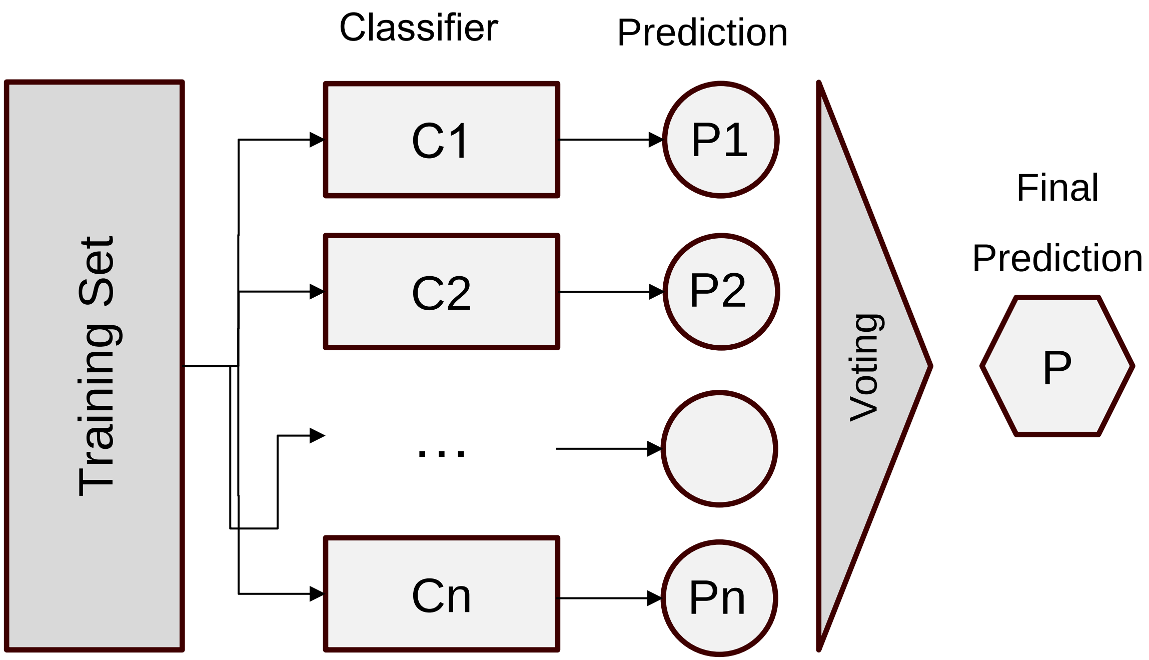
Does not buy Stocks

Bought Stocks (yes)	Bought Stocks (no)
1	2

- **A more advance technique is Pruning**, which implies removing part of the decision tree that are not relevant (leaves, but also branches). Pruning leads to a simpler tree with lower probabilities of overfitting

Ensemble methods

- The main idea of Ensemble Methods is that a group of **weak learners** (e.g., decision trees) come together to form a **strong Lerner**



- The final prediction is the average of each prediction.
- NOTE: We usually say that each weak learner ‘vote’ and that the final prediction is the result of the voting. Averaging is not the only way of voting (e.g., weighted average)

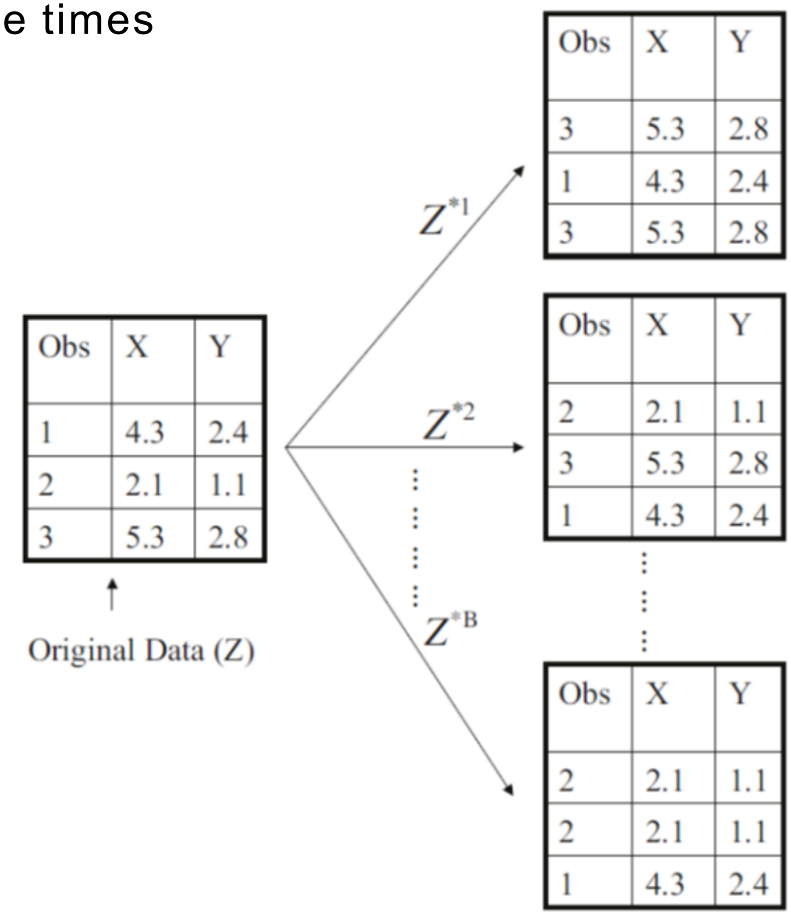
Ensemble methods: Bootstrapping

- Bootstrapping is often used with ensemble methods.
- Bootstrapping creates artificial data by sampling randomly from the original data set.
- Bootstrapping uses **sampling with replacement**, which means that the same observation can appear multiple times

- A dataset generated using bootstrapping is called **bootstrapped** (BS) **dataset**.

- If the data is small (e.g., Tesla stock), the bootstrapped dataset has the same size as the original dataset (10 datapoints)

- If the data is very large, the bootstrapped dataset can be smaller.



Original Data (Z)

Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Z^{*1}

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Z^{*2}

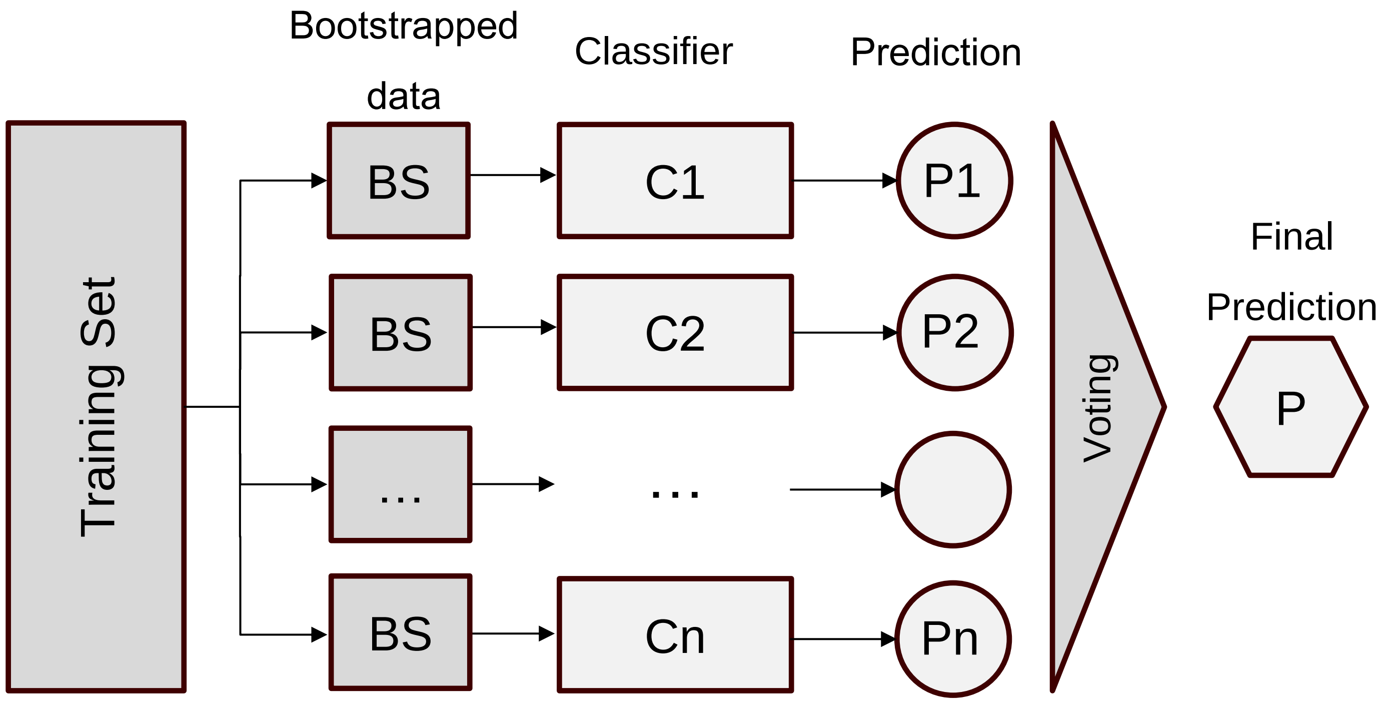
Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Z^{*B}

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

Ensemble methods: Bagging and Random Forest

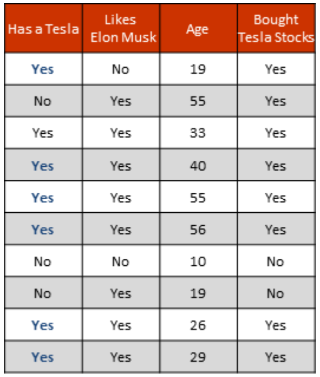
- Bagging combines Bootstrapping and Decision Trees



- Essentially, first a set of n bootstrapped datasets is created
- For each of them, we grow a full decision tree (**weak learner**)
- Because Decision Trees have low bias but high variance, the idea is to start with a simple model that has high variance and low bias, and reduce the variance by averaging or majority voting (**strong learner**)

Ensemble methods: Bagging and Random Forest

- **Random Forest** is essentially an enhancement over Basic Decision Tree with Bagging.
- **Step 1:** We build a bootstrapped dataset
- **Step 2**: Remove some features from the dataset (e.g., Has a Tesla)
- **Step 3:** Build a decision tree classifier on the dataset**.**

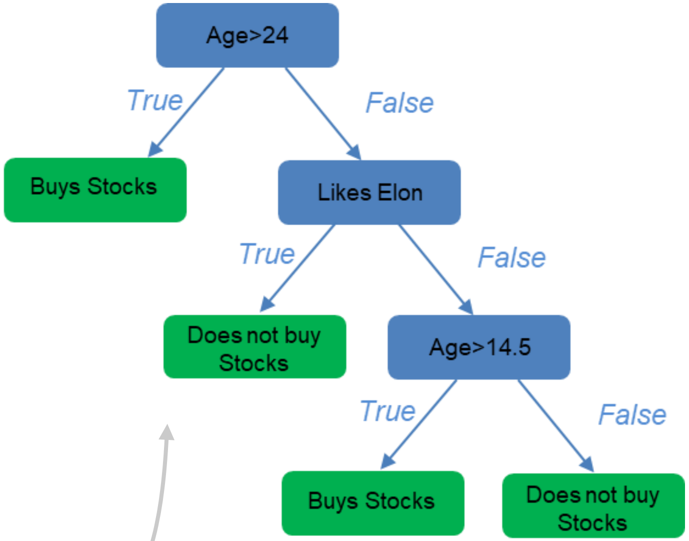


Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
Yes	No	19	Yes
No	Yes	55	Yes
Yes	Yes	33	Yes
Yes	Yes	40	Yes
Yes	Yes	55	Yes
Yes	Yes	56	Yes
No	No	10	No
No	Yes	19	No
Yes	Yes	26	Yes
Yes	Yes	29	Yes

original dataset

bootstrapped dataset

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
Yes	No	**19**	**Yes**
No	Yes	**55**	**Yes**
No	No	**10**	**No**
No	No	**10**	**No**
Yes	No	**19**	**Yes**
Yes	Yes	**56**	**Yes**
Yes	Yes	**29**	**Yes**
No	Yes	**19**	**No**
No	Yes	**19**	**No**
Yes	Yes	**29**	**Yes**



Ensemble methods: Bagging and Random Forest

- **Step 4:** We can use the ‘out of bag’ data for validation. If the out data outside the bootstrapped data sets are poorly predicted, we can change the settings of the model (e.g., remove more/less features from the original data set, build deeper trees, etc)

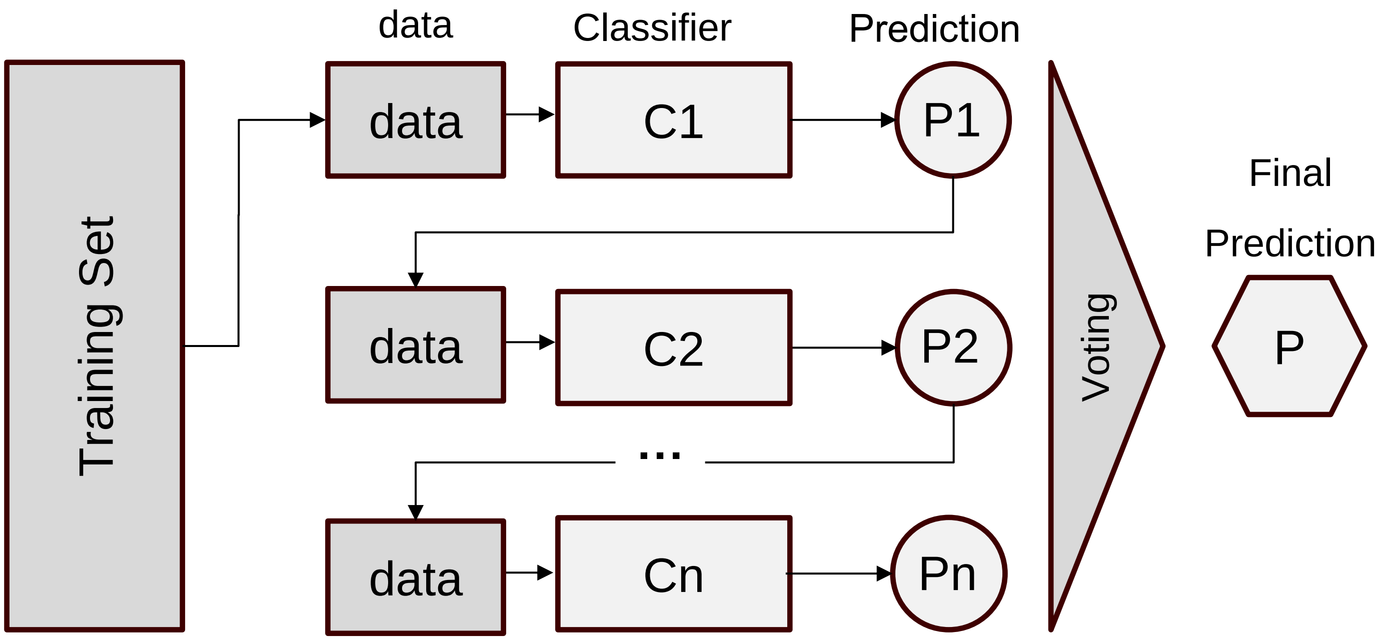
- Because Random Forest starts with very large trees and decision trees tend to overfit, Random Forest start from low bias (compared to the training set) but high variance. It then reduces it by averaging the results of the individual trees.

The importance of being ‘sloppy’:

- As Random forest removes some features, trees are less correlated. It is more beneficial to aggregate their responses
- The model is efficient on large data set
- It is robust to missing information

Ensemble methods: Boosting and Gradient Boosting

- Bosting improves the prediction power by sequentially training weak learners, each focused on correcting the mistakes of the previous learner.



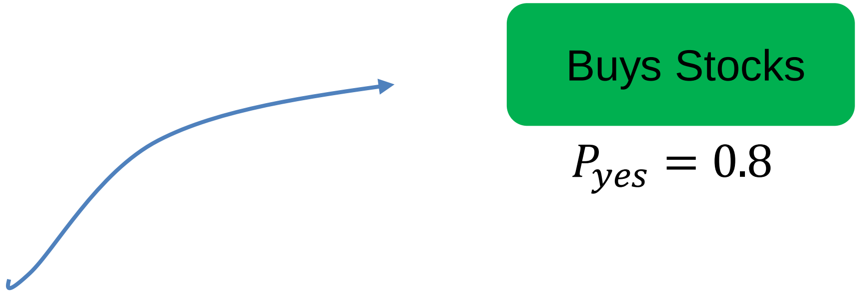
- We start with a weak learner (e.g., shallow tree)
- Some observations are properly classified, others are not
- In the next iteration, weights are adjusted to give more relevance to those data points that were poorly classified. The process continues until a stopping criteria is reach (max number of learners, no improvement)
- We start from a simple classifier with low variance and high bias, and try to reduce the bias by developing more complex models

Ensemble methods: Boosting and Gradient Boosting

- Gradient Boosting is a Boosting technique that uses Decision Trees.
- However, decision trees usually have **low variance** and **high Bias.**
- Gradient Boosting takes that into account by having small trees (usually between 8 and 32 leaves) and having a learning rate to reduce overfitting.

- **Step 1:** Conceptually, gradient boosting start from a ‘naïve’ prediction based on the training data

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks
No	No	10	0
Yes	No	19	1
No	Yes	19	0
Yes	Yes	26	1
Yes	Yes	29	1
Yes	Yes	33	1
Yes	Yes	40	1
No	Yes	55	1
Yes	Yes	55	1
Yes	Yes	56	1

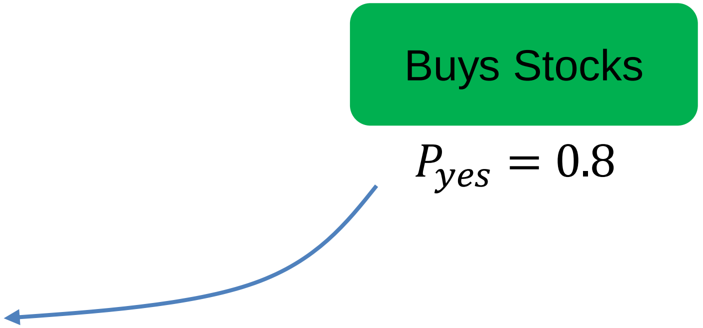


$P_{yes} = 0.8$

Ensemble methods: Boosting and Gradient Boosting

- **Step 1:** Conceptually, gradient boosting start from a 'naïve' prediction based on the training data
- **Step 2:** Based on the probabilities, it compute the residual error between the observed and predicted behavior

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks	Redidual
No	No	10	0	-0.8
Yes	No	19	1	0.2
No	Yes	19	0	-0.8
Yes	Yes	26	1	0.2
Yes	Yes	29	1	0.2
Yes	Yes	33	1	0.2
Yes	Yes	40	1	0.2
No	Yes	55	1	0.2
Yes	Yes	55	1	0.2
Yes	Yes	56	1	0.2



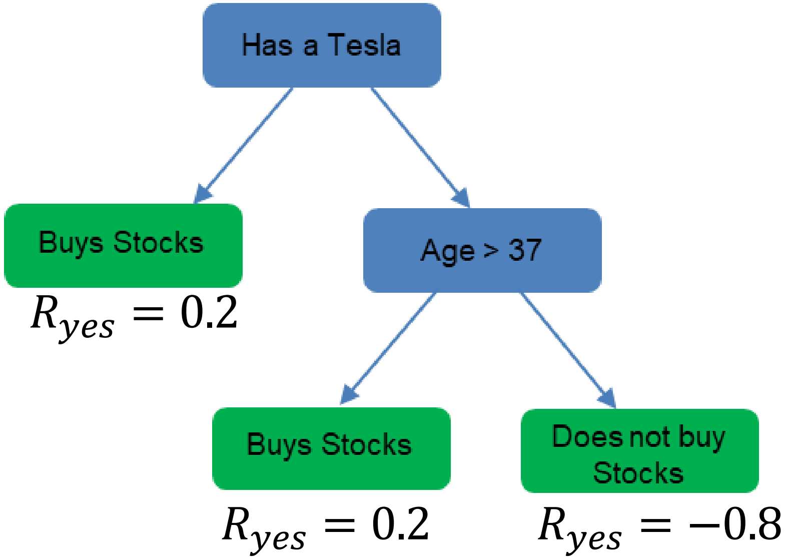
Buys Stocks

$P_{yes} = 0.8$

Ensemble methods: Boosting and Gradient Boosting

- **Step 1:** Conceptually, gradient boosting start from a 'naïve' prediction based on the training data
- **Step 2:** Based on the probabilities, it compute the residual error between the observed and predicted behavior
- **Step 3:** Build a Decision Tree to predict the deviation

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks	Redidual
No	No	10	0	-0.8
Yes	No	19	1	0.2
No	Yes	19	0	-0.8
Yes	Yes	26	1	0.2
Yes	Yes	29	1	0.2
Yes	Yes	33	1	0.2
Yes	Yes	40	1	0.2
No	Yes	55	1	0.2
Yes	Yes	55	1	0.2
Yes	Yes	56	1	0.2



$R_{yes} = 0.2$

$R_{yes} = 0.2$

$R_{yes} = -0.8$

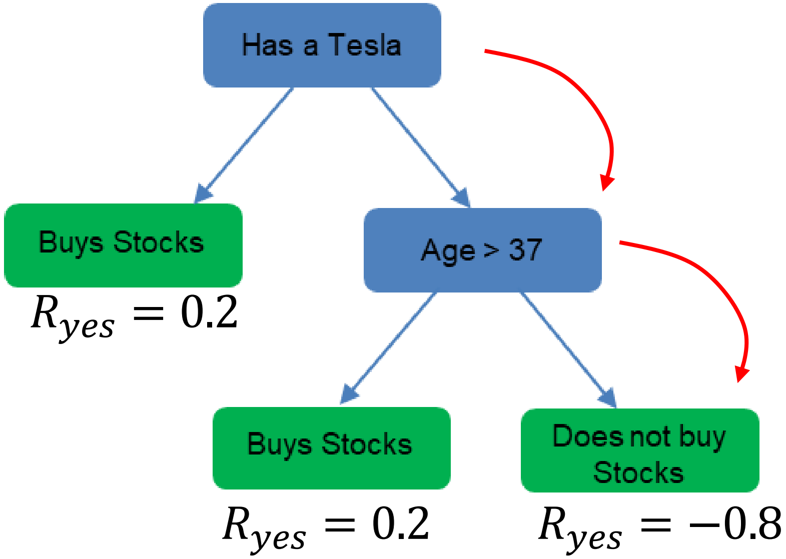
Ensemble methods: Boosting and Gradient Boosting

- **Step 1:** Conceptually, gradient boosting start from a 'naïve' prediction based on the training data
- **Step 2:** Based on the probabilities, it compute the residual error between the observed and predicted behavior
- **Step 3:** Build a Decision Tree to predict the deviation
- **Step 4:** It obtains a new prediction by using the output of the new tree to correct the initial probability

For example, the prediction for the first observation would be:

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks	Redidual
No	No	10	0	-0.8

$$P = 0.8 - 0.8 = 0$$



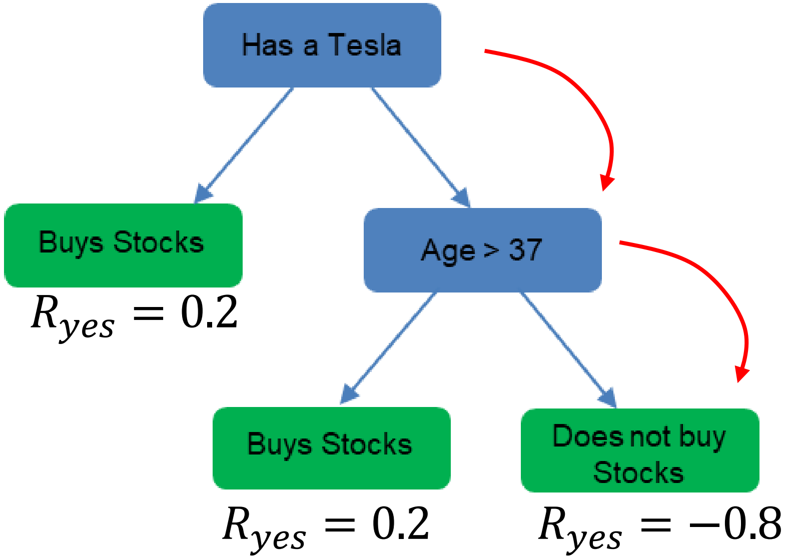
Ensemble methods: Boosting and Gradient Boosting

- **Step 1:** Conceptually, gradient boosting start from a 'naïve' prediction based on the training data
- **Step 2:** Based on the probabilities, it compute the residual error between the observed and predicted behavior
- **Step 3:** Build a Decision Tree to predict the deviation
- **Step 4:** It obtains a new prediction by using the output of the new tree to correct the initial probability

Because the model would overfit the training data, we include a learning rate, 0.1 in this case, to avoid overfitting

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks	Redidual
No	No	10	0	0.2

$$P = 0.8 + 0.1\ (-0{,}8) = 0.72$$



Ensemble methods: Boosting and Gradient Boosting

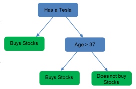
- **Step 1:** Conceptually, gradient boosting start from a 'naïve' prediction based on the training data
- **Step 2:** Based on the probabilities, it compute the residual error between the observed and predicted behavior
- **Step 3:** Build a Decision Tree to predict the deviation
- **Step 4:** It obtains a new prediction by using the output of the new tree to correct the initial probability
- **Step 5:** Compute the new residuals and build a new tree to predict them

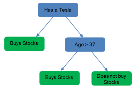
Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks	Predicted	Residual
No	No	10	0	0.72	-0.72
Yes	No	19	1	0.82	0.18
No	Yes	19	0	0.72	-0.72
Yes	Yes	26	1	0.82	0.18
Yes	Yes	29	1	0.82	0.18
Yes	Yes	33	1	0.82	0.18
Yes	Yes	40	1	0.82	0.18
No	Yes	55	1	0.82	0.18
Yes	Yes	55	1	0.82	0.18
Yes	Yes	56	1	0.82	0.18

Ensemble methods: Boosting and Gradient Boosting

- The final equation is obtained by summing up the initial ‘naïve’ estimates, with the outputs of all the decision trees.

Has a Tesla	Likes Elon Musk	Age	Bought Tesla Stocks	Redidual
No	No	10	0	-0.8

$P = 0.8 + 0.1\,(-0.8) + 0.1($ 

$) + \cdots + 0.1($ 

$) = 1$

- In reality, the procedure is a bit more complicated. While Gradient boosting for Regression works as above, for classification it works slightly different. In fact, the procedure presented could provide probabilities larger than 1 (or lower than zero).

Ensemble methods: Boosting and Gradient Boosting

- **Step 1:** Compute the Logarithm of the Odds. This is the ‘naïve estimate’ (not the actual probability)

$$Odds = \; ln\left(\frac{p(y=1)}{1-p(y=1)}\right)$$

- **Step 2:** Based on the current Odds, compute the probability and the residual error between the observed and predicted behavior

$$p(y=1) = \frac{1}{1+e^{-Odds}} \qquad\qquad residual \; = y_i - p(y=1)$$

- **Step 3:** Build a Decision Tree to predict the residuals
- **Step 4:** Since the Decision Tree predicts probabilities, we need to transform its before adding it to the logarithm of the odds.

$$X_i = \frac{\sum Residual_i}{\sum Previous\ Probability_i \times (1 - Previous\ Probability_i)}$$

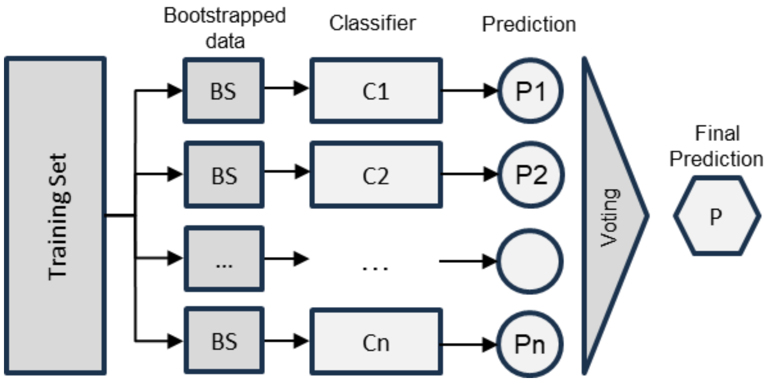
Where $Previous\ Probability_i$ is the probability that estimated by the model without the decision tree (at the first iteration, the naïve estimate). If multiple observations are in the same leaf, we sum them.

- **Step 5:** We obtain the new Odds and restart from Step 2.

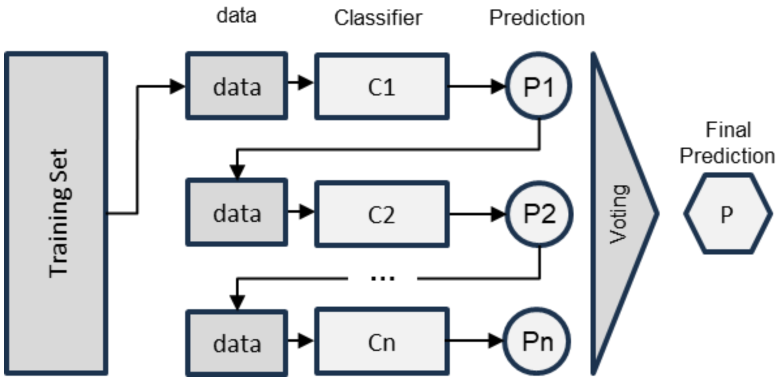
$$Odds = \; ln\left(\frac{p(y=1)}{1-p(y=1)}\right) + 0.1\, X_i$$

Comparison: Boosting Vs Bagging

Bagging



Boosting



	Weak Learners	Bias-Variance	Model
Bagging	Independent, trained in parallel	Reduce Variance	Random Forest
Boosting	Dependent, trained sequentially	Reduce Bias	Gradient Boosting

Playtime

- Open the "4. DecisionTree.ipynb" notebook
- Do Part 2
- Estimated duration: 30 min

Classification Performance Metrics

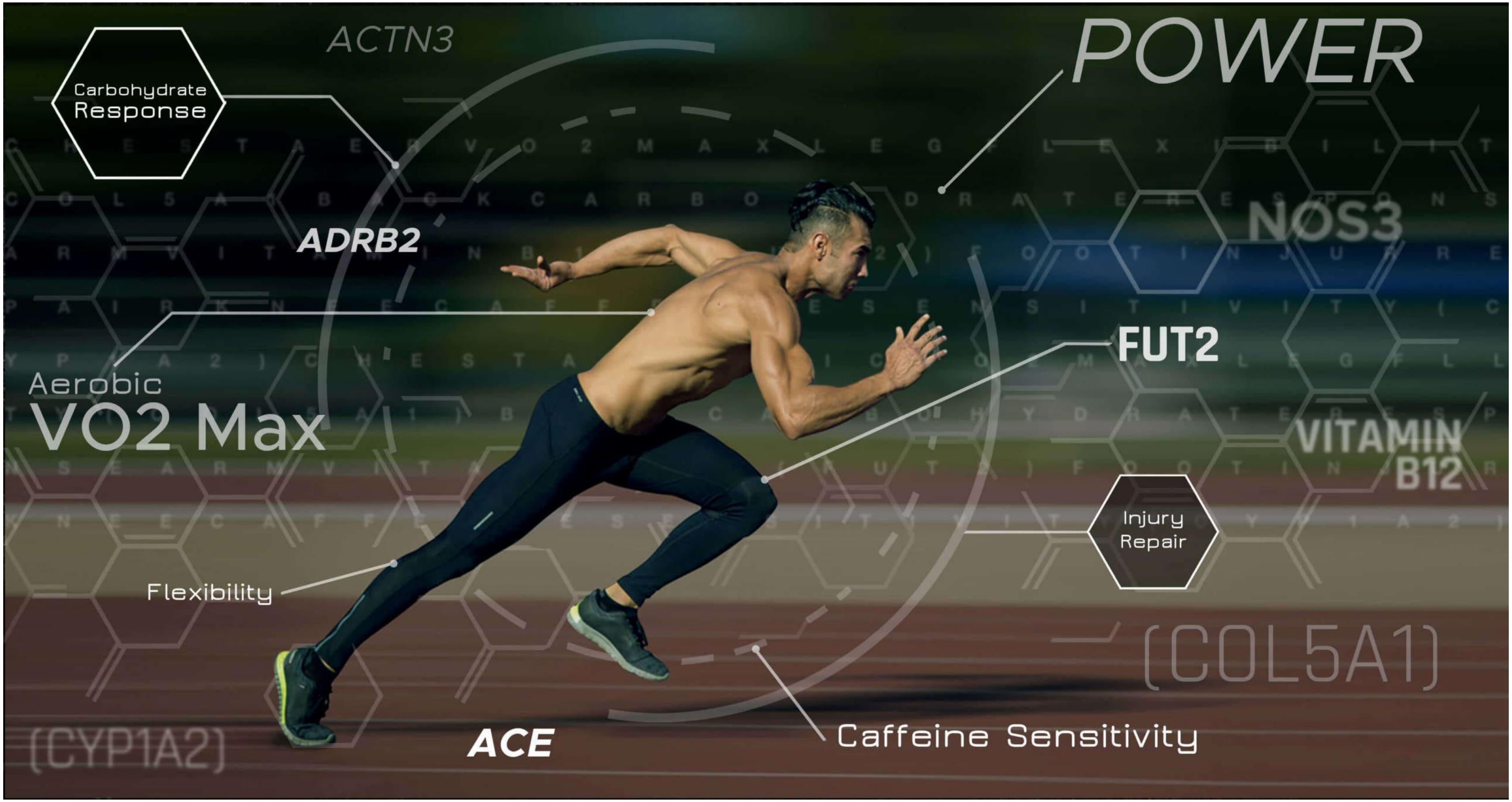


Image Source: https://healthcoach.clinic/wp-content/uploads/2020/07/Sports-Performance-Part-1-scaled.jpg

Binary classification

- A classification model f is fitted to the data to predict **score** $s^{(i)} = f\left(x^{(i)}\right)$ measuring model’s belief that the sample $x^{(i)}$ belongs to the class 1 (“positive” class)
- The predicted class is based on the **decision threshold** $\tilde{s}$: $x^{(i)}$ belongs to the positive class if $s^{(i)} \geq \tilde{s}$ and to the “negative” class otherwise

- Given the predicted scores and the threshold, a **confusion matrix** can be constructed

		Actual	
		Positive class	**Negative class**
Predicted	**Positive class**	True Positives (TP)	False Positives (FP) “Type I error”
	Negative class	False Negatives (FN) “Type II error”	True Negatives (TN)

Binary classification performance metrics

- Based on the scores and class thresholds, three types of performance metrics can be considered:

1. **Threshold metrics**
   Measure the classification prediction errors

2. **Ranking metrics**
   How well the model ranks the examples and separates classes

3. **Probability metrics**
   Measure the deviation of the predicted probabilities from the true ones

1. Threshold metrics

- Metrics based on a threshold and a qualitative understanding of error.

- *Business value = f(CM)*

- **Examples:** Almost all classification problems

- **Most popular metrics:**
  - Accuracy
  - Precision
  - Recall
  - F-score

https://www.sciencedirect.com/science/article/abs/pii/S0167865508002687

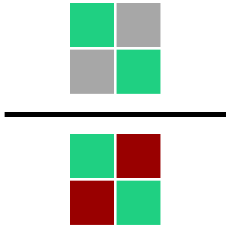
https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

1. Threshold metrics: Accuracy

	True 1	True 0
Pred 1	TP	FP
Pred 0	FN	TN

- Accuracy is the proportion of the correct predictions

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



- **Comments:**
  - Both errors are equally important ~ “Making any error is costly”
  - Optimising for binary/categorical cross-entropy = optimising for accuracy
  - Usually, the default metrics
  - Works poorly for imbalanced problems (always predict the major class and get high accuracy)

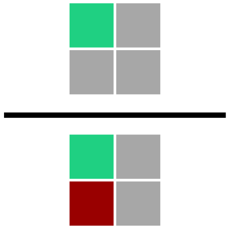
https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

1. Threshold metrics: Recall

	True 1	True 0
Pred 1	TP	FP
Pred 0	FN	TN

- Recall is the proportion of the true positives which are correctly predicted

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



- **Comments:**
  - FN are more important ~ “Misclassifying true positives is costly”
  - Recall is a valid choice of evaluation metric when we want to capture as many positives as possible. For example: If we are building a system to predict if a person has cancer or not, we want to capture the disease even if we are not very sure.
  - Recall is 1 if we predict 1 for all examples.

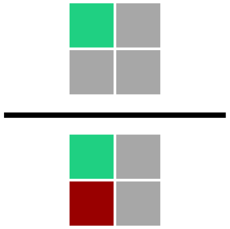
https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

1. Threshold metrics: Recall

	True 1	True 0
Pred 1	TP	FP
Pred 0	FN	TN

- Recall is the proportion of the true positives which are correctly predicted

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



- **Comments:**
  - FN are more important ~ “Misclassifying true positives is costly”
  - Recall is a valid choice of evaluation metric when we want to capture as many positives as possible. For example: If we are building a system to predict if a person has cancer or not, we want to capture the disease even if we are not very sure.
  - Recall is 1 if we predict 1 for all examples.

	True 1	True 0
Pred 1	100	100
Pred 0	0	0

→

$$Accuracy = 0{,}5$$

$$\text{Recall}(TPR) = 1$$

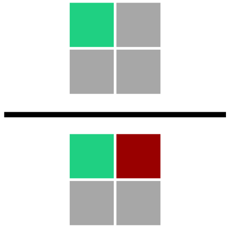
https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

1. Threshold metrics: Precision

	True 1	True 0
Pred 1	TP	FP
Pred 0	FN	TN

- Precision is the proportion of predicted positives which are truly positive

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



- **Comments:**
  - FP are more important ~ “Misclassifying true negatives is costly”
  - Precision is a valid choice when we want to be very sure of our prediction.
  - Example: If we build a system to predict if we should decrease the credit limit on specific bank accounts, we want to be very sure or it may result in customer dissatisfaction. Being very precise means our model will leave a lot of credit defaulters untouched.

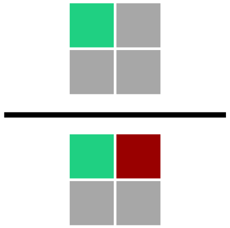
https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

1. Threshold metrics: Precision

	True 1	True 0
Pred 1	TP	FP
Pred 0	FN	TN

- Precision is the proportion of predicted positives which are truly positive

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



- **Comments:**
  - FP are more important ~ “Misclassifying true negatives is costly”
  - Precision is a valid choice when we want to be very sure of our prediction.
  - Example: If we build a system to predict if we should decrease the credit limit on specific bank accounts, we want to be very sure or it may result in customer dissatisfaction. Being very precise means our model will leave a lot of credit defaulters untouched.

	True 1	True 0
Pred 1	100	100
Pred 0	0	0

→

$$Accuracy = 0{,}5$$

$$\text{Recall}(TPR) = 1$$

$$Precision = 0{,}5$$

	True 1	True 0
Pred 1	1	0
Pred 0	99	100

→

$$Accuracy = 0{,}505$$

$$\text{Recall}(TPR) = 0{,}1$$

$$Precision = 1$$

https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

1. Threshold metrics: F-score

	True 1	True 0
Pred 1	TP	FP
Pred 0	FN	TN

- F-score is a weighted harmonic average between precision and recall

- $F_\beta = (1+\beta^2)\frac{\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}}$
  - $\beta = 1 \rightarrow$ Precision and Recall are equally important
  - $\beta = 0.5 \rightarrow$ Precision is more important
  - $\beta = 2 \rightarrow$ Recall is more important

- **Comments:**
  - The F1 score maintains a balance between the precision and recall for your classifier.

https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

2. Ranking metrics

- Ranking metrics are important when good **class separation** is crucial

- *Business value = f(ranking)*

- **Business cases:** Recommend closing a facility, distribute limited marketing budget among potential churns, …

- **Most popular metrics:**
  - Receiver Operating Characteristic (ROC) curve
  - Precision-Recall (PR) curve
  - Area Under a Curve (AUC) is calculated to get a single number

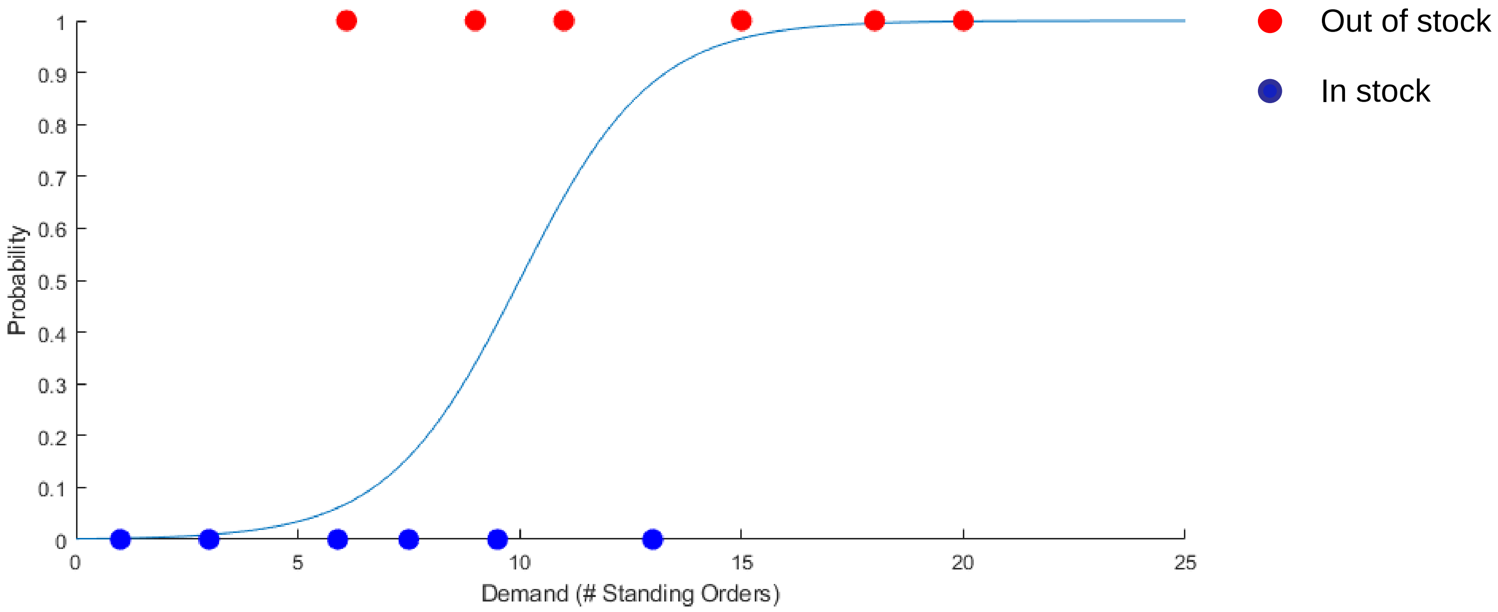
https://www.sciencedirct.com/science/article/abs/pii/S0167865508002687

https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

2. Ranking metrics: General approach

Example - Inventory Problem

Calculate when to order new products based standing orders

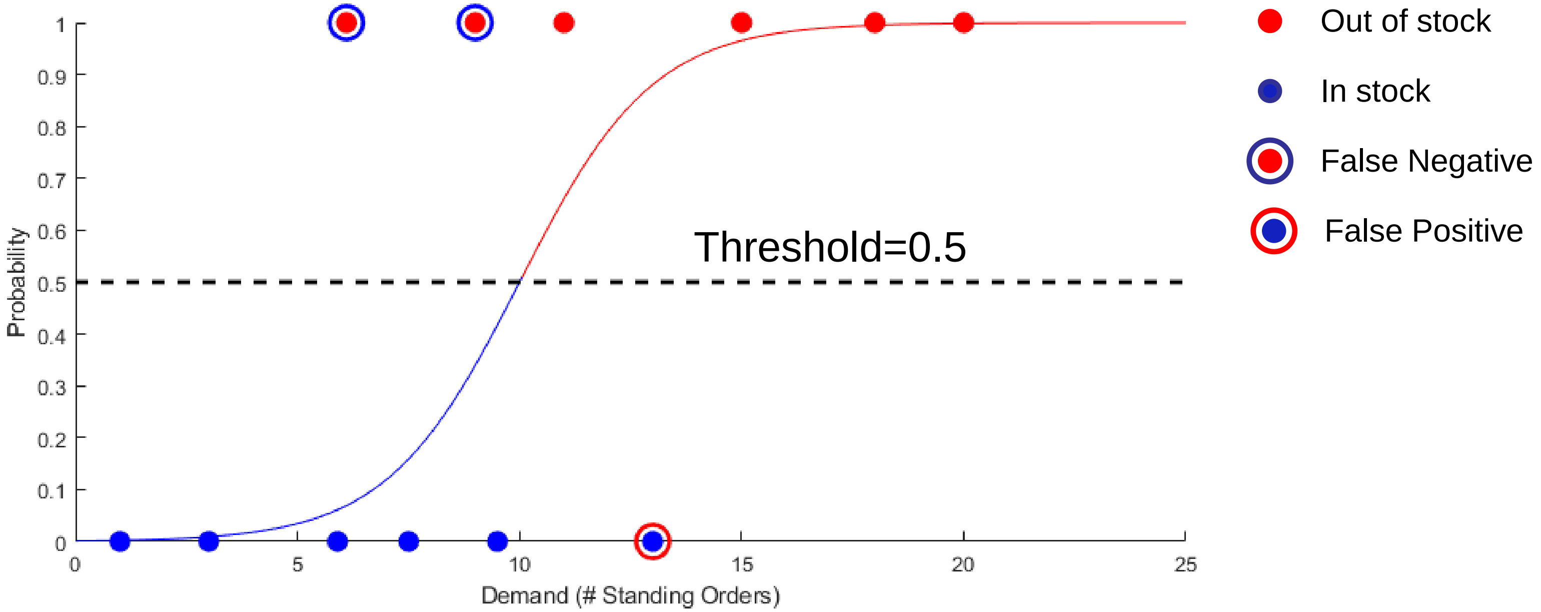


2. Ranking metrics: General approach

		Actual	
		Positive class	Negative class
Predicted	Positive class	4 (TP)	1 (FP)
	Negative class	2 (FN)	5 (TN)

TPR= Recall = TP/(TP+FN)=4/4+2 = 0,66

FPR = FP/(FP+TN)=1/(1+5) = 0,16



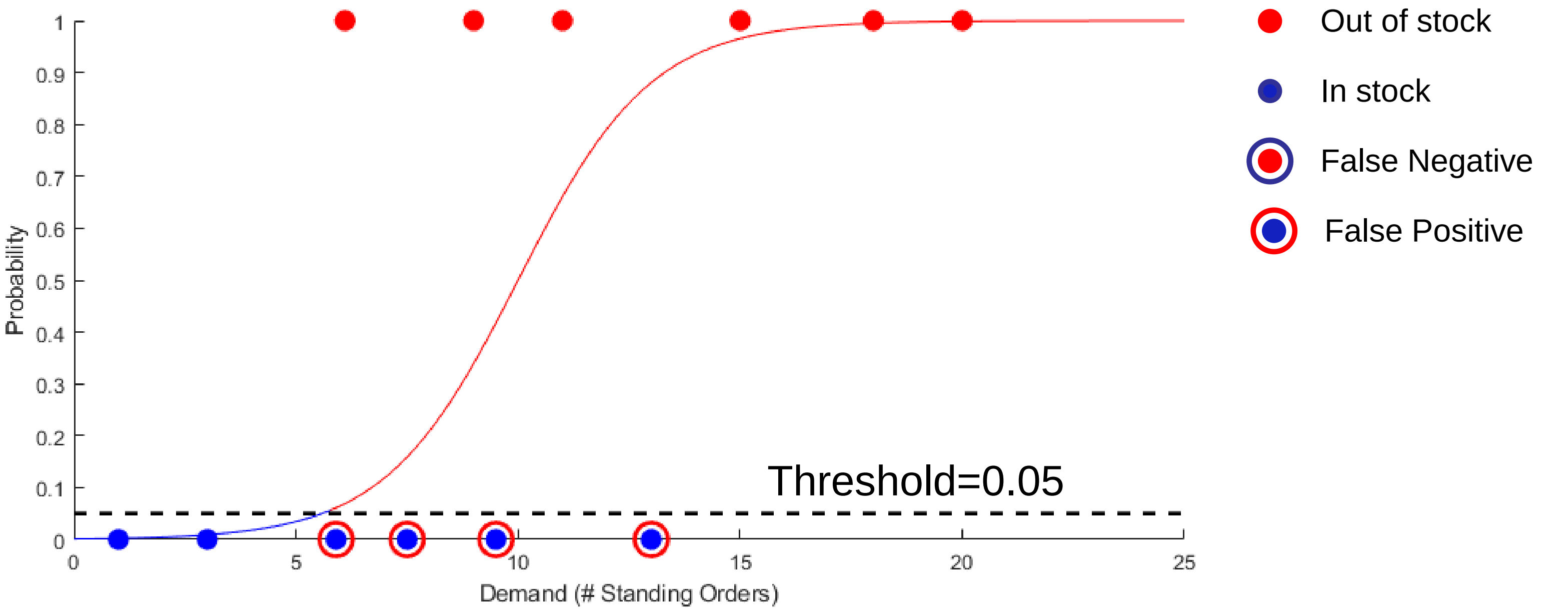
2. Ranking metrics: General approach

		Actual	
		Positive class	Negative class
Predicted	Positive class	6 (TP)	4(FP)
	Negative class	0 (FN)	2 (TN)

Precision = 6/(6+4)=0.6 TP/(TP+FP)

TPR= 6/6+0 = 1

FPR = 4/(4+2) = 0.66



2. Ranking metrics: General approach

Threshold=0

		Actual	
		Positive class	Negative class
Predicted	Positive class	6 (TP)	6(FP)
	Negative class	0 (FN)	0 (TN)

TPR= 6/(6+0) = 1

FPR = 6/(6+0) = 1

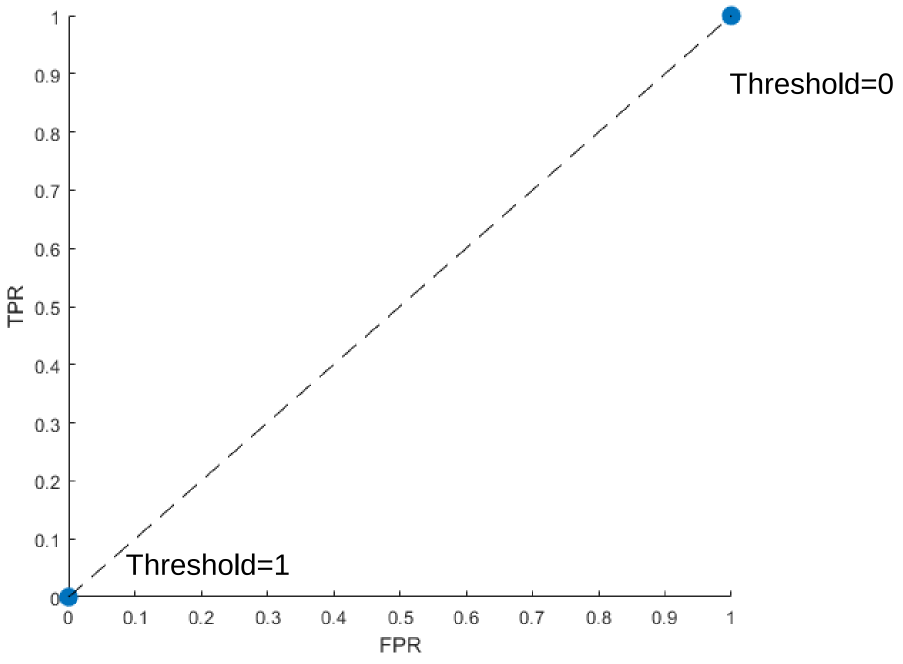
Threshold=1

		Actual	
		Positive class	Negative class
Predicted	Positive class	0 (TP)	0(FP)
	Negative class	6 (FN)	6 (TN)

TPR= 0/(0+6) = 0

FPR = 0/(0+6) = 0

ROC



2. Ranking metrics: General approach

Threshold=0

		Actual	
		Positive class	Negative class
Predicted	Positive class	6 (TP)	6(FP)
	Negative class	0 (FN)	0 (TN)

TPR= 6/(6+0) = 1

FPR = 6/(6+0) = 1

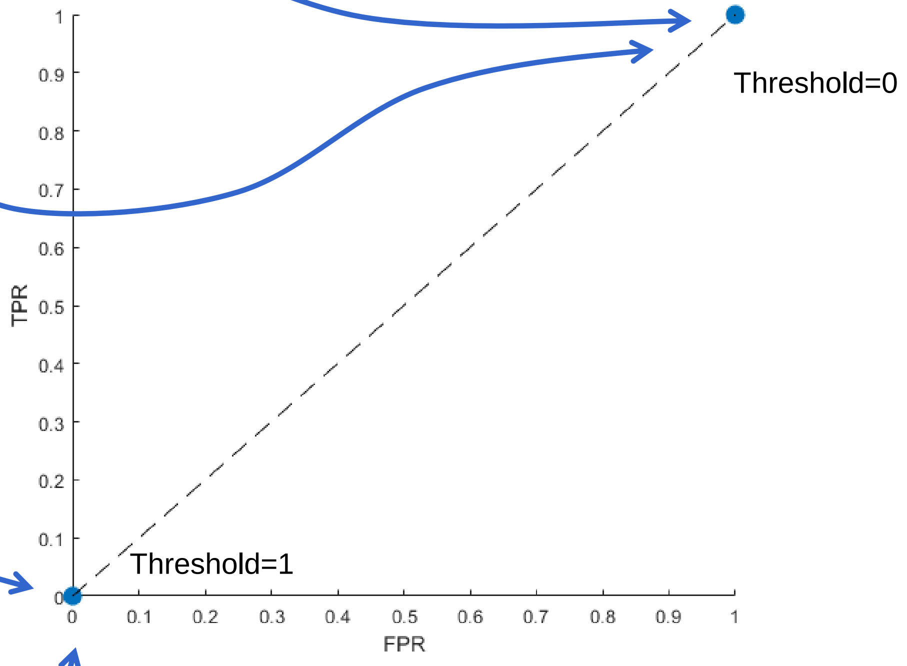
Threshold=1

		Actual	
		Positive class	Negative class
Predicted	Positive class	0 (TP)	0(FP)
	Negative class	6 (FN)	6 (TN)

TPR= 0/(0+6) = 0

FPR = 0/(0+6) = 0

ROC



2. Ranking metrics: General approach

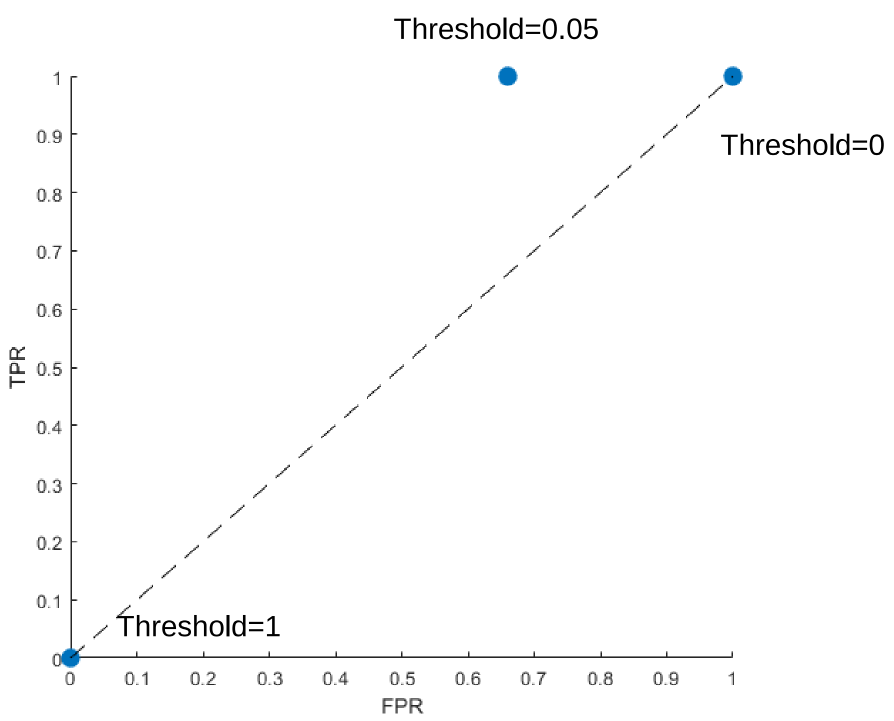
Threshold=0,05

		Actual	
		Positive class	Negative class
Predicted	Positive class	6 (TP)	4(FP)
	Negative class	0 (FN)	2 (TN)

TPR= 6/(6+0) = 1

FPR = 4/(4+2) = 0,66

ROC



2. Ranking metrics: General approach

Threshold=0,05

		Actual	
		Positive class	Negative class
Predicted	Positive class	6 (TP)	4(FP)
	Negative class	0 (FN)	2 (TN)

TPR= 6/(6+0) = 1

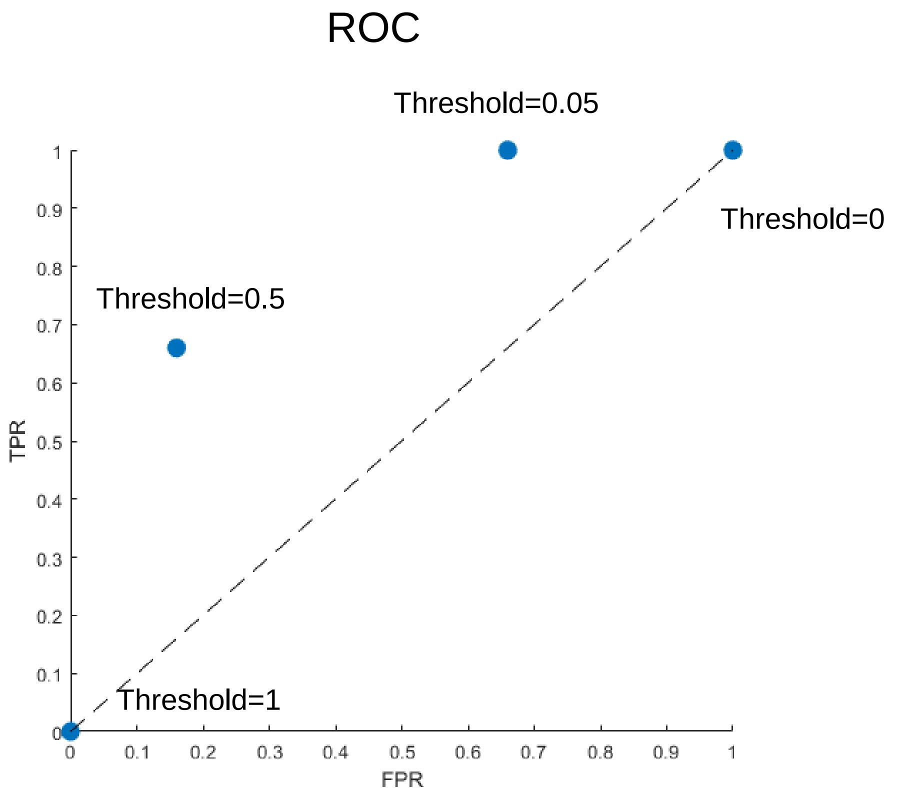
FPR = 4/(4+2) = 0,66

Threshold=0,5

		Actual	
		Positive class	Negative class
Predicted	Positive class	4 (TP)	1 (FP)
	Negative class	2 (FN)	5 (TN)

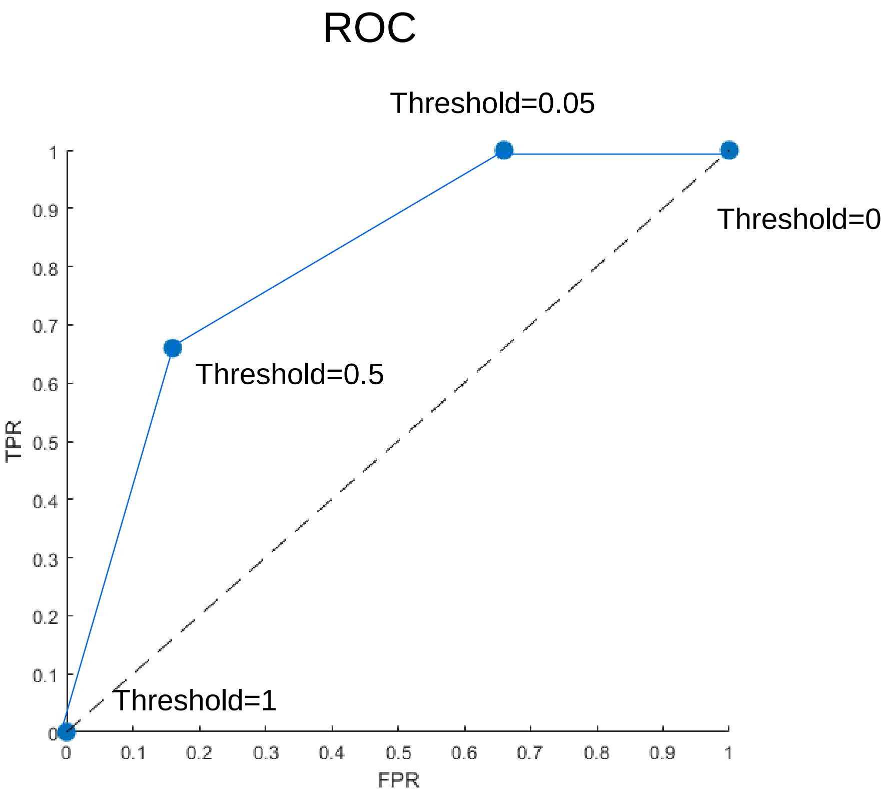
TPR= 6/(6+0) = 0,66

FPR = 4/(4+2) = 0,16



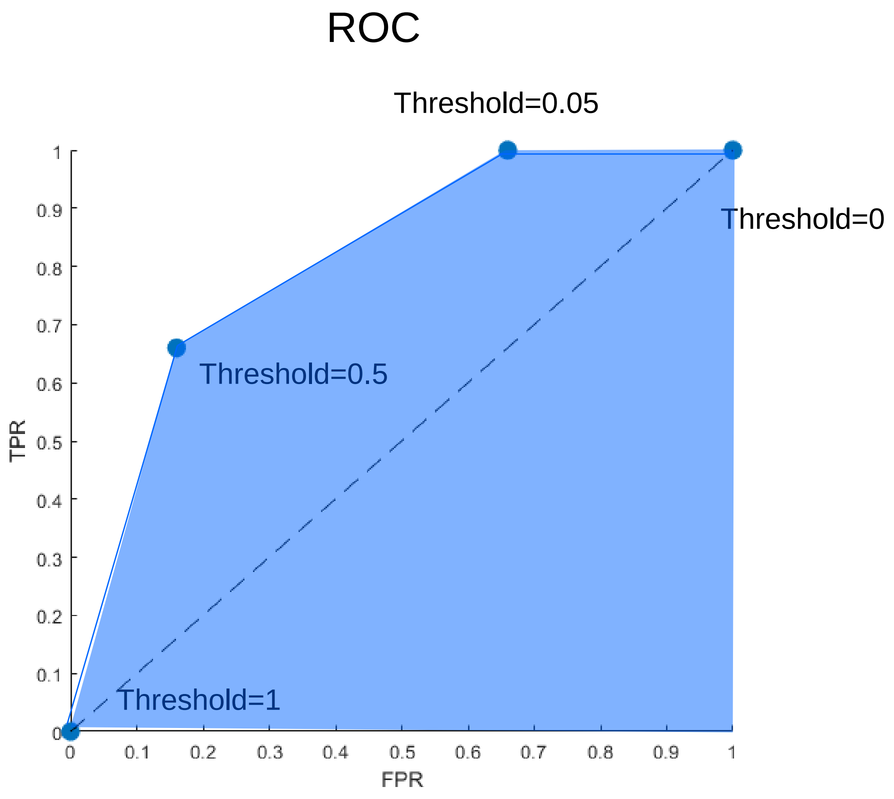
2. Ranking metrics: General approach

1. We obtain the ROC curve by connecting all the points in the plot (blue line)
2. Here we did it for 4 points (threshold = 0; 0,05; 0,5; 1) . However, we can do that for more.
3. The ROC summarizes all of the confusion matrices obtained with different threshold
4. We know that the best model should have:
   1. TPR = 1 (All positive correctly classified)
   2. FPR = 0 (All negative correctly classified)
5. In practice, you can decide how many false positives are you willing to accept (FPR) and then select the model with the higher TPR and lower FPR.



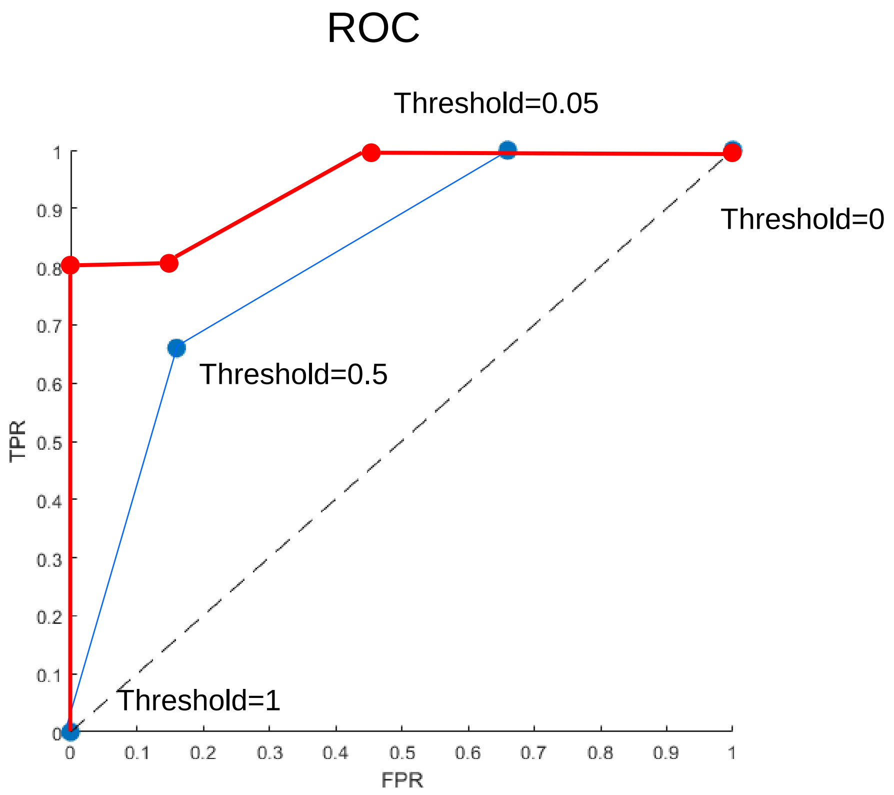
2. Ranking metrics: General approach

1. The AUC is simply the area under the curve (ROC)
2. It is useful to associate a score to a model:
   1. AUC BlueModel = 0,8



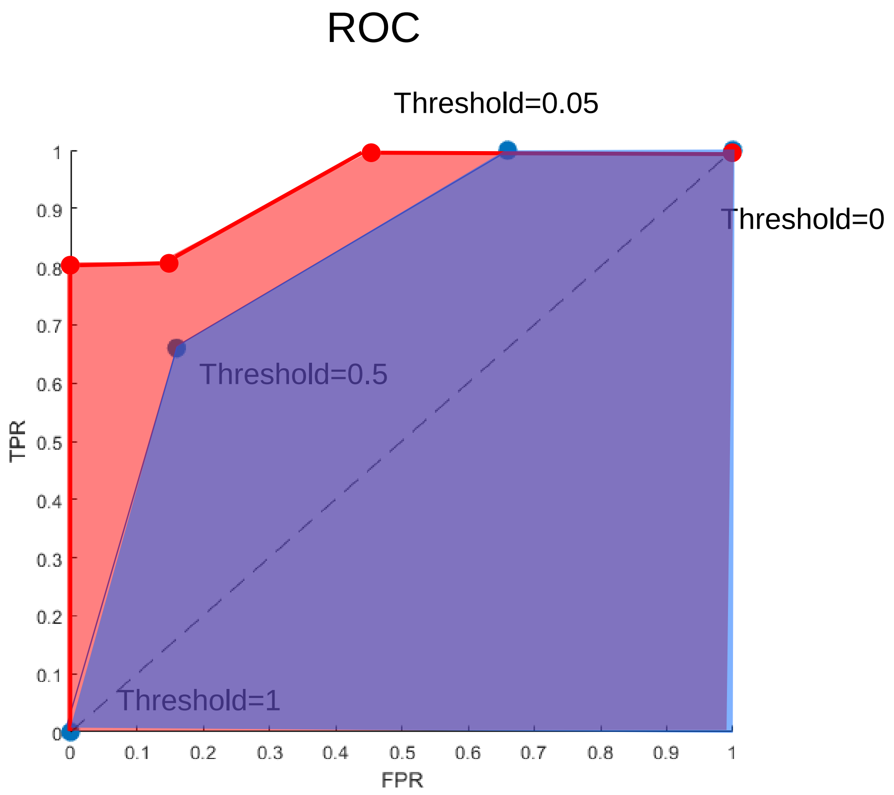
2. Ranking metrics: General approach

1. The AUC is simply the area under the curve (ROC)
2. It is useful to associate a score to a model:
    1. AUC BlueModel = 0,8
3. This can be useful to compare models. For example, which model between the red and the blue is better?



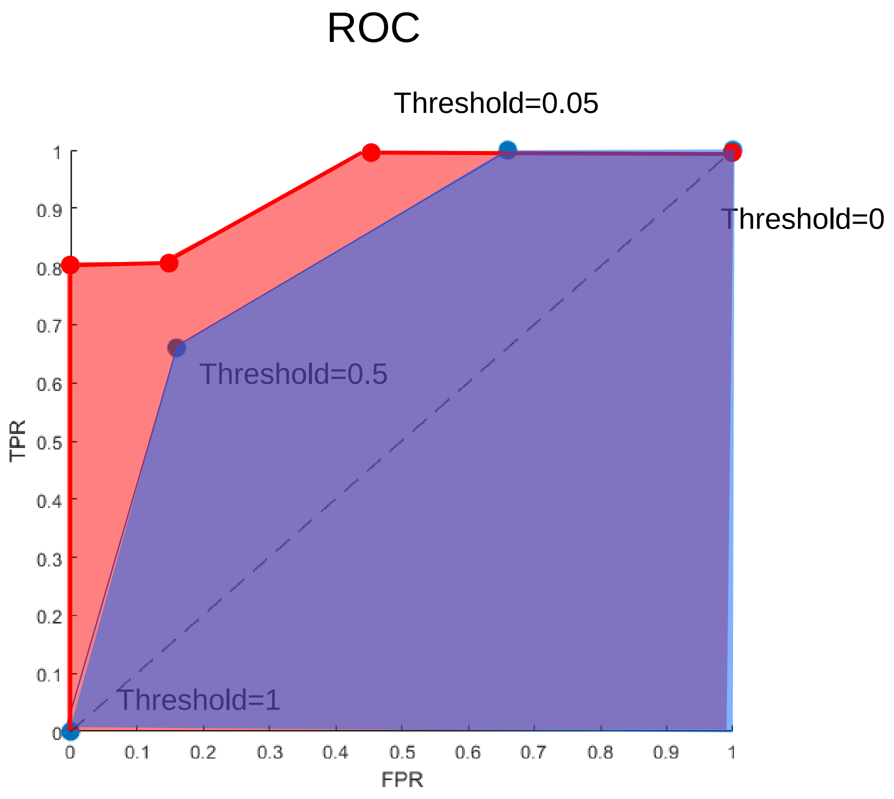
2. Ranking metrics: General approach

1. The AUC is simply the area under the curve (ROC)
2. It is useful to associate a score to a model:
    1. AUC BlueModel = 0,8
3. This can be useful to compare models. For example, which model between the red and the blue is better?
4. Looking at the curve, the red model is better (higher TPR for the same FPR). The AUC compress this information into one number
    1. AUC RedModel = 0,93

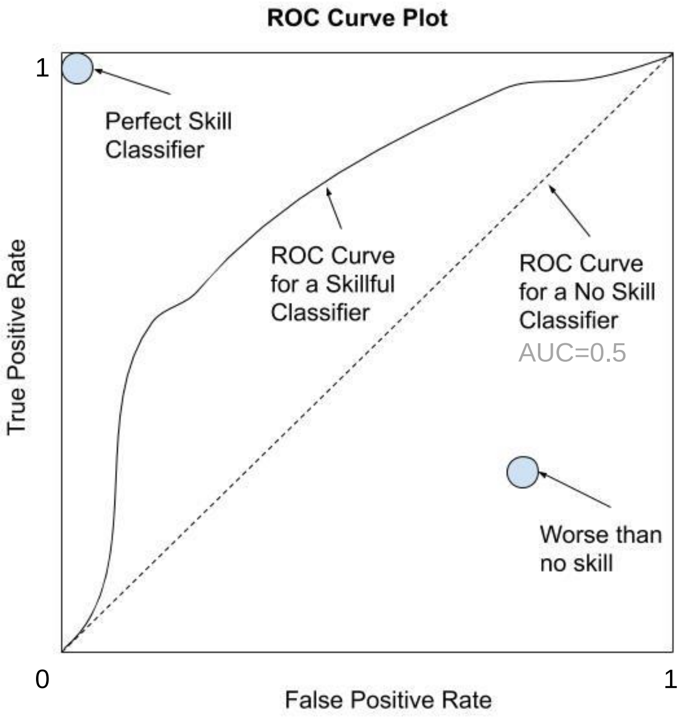


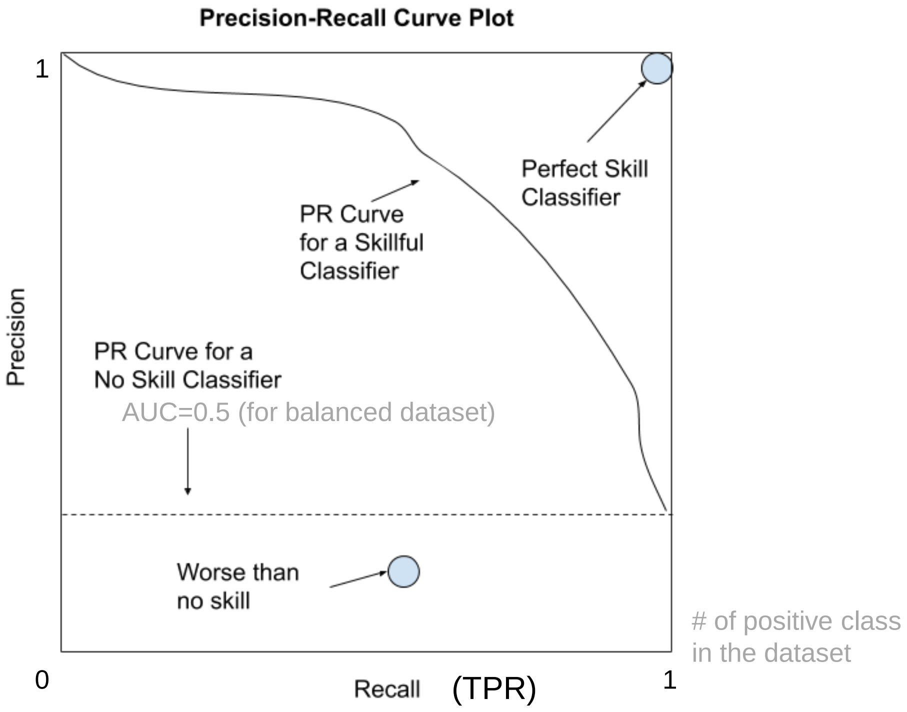
2. Ranking metrics: General approach

1. The AUC is simply the area under the curve (ROC)
2. It is useful to associate a score to a model:
    1. AUC BlueModel = 0,8
3. This can be useful to compare models. For example, which model between the red and the blue is better?
4. Looking at the curve, the red model is better (higher TPR for the same FPR). The AUC compress this information into one number
    1. AUC RedModel = 0,93
5. The ROC tells you what is the best threshold (i.e., 0,05 is better than 0 for the blue model)
6. The AUC tells you which model is the best (The blue model is better than the blue one)



2. Ranking metrics: Plots





Treats two classes equally

Focused more on the positive class

https://www.sciencedirect.com/science/article/abs/pii/S0167865508002687

https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

2. Ranking metrics: General approach

1. Predict scores using a model

2. Change decision threshold from 0 to max

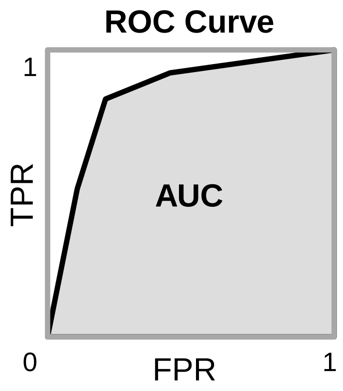
   1. For each decision threshold, calculate confusion matrix
      - **For ROC, calculate**
        TPR (True Positive Rate) = Recall = TP/(TP+FN)
        FPR (False Positive Rate) = 1- Specificity = FP/(TN+FP)

      - **For PR, calculate**
        Precision = TP/(TP+FP)
        Recall = TP/(TP+FN)
   2. Plot one versus another

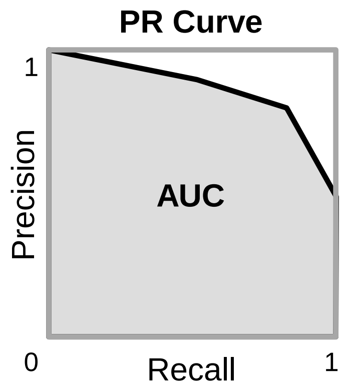
3. To get one number, calculate Area Under the Curve (AUC)

	True 1	**True 0**
Pred 1	**TP**	**FP**
Pred 0	**FN**	**TN**

ROC Curve



PR Curve



https://www.sciencedirect.com/science/article/abs/pii/S0167865508002687

https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

3. Probability metrics

- Metrics based on a probabilistic understanding of error, i.e. **measuring the deviation from the true probability**.

- *Business value = f(class probabilities)*

- **Business cases:** Assign dangerous treatment to a patient when we really sure about it, …

- **Most popular metrics:**
  - Binary loss/cross-entropy = log-likelihood of a categorical variable
  - Brier score

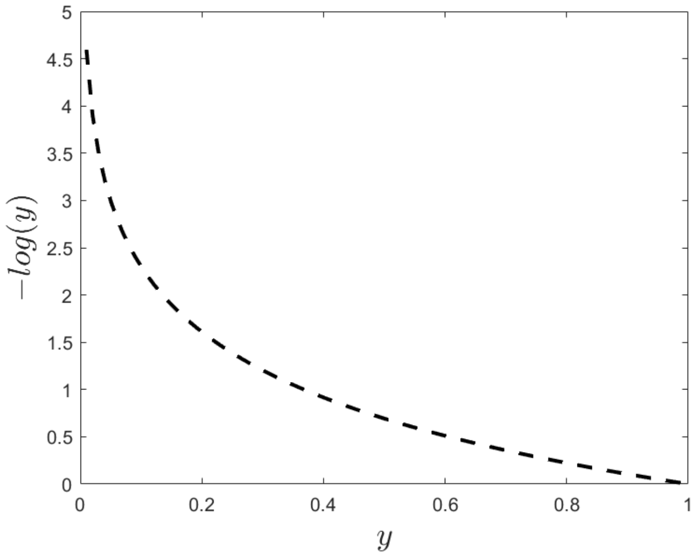
https://www.sciencedirect.com/science/article/abs/pii/S0167865508002687

https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

3. Probability metrics

- **Binary loss/cross-entropy** = log-likelihood of a categorical variable (

$$\mathrm{LL}_i = -\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{y}_k^{(i)}\right)$$



- **Brier score**

$$\mathrm{BS}_i = \sum_{k=1}^{K} \left(y_k^{(i)} - \hat{y}_k^{(i)}\right)^2$$

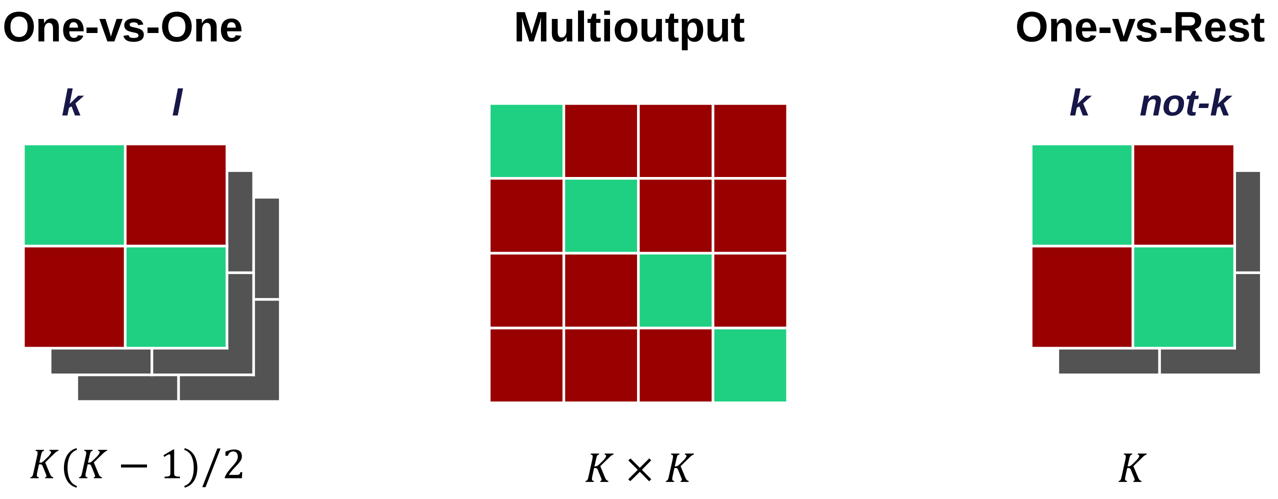
- **Comments:**
  - In principle, both can be weighted for each class, e.g.

$$-\sum_{k=1}^{K} w_k y_k^{(i)} \log\left(\hat{y}_k^{(i)}\right)$$

$\hat{y}_k$= Estimated y_k= Observerved

Multiclass classification (*K* classes)

- **Multiouput**: Single model f predicts **score for each class** $s_k^{(i)} = f\left(x^{(i)}\right)$ measuring model's belief that the sample $x^{(i)}$ belongs to the class k



One-vs-One

k *l*

$K(K-1)/2$

Multioutput

$K \times K$

One-vs-Rest

k *not-k*

K

Multiclass classification performance metrics

- **Common way:** K one-vs-rest confusion matrices, CM_k, are considered
  - **Macro-average:** Calculate the F-score for each CM_k, then average the scores
  - **Micro-average:** Calculate the total confusion matrix $\mathrm{CM} = \sum_{k=1}^{K} \mathrm{CM}_k$, then calculate the F-score
    - More frequent classes will dominate ~ weighted macro-average

Playtime

- Open the "4. DecisionTree.ipynb" notebook
- Do Part 3

Relevant references

- The text in Content -> ‘Textbooks and References’
  - An Introduction to Statistical Learning: with Applications in Python
    - Algorithms, codes, examples for decision trees, bagging, boosting, and random forest
    - Introduction to the ROC curve (under ‘Generative Models’)
  - Data Science from Scratch
    - Detailed and intuitive introduction to Decision Trees
    - Very basic discussion about Random Foresr
    - Performance metrics: threshold metrics are discussed, but the ROC and Precion-recall curve is not described.

- StatQuest with Josh Starmer
  - More friendly approach, also videos
  - Very clear examples (some of them, inspired the ones in this lecture)

https://medium.com/comet-ml/a-data-scientists-guide-to-communicating-results-c79a5ef3e9f1