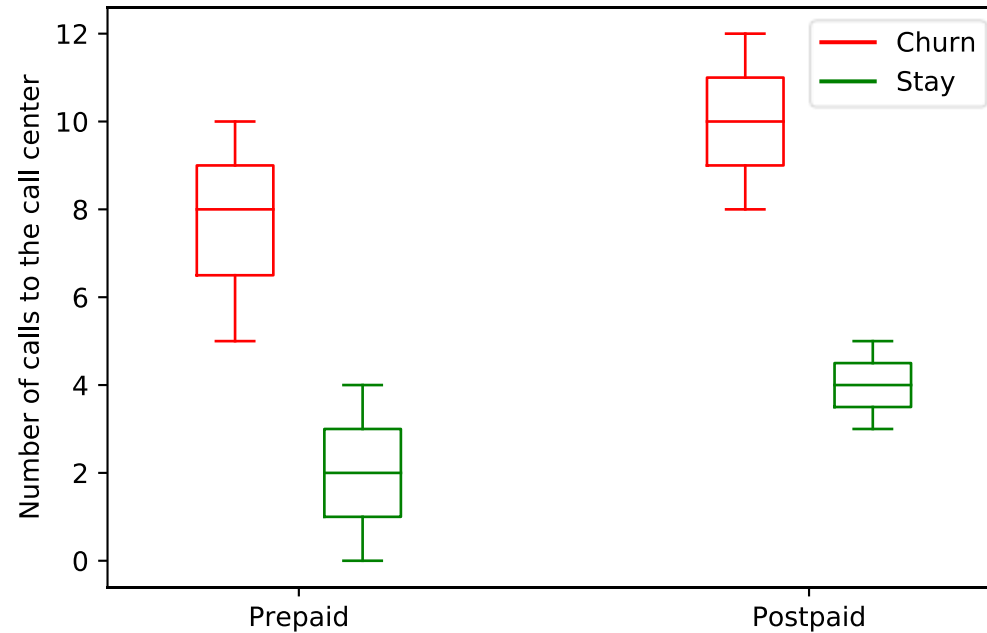




# DECISION TREE AND RANDOM FOREST

Ratchainant Thammasudjarit, Ph.D.

- Visualization



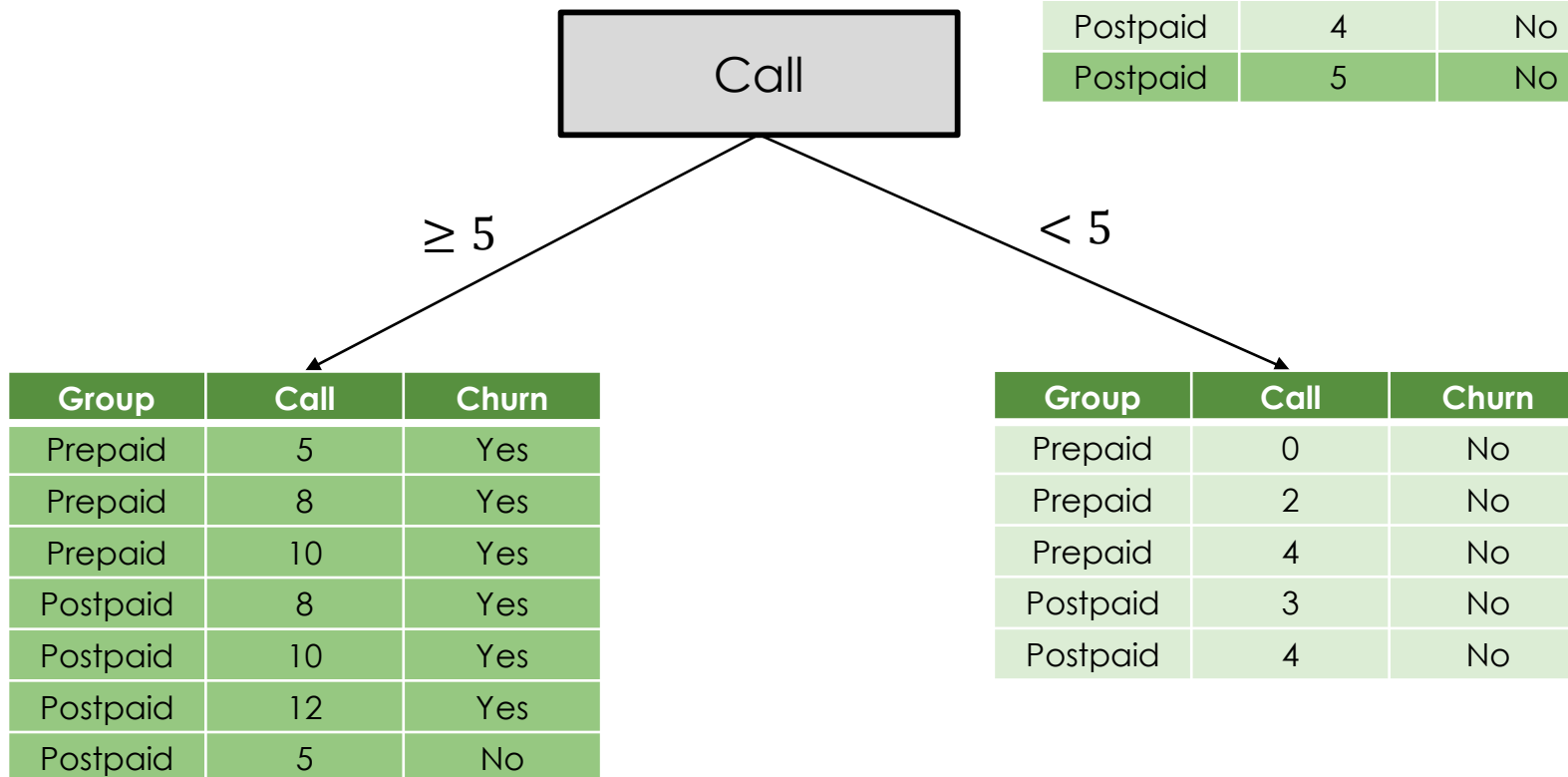
## Decision Tree

Given data

Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes
Prepaid	0	No
Prepaid	2	No
Prepaid	4	No
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	3	No
Postpaid	4	No
Postpaid	5	No

- Concepts

Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes
Prepaid	0	No
Prepaid	2	No
Prepaid	4	No
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	3	No
Postpaid	4	No
Postpaid	5	No

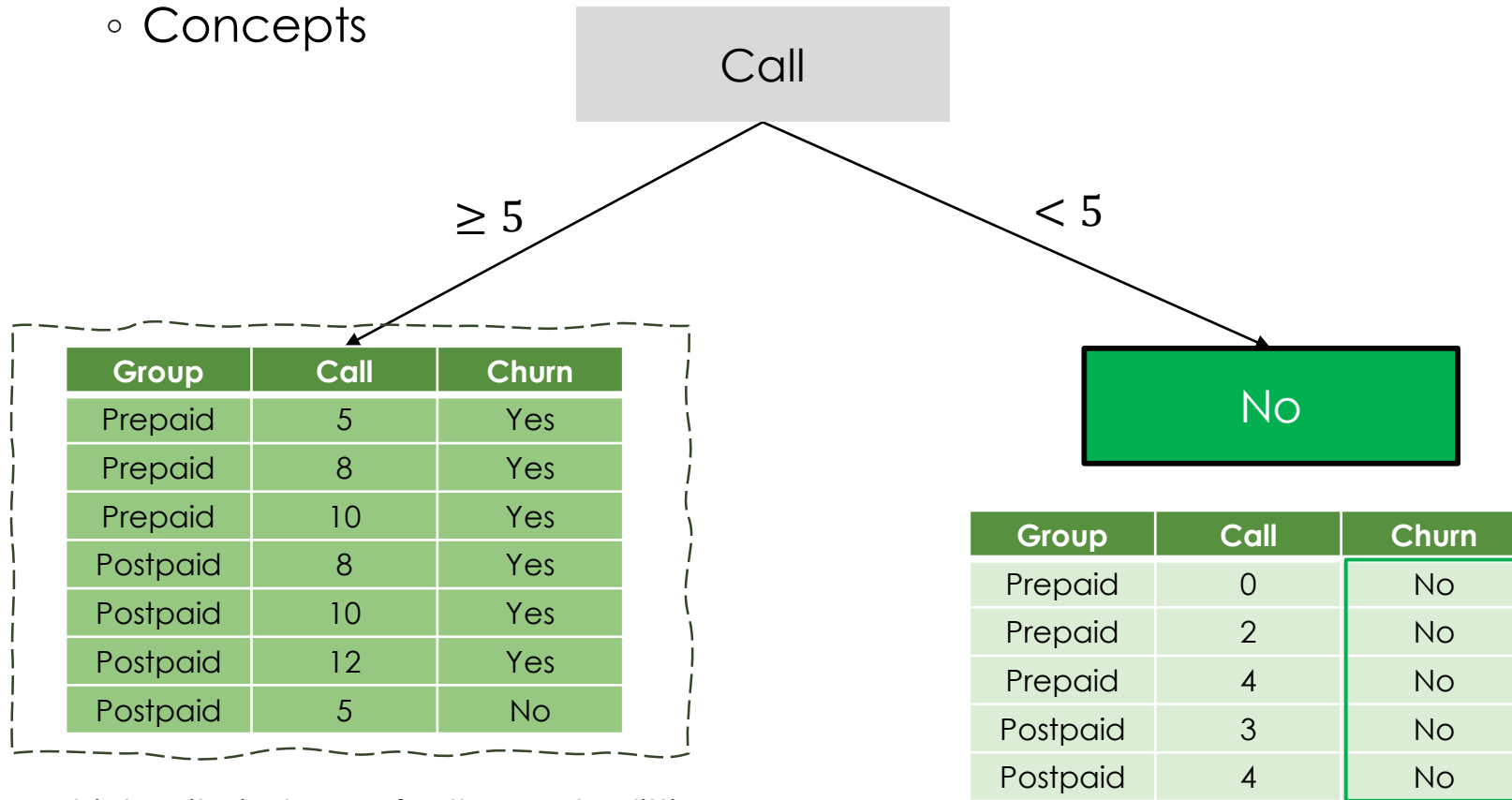


## Decision Tree

The decision tree makes data partitioning until no more data to be partitioned

Note: In this example, **call** refers to the number of calls to the call center

- Concepts

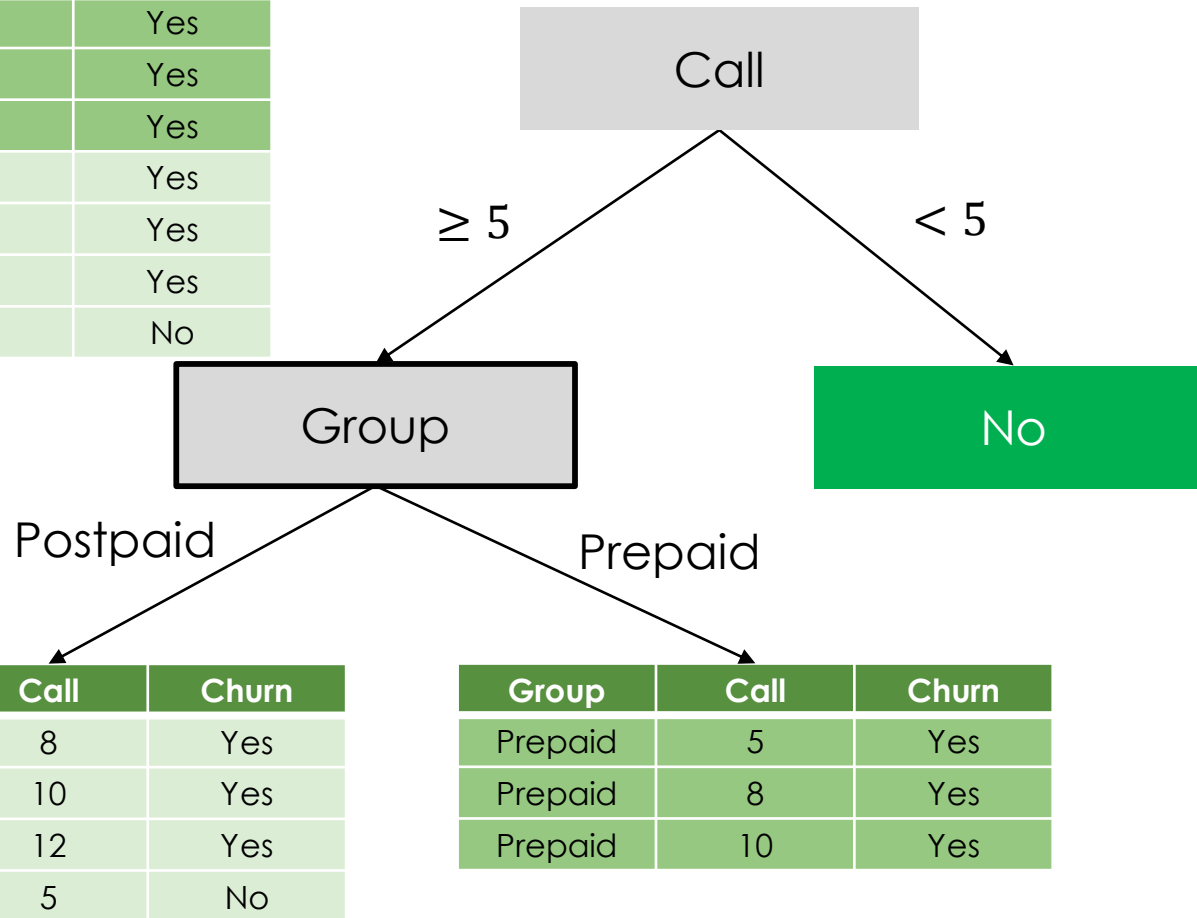


Which criteria to use for the next splitting

## Decision Tree

The decision tree makes data partitioning until no more data to be partitioned

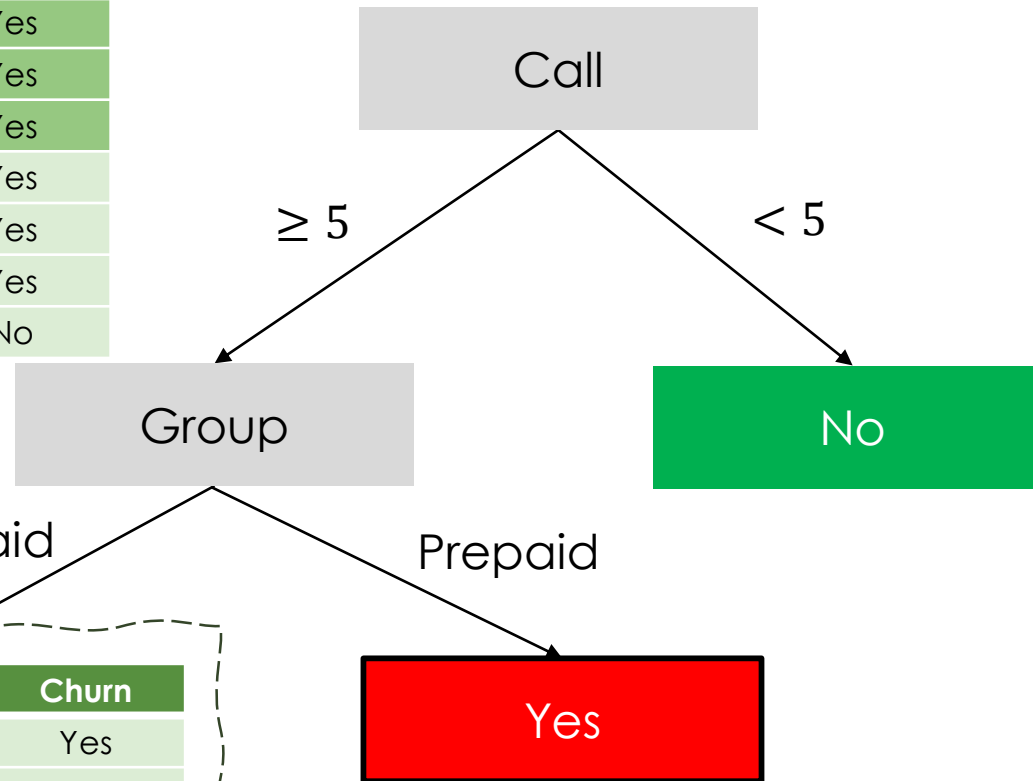
Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	5	No



# Decision Tree

The decision tree makes data partitioning until no more data to be partitioned

Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	5	No



Group	Call	Churn
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	5	No

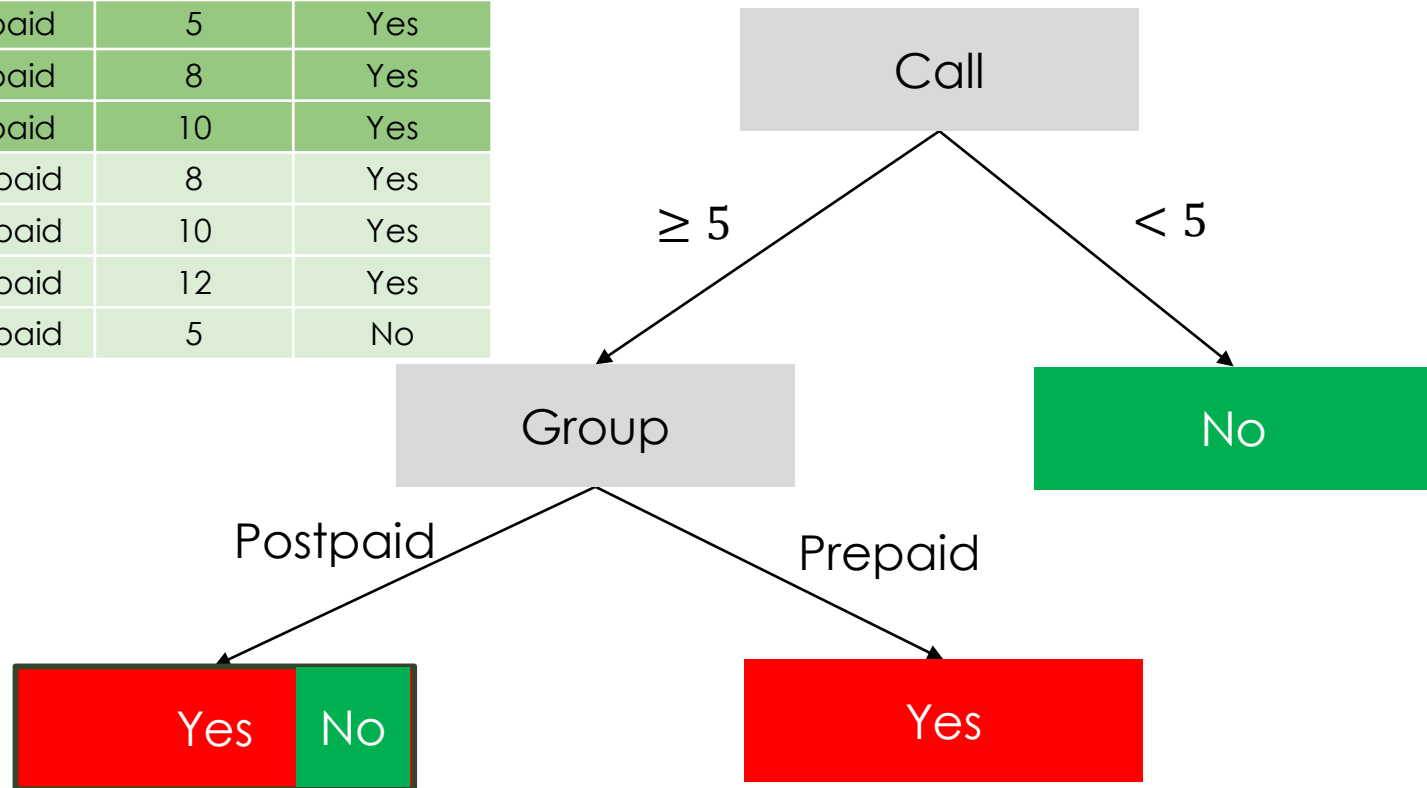
Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes

Which criteria to use for the next splitting

## Decision Tree

The decision tree makes data partitioning until no more data to be partitioned

Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	5	No



Group	Call	Churn
Postpaid	8	Yes
Postpaid	10	Yes
Postpaid	12	Yes
Postpaid	5	No

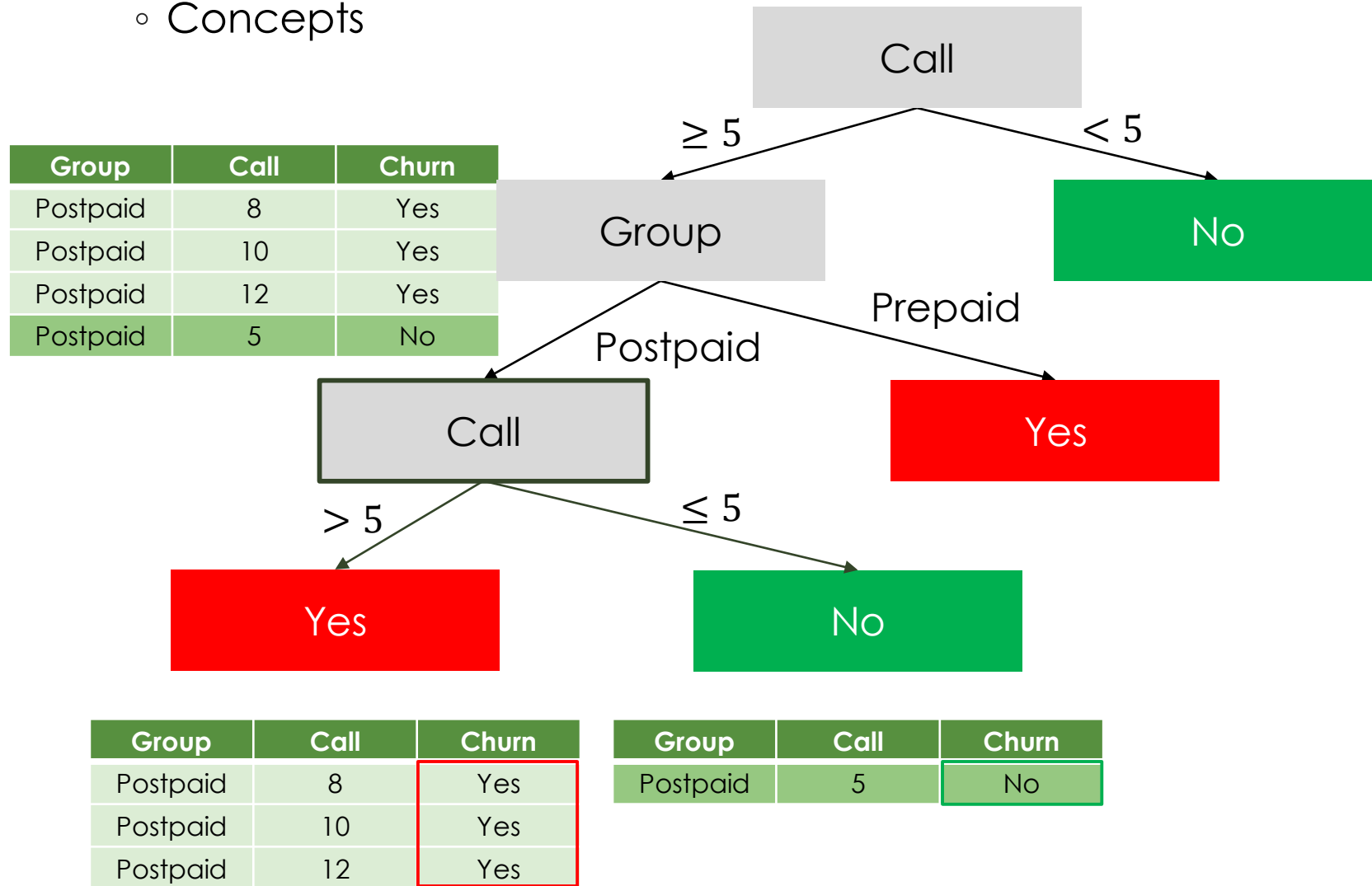
Group	Call	Churn
Prepaid	5	Yes
Prepaid	8	Yes
Prepaid	10	Yes

## Decision Tree

The decision tree makes data partitioning until no more data to be partitioned

If the maximum depth is defined as 2, the decision tree will stop learning at this stage

- Concepts



Otherwise, the decision tree will keep partitioning data until the end

## Decision Tree

The decision tree makes data partitioning until no more data to be partitioned



- Impurity measures for classification task
  - Entropy
  - Gini
  - Classification error
- Impurity measure for regression task
  - Variance

# Impurity Measure

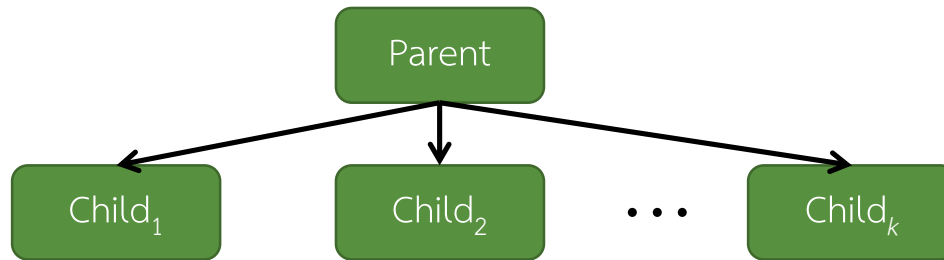
Impurity measure is applied to make decision which criteria to use for splitting

- Entropy

$$H(x) = \sum_j -P(j) \log_2 P(j)$$

where  $j$  is any possible value in a given feature  
 $P(j)$  is the probability of  $j$

- Splitting



$$IG(Parent, Children) = H(Parent) - \sum_i \frac{N_{child_k}}{N_{parent}} \cdot H(Child_i)$$

Where  $H(\cdot)$  is entropy of a particular node

$N_{parent}$  is the number of datapoint of the parent node

$N_{child_k}$  is the number of datapoint of the  $k^{th}$  child node

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting

- Entropy (Example): Measure Entropy of Call

Call	Churn
5	Yes
8	Yes
10	Yes
0	No
2	No
4	No
8	Yes
10	Yes
12	Yes
3	No
4	No
5	No

Call

$$H(\text{parent}) = -\frac{6}{12} \log_2 \frac{6}{12} - \frac{6}{12} \log_2 \frac{6}{12} = 1$$

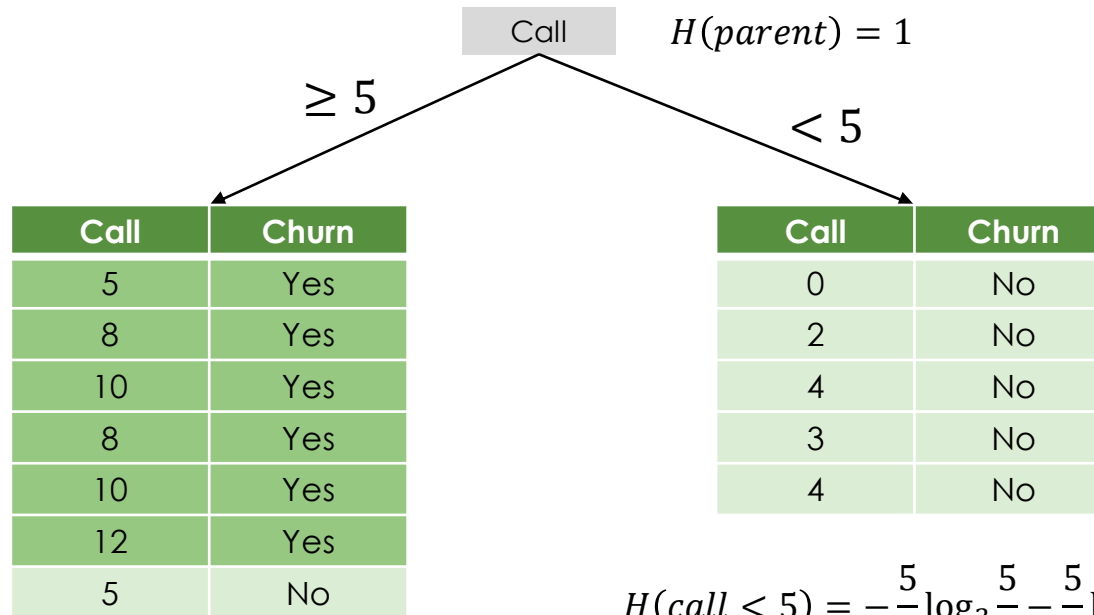
*P(churn = Yes)*

*P(churn = No)*

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting

- Entropy (Example): Measure Entropy of Call



$$H(\text{call} < 5) = -\frac{5}{5} \log_2 \frac{5}{5} - \frac{5}{5} \log_2 \frac{5}{5} = 0$$

$$H(\text{call} \geq 5) = -\frac{6}{7} \log_2 \frac{6}{7} - \frac{1}{7} \log_2 \frac{1}{7} = 0.59$$

$$\begin{aligned}
 IG(\text{parent}, \text{children}) &= H(\text{parent}) - \frac{N_{\text{call} \geq 5}}{N_{\text{parent}}} \cdot H(\text{call} \geq 5) - \frac{N_{\text{call} < 5}}{N_{\text{parent}}} \cdot H(\text{call} < 5) \\
 &= 1 - \frac{7}{12} (0.59) - \frac{5}{12} (0) \\
 &= 0.65
 \end{aligned}$$

## Impurity Measure

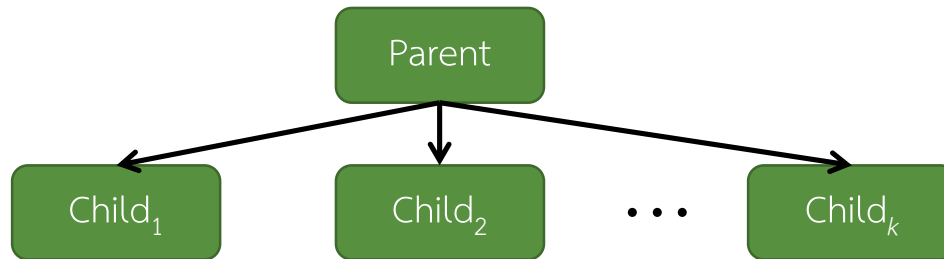
Impurity measure is applied to make decision which criteria to use for splitting

- Gini

$$G(x) = 1 - \sum_j P(j)^2$$

where  $j$  is any possible value in a given feature  
 $P(j)$  is the probability of  $j$

- Splitting



$$Gini(Parent, Children) = G(Parent) - \sum_i \frac{N_{Child_k}}{N_{parent}} \cdot G(Child_i)$$

Where  $G(\cdot)$  is Gini of a particular node

$N_{parent}$  is the number of datapoint of the parent node

$N_{child_k}$  is the number of datapoint of the  $k^{th}$  child node

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting

- Gini (Example): Measure Gini of Call

Call	Churn
5	Yes
8	Yes
10	Yes
0	No
2	No
4	No
8	Yes
10	Yes
12	Yes
3	No
4	No
5	No

Call

$$G(\text{parent}) = 1 - \left(\frac{6}{12}\right)^2 - \left(\frac{6}{12}\right)^2 = 0.5$$

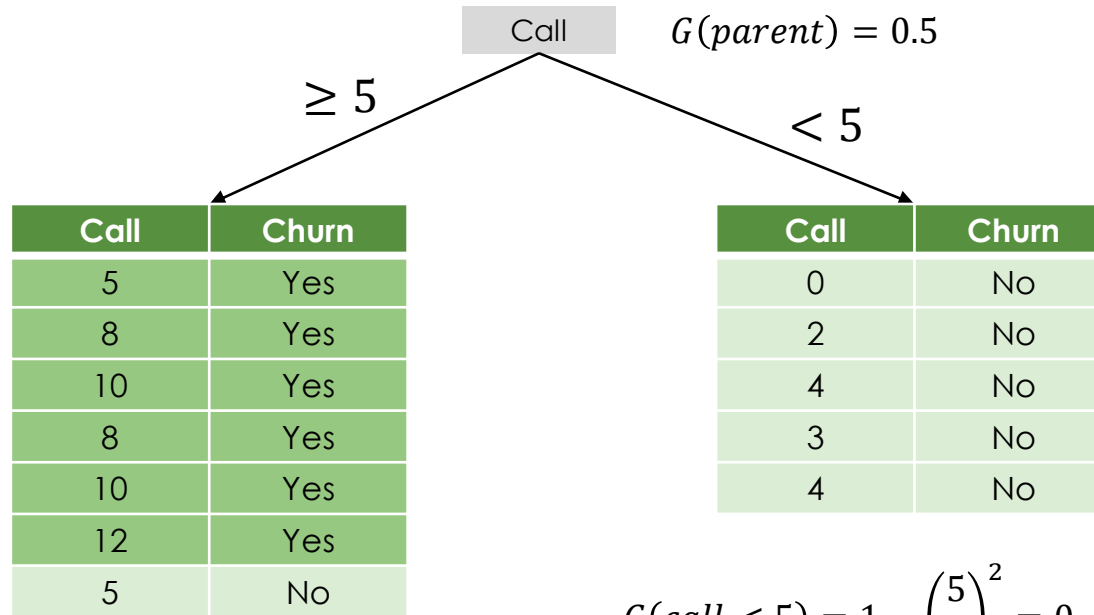
*P(churn = Yes)*

*P(churn = No)*

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting

- Gini (Example): Measure Gini of Call



$$G(\text{call} \geq 5) = 1 - \left(\frac{1}{7}\right)^2 - \left(\frac{6}{7}\right)^2 = 0.24$$

$$G(\text{call} < 5) = 1 - \left(\frac{5}{5}\right)^2 = 0$$

$$\begin{aligned}
 G(\text{parent}, \text{children}) &= G(\text{parent}) - \frac{N_{\text{call} \geq 5}}{N_{\text{parent}}} \cdot G(\text{call} \geq 5) - \frac{N_{\text{call} < 5}}{N_{\text{parent}}} \cdot G(\text{call} < 5) \\
 &= 0.5 - \frac{7}{12} (0.24) - \frac{5}{12} (0) \\
 &= 0.36
 \end{aligned}$$

## Impurity Measure

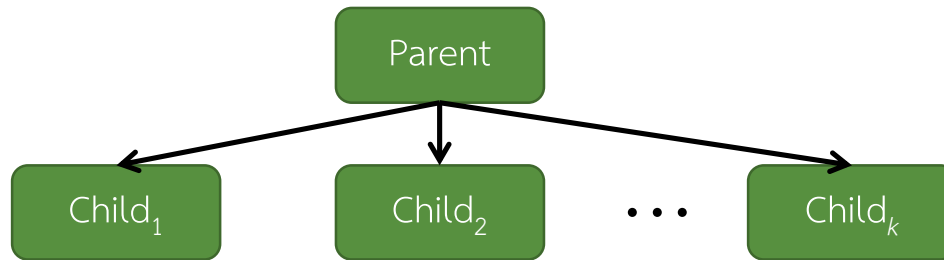
Impurity measure is applied to make decision which criteria to use for splitting

- Classification Error

$$E(x) = 1 - \max\{P(j)\}$$

where  $j$  is any possible value in a given feature  
 $P(j)$  is the probability of  $j$

- Splitting



$$IG(Parent, Children) = E(Parent) - \sum_i \frac{N_{child_k}}{N_{parent}} \cdot E(Child_i)$$

Where  $E(\cdot)$  is classification error of a particular node

$N_{parent}$  is the number of datapoint of the parent node

$N_{child_k}$  is the number of datapoint of the  $k^{th}$  child node

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting



- Classification Error (Example): Measure Classification error of Call

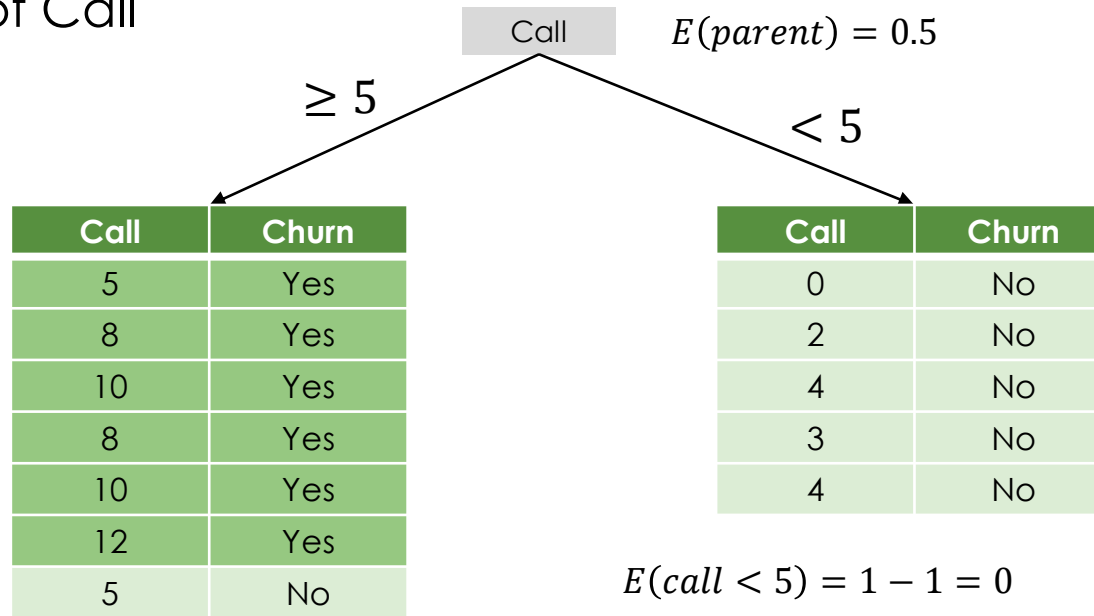
Call	Churn
5	Yes
8	Yes
10	Yes
0	No
2	No
4	No
8	Yes
10	Yes
12	Yes
3	No
4	No
5	No

Call  $E(\text{parent}) = 1 - \frac{6}{12} = 0.5$

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting

- Classification Error (Example): Measure Classification error of Call



$$E(\text{call} \geq 5) = 1 - \frac{6}{7} = 0.14$$

$$\begin{aligned}
 E(\text{parent}, \text{children}) &= E(\text{parent}) - \frac{N_{\text{call} \geq 5}}{N_{\text{parent}}} \cdot E(\text{call} \geq 5) - \frac{N_{\text{call} < 5}}{N_{\text{parent}}} \cdot E(\text{call} < 5) \\
 &= 0.5 - \frac{7}{12} (0.14) - \frac{5}{12} (0) \\
 &= 0.41
 \end{aligned}$$

## Impurity Measure

Impurity measure is applied to make decision which criteria to use for splitting

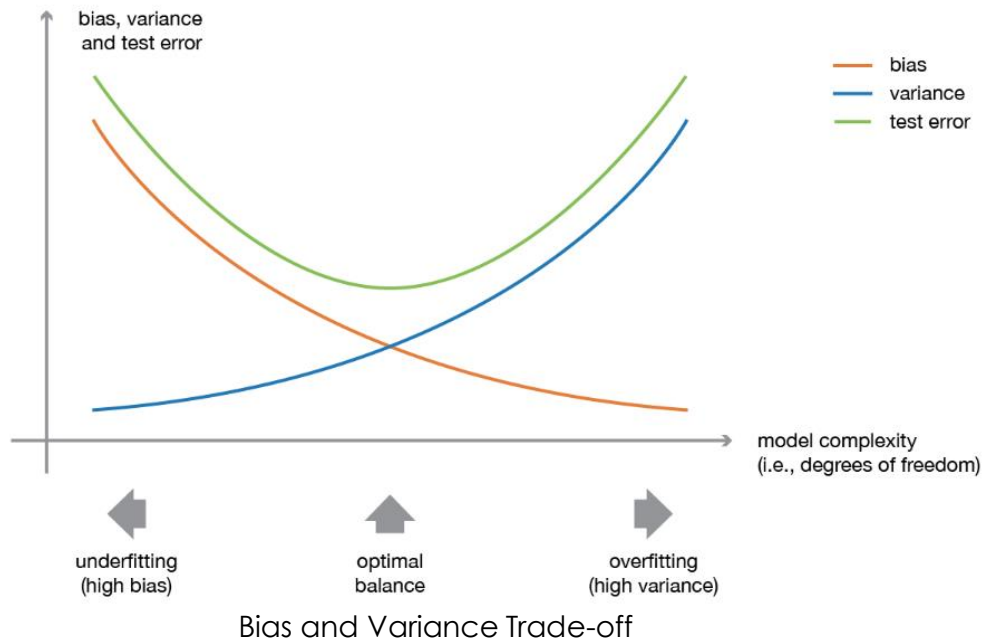


# Random Forest

Ensemble Technique

- Theory

- **Weak learners** (or **base models**) models can be used as building blocks for designing more complex models by combining several of them
- Basic Idea:
  - Trying reducing bias and/or variance of such weak learners by combining several of them together
  - Such combination creates a **strong learner** (or **ensemble model**) that achieves better performances



# Ensemble Learning

Combining multiple weak models is outperform a single strong model

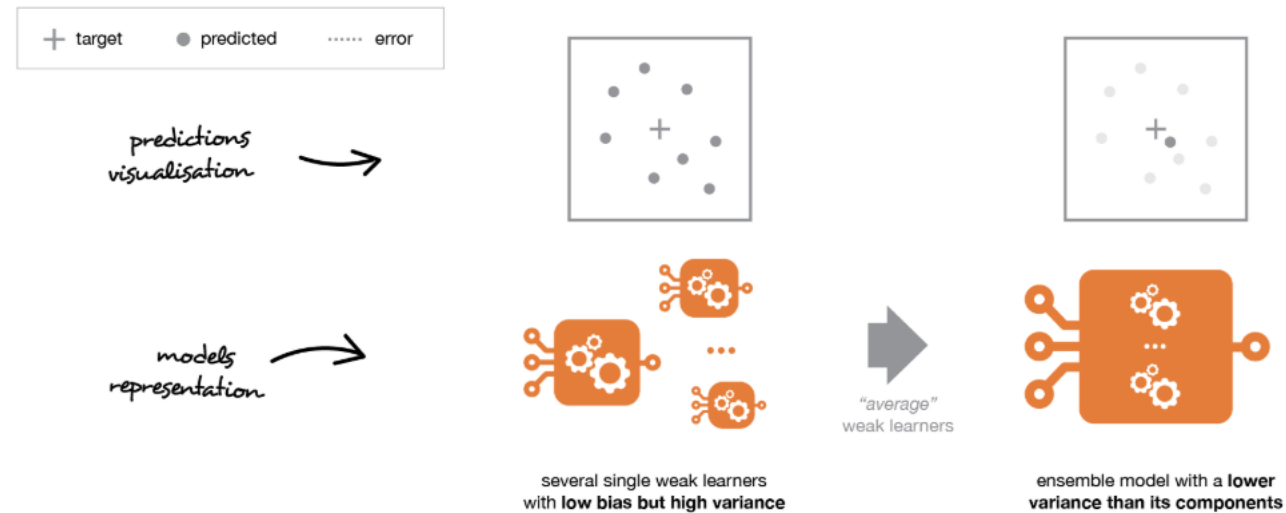
- Bagging
  - Homogeneous weak learners
  - Learn independently in parallel then combine output using averaging method
- Boosting
  - Homogeneous weak learners
  - Learn sequentially in adaptive way then combine output using specific strategy
- Stacking
  - Heterogeneous weak learners
  - Learn independently in parallel then train the meta-model from the output of weak learners

# Ensemble Learning

Three ensemble concepts

- Bagging
- Boosting
- Stacking

- Concepts

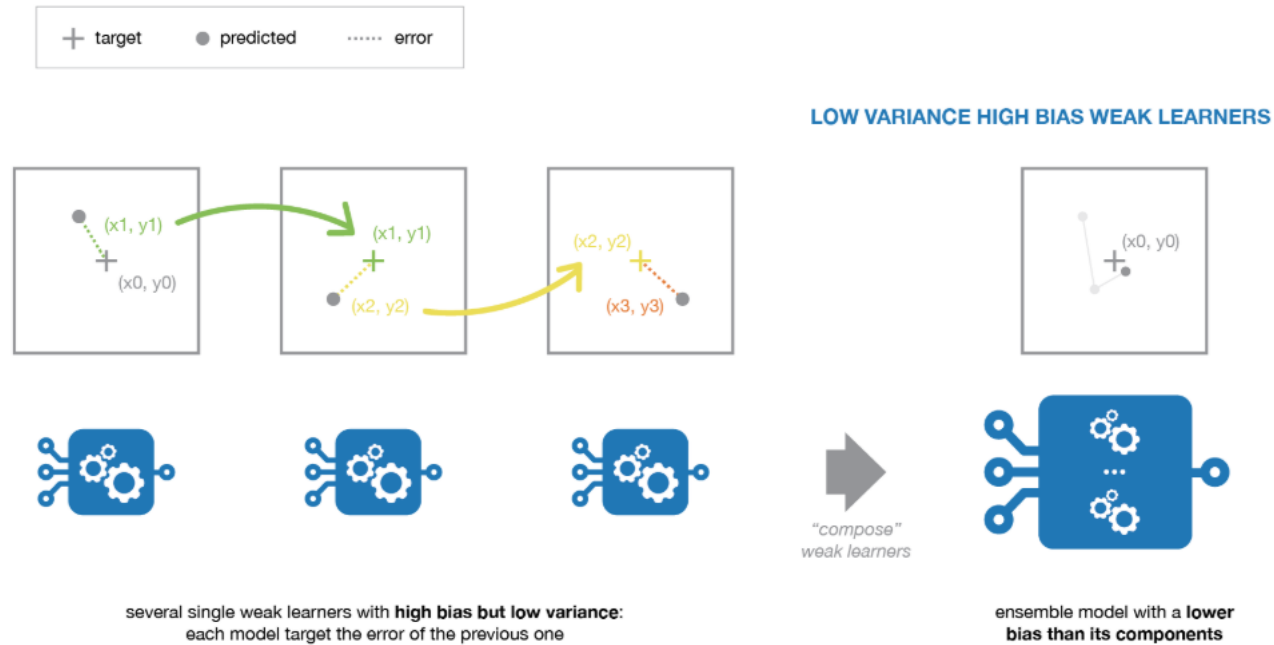


LOW BIAS HIGH VARIANCE WEAK LEARNERS

# Ensemble Learning

Bagging employs **homogeneous weak learners** to learn **independently in parallel** then **combine output using averaging method**

- Concepts

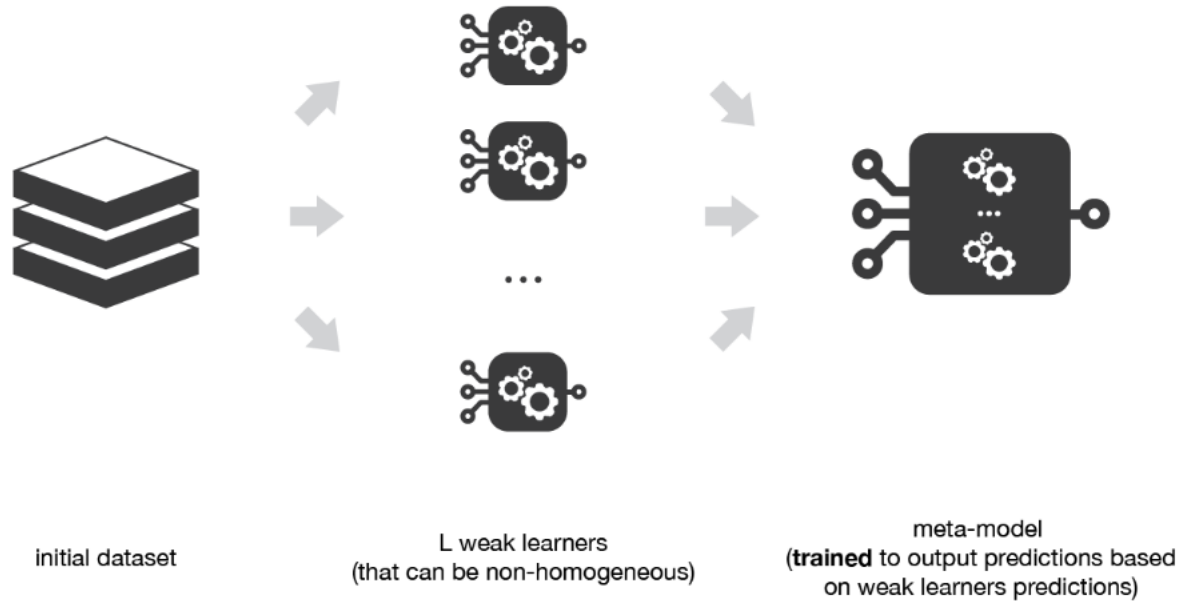


# Ensemble Learning

Boosting employs **homogeneous weak learners** to **learn sequentially** then **combine output using specific strategy**



- Concepts

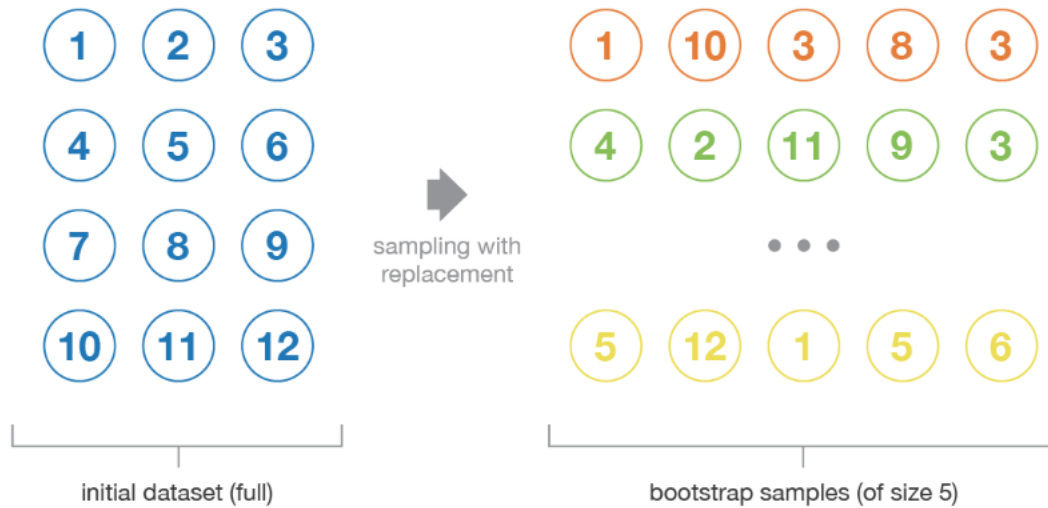


## Ensemble Learning

Stacking employs **heterogeneous weak learners** to learn **independently in parallel** then **train the meta-model from the output of weak learners**



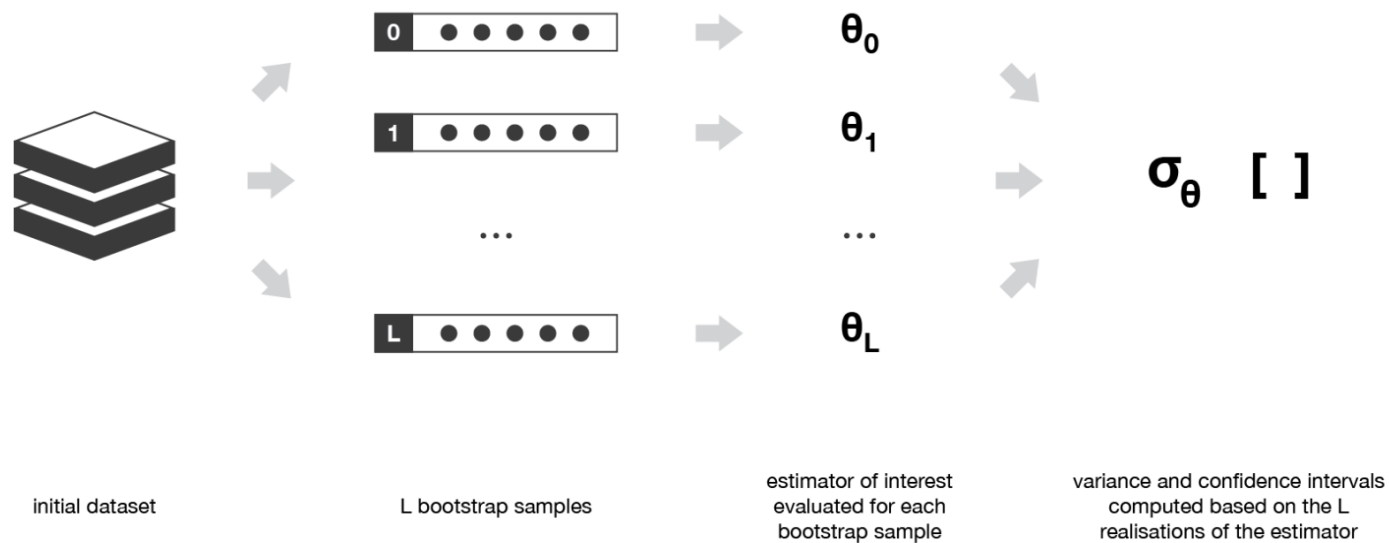
- Concepts: Random sampling with replacement



# Training Bagging Ensemble

Bootstrap sampling

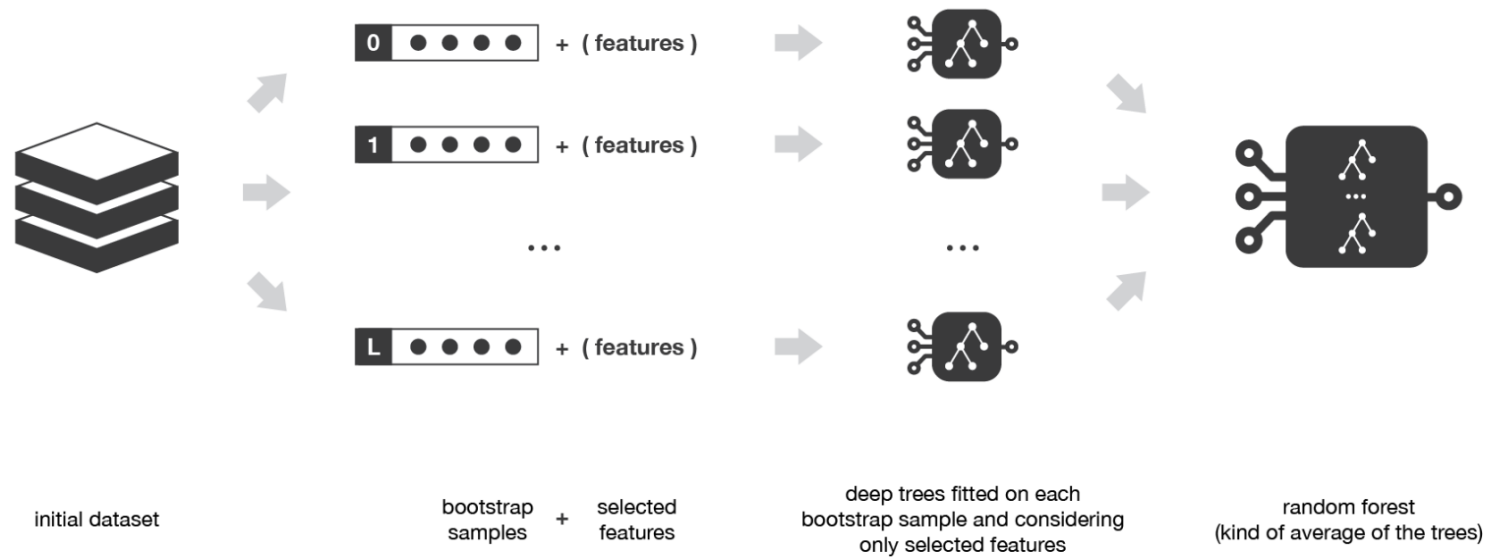
- Concepts



# Training Bagging Ensemble

Multiple sets of bootstrap samples are almost equivalent to the population

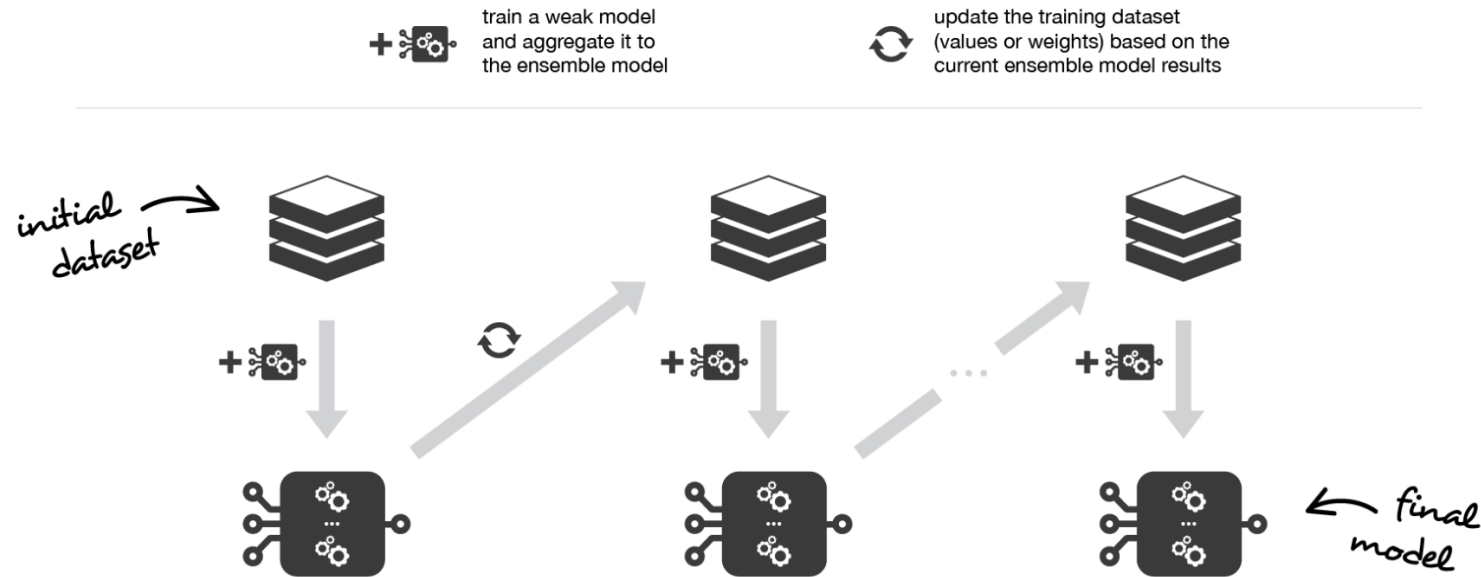
- Concepts



# Training Bagging Ensemble

Bootstrap samples is often used for training bagging ensemble model

- Concepts

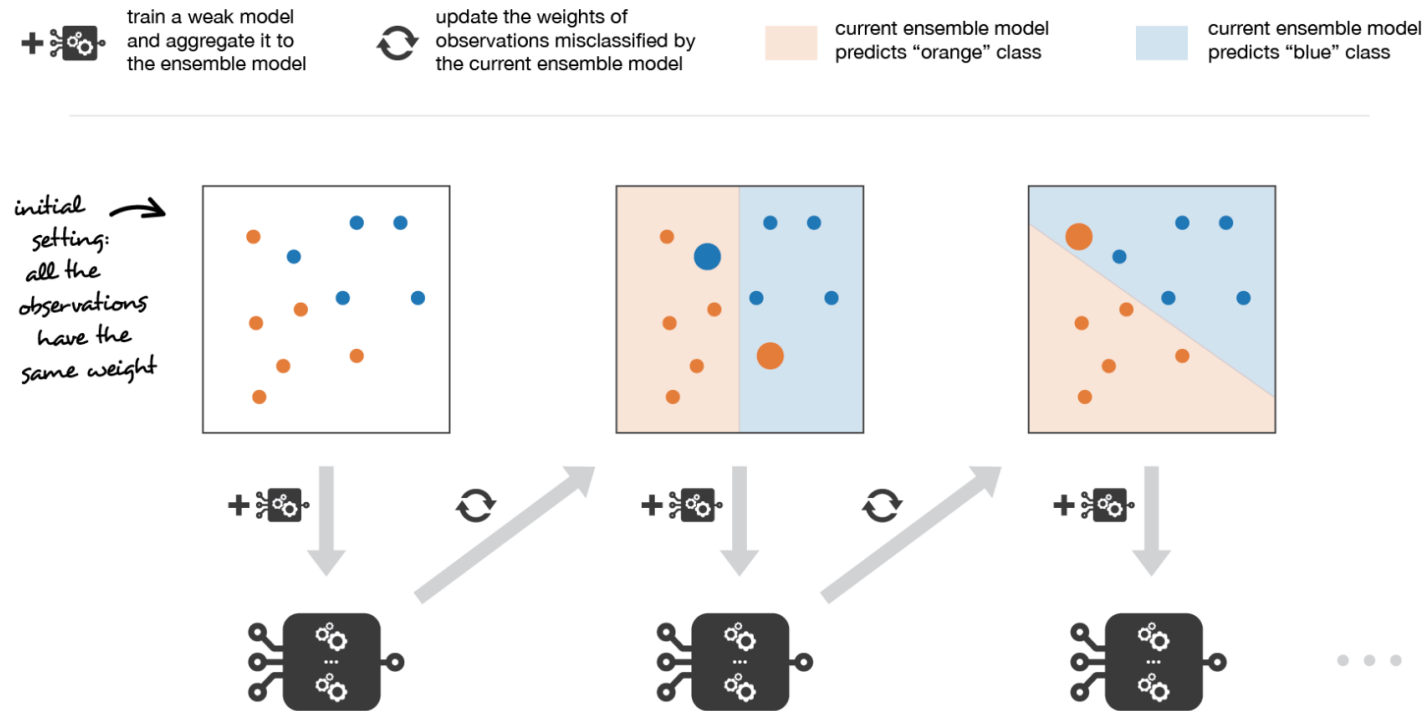


Boosting consists in, iteratively, fitting a weak learner, aggregate it to the ensemble model and "update" the training dataset to better take into account the strengths and weakness of the current ensemble model when fitting the next base model.

# Training Boosting Ensemble

Focus on reducing bias

- Adaptive Boosting (AdaBoost)



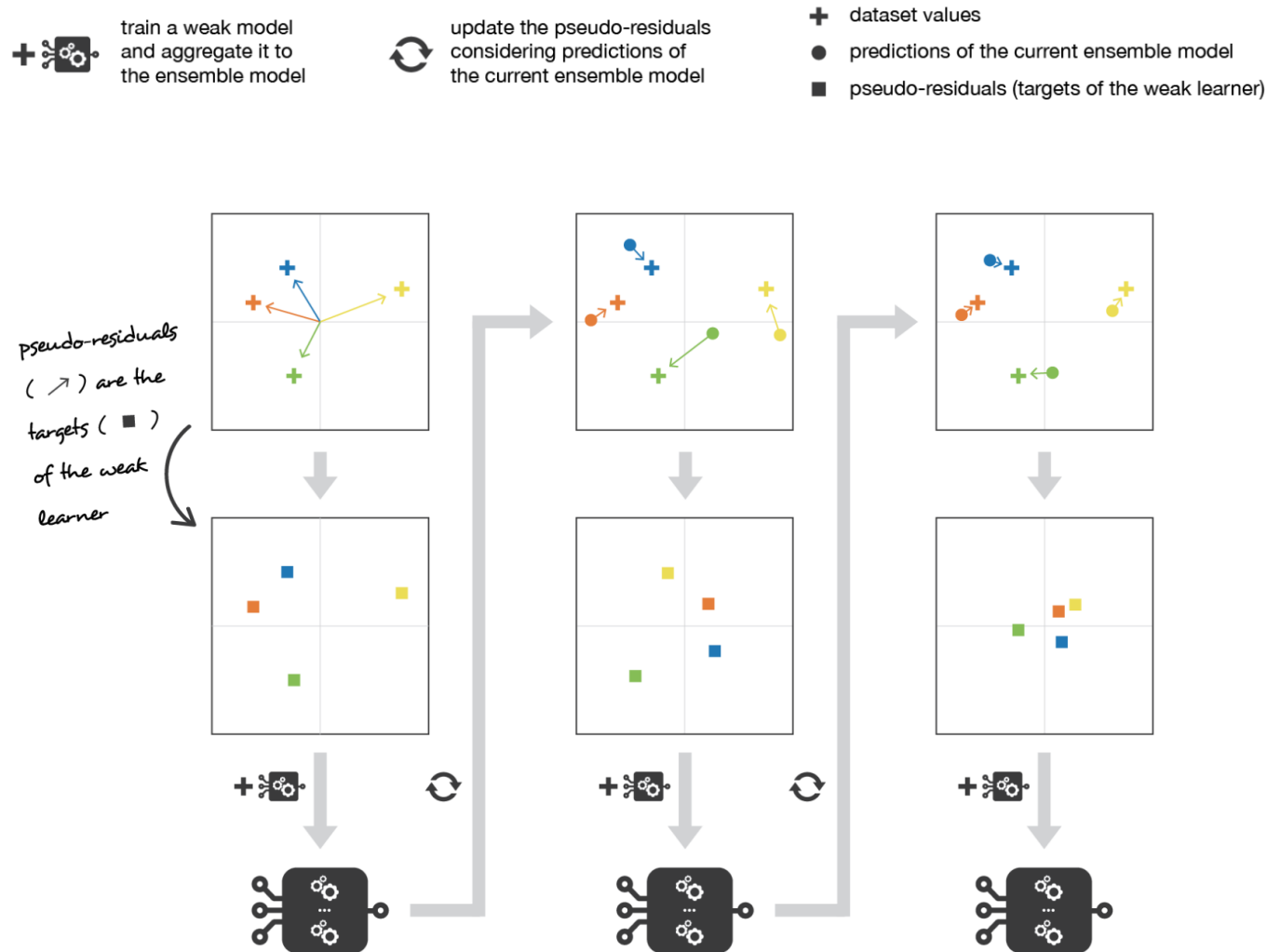
# Training Boosting Ensemble

Adaboost updates weights of the observations at each iteration

Weights of well classified observations decrease relatively to weights of misclassified observations

Models that perform better have higher weights in the final ensemble model

## ◦ Gradient Boosting

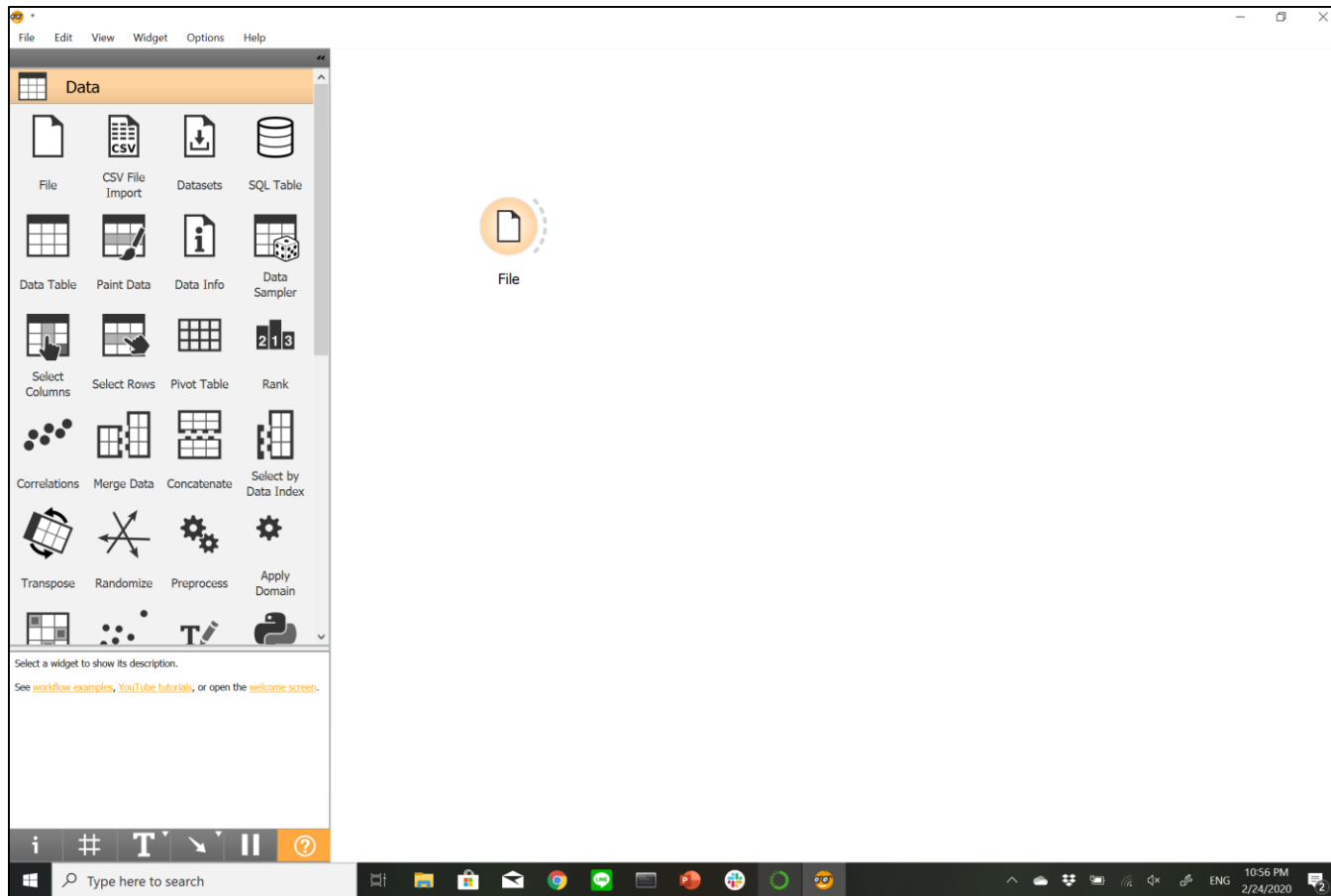


# Training Boosting Ensemble

Gradient boosting updates values of the observations at each iteration

Weak learners are trained to fit the pseudo-residuals that indicate in which direction to correct the current ensemble model predictions to lower the error

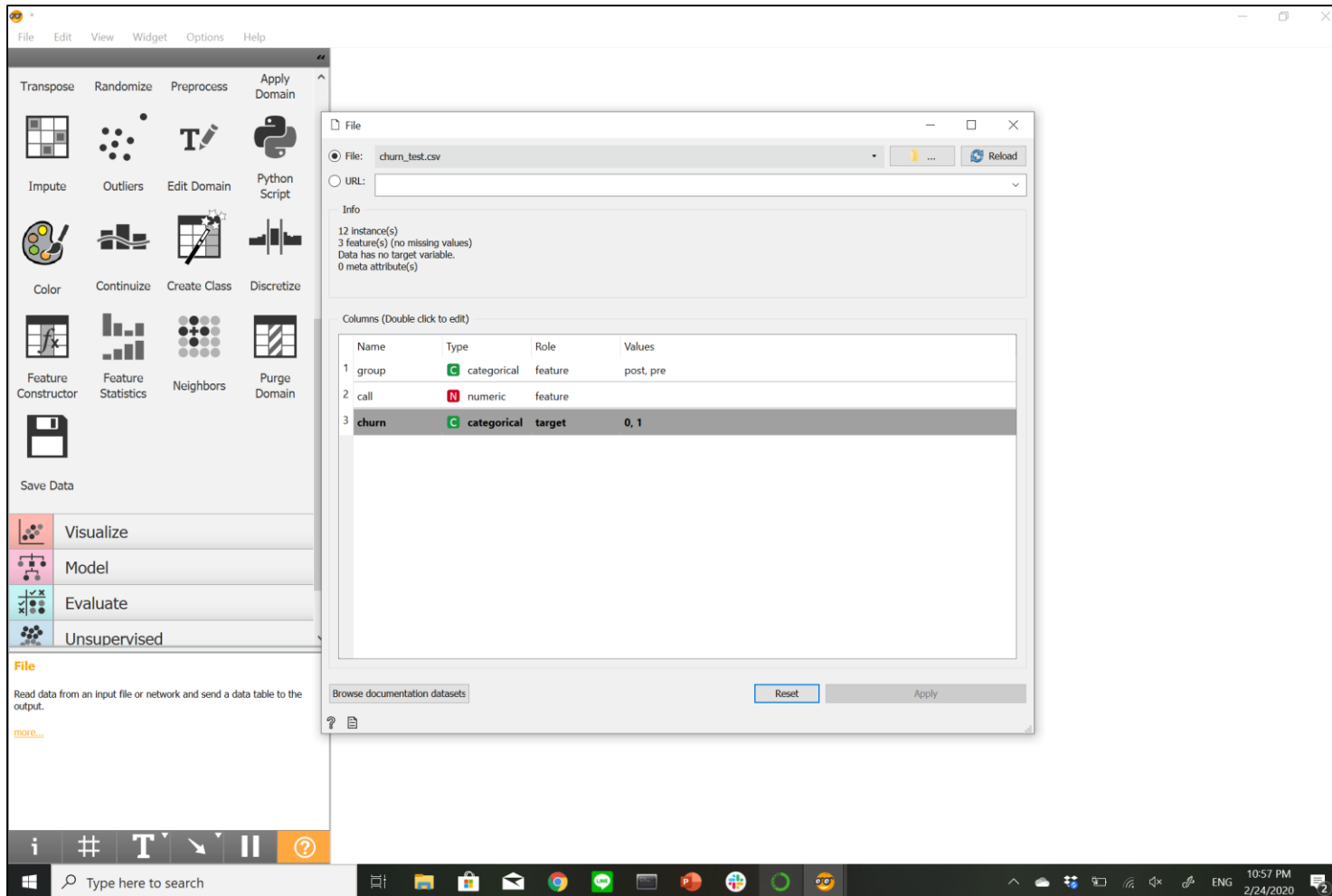
- Import data



# Decision Tree in Orange

Build your model in 10  
seconds

- Identify features and target



# Decision Tree in Orange

Build your model in 10  
seconds



- Add model

The screenshot displays the Orange3 data mining software interface. On the left, the 'Model' widget palette is visible, containing various machine learning algorithms such as Constant, CN2 Rule Induction, Calibrated Learner, kNN, Tree, Random Forest, SVM, Linear Regression, Logistic Regression, Naive Bayes, AdaBoost, Neural Network, Stochastic Gradient Descent, Stacking, Save Model, and Load Model. Below the palette, the 'Tree' widget is selected, showing a description: 'A tree algorithm with forward pruning.' and a link to 'more...'. The main workspace shows a workflow diagram with a 'File' widget (represented by a document icon) connected to a 'Tree' widget (represented by a tree icon) via a 'Data' connection line. The bottom of the screen shows the Windows taskbar with various application icons and the system clock indicating 7:08 PM on 3/1/2020.

# Decision Tree in Orange

Build your model in 10  
seconds

- Evaluate your model

The screenshot displays the Orange3 data mining software interface. The main workflow area shows a process starting with a 'File' widget, which connects to a 'Data' widget. This 'Data' widget then connects to a 'Tree' widget (a pink circle with a tree icon). The 'Tree' widget connects to a 'Learner' widget (a pink circle with a tree icon), which finally connects to a 'Test and Score' widget (a blue circle with a test tube icon). The 'Test and Score' widget is highlighted in the left sidebar under the 'Evaluate' category. Below the workflow, the 'Test and Score' widget's configuration window is open, showing the 'Evaluation Results' table.

**Test and Score**

Sampling

- ☐ Cross validation
  - Number of folds: 10
  - ☒ Stratified
- ☐ Cross validation by feature
- ☐ Random sampling
  - Repeat train/test: 10
  - Training set size: 70 %
  - ☒ Stratified
- ☐ Leave one out
- ☒ Test on train data
- ☐ Test on test data

Target Class

(Average over classes)

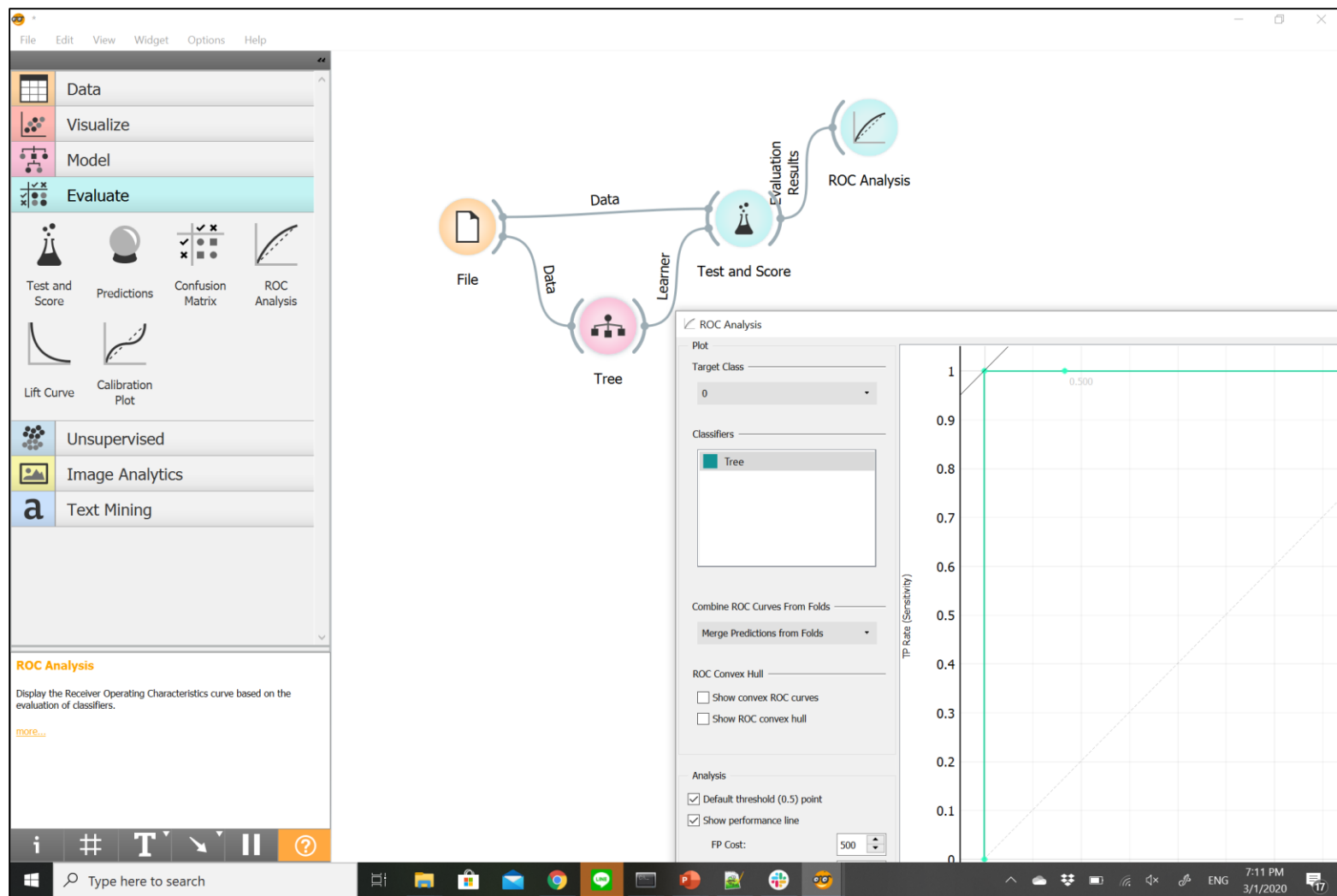
**Evaluation Results**

Model	AUC	CA	F1	Precision	Recall
Tree	0.986	0.917	0.916	0.929	0.917

# Decision Tree in Orange

Build your model in 10  
seconds

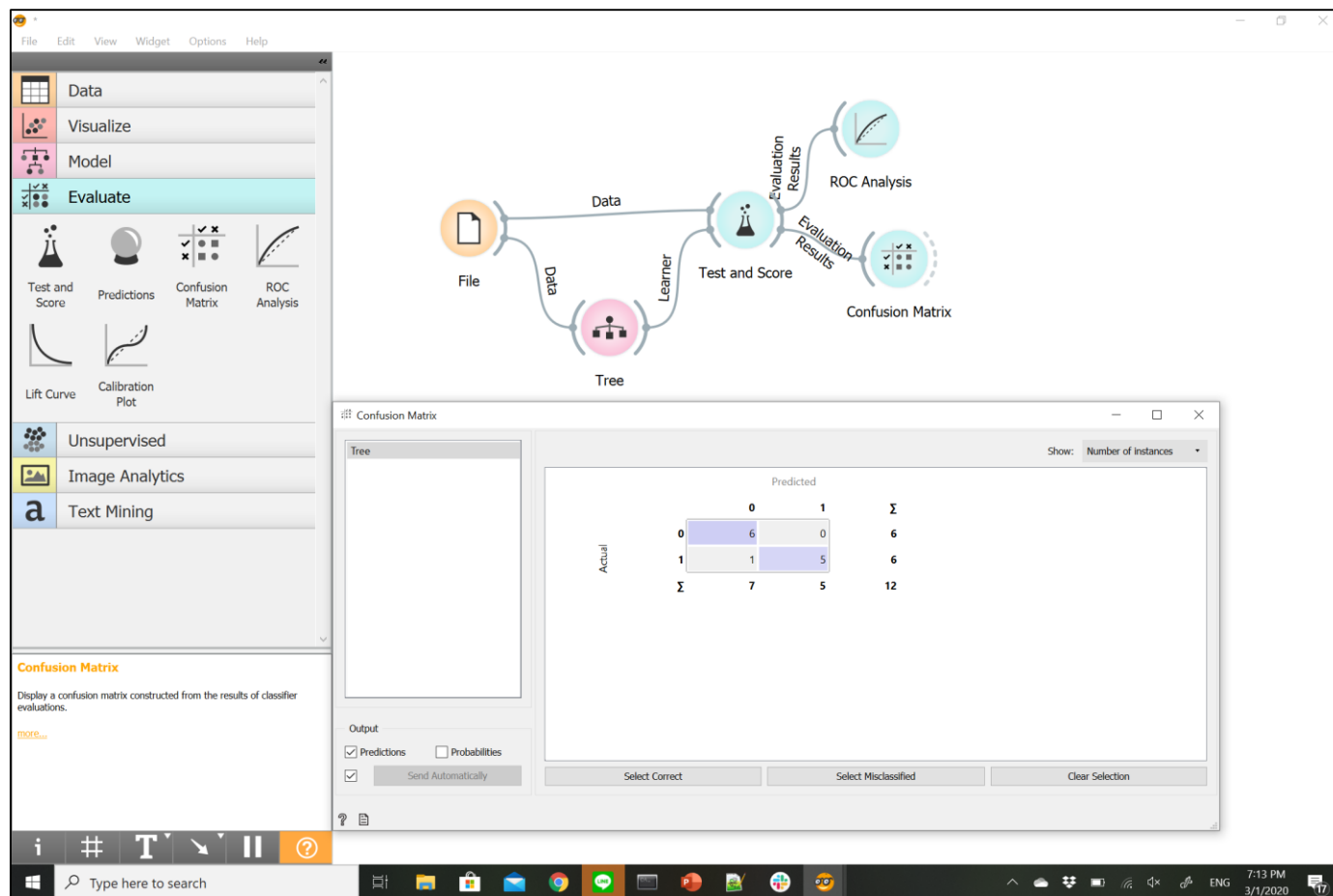
- ROC Plot



# Decision Tree in Orange

Build your model in 10  
seconds

- Confusion Matrix



# Decision Tree in Orange

Build your model in 10  
seconds

- View your tree

The screenshot displays the Orange3 data mining software interface. The main workflow area shows a sequence of widgets: 'File' (data source), 'Data' (data table), 'Learner' (model training), 'Test and Score' (evaluation), and 'Evaluation Results' (ROC Analysis and Confusion Matrix). A 'Tree' widget is also connected to the 'Learner' widget. The 'Tree Viewer' window is open, showing a decision tree with 5 nodes and 3 leaves. The tree structure is as follows:

- Root node (0, 50.0%, 6/12, call)
  - Left branch ( $\leq 4$ ) leads to node 0 (100%, 5/5).
  - Right branch ( $> 4$ ) leads to node 1 (85.7%, 6/7, call)
    - Left branch ( $\leq 5$ ) leads to node 0 (50.0%, 1/2).
    - Right branch ( $> 5$ ) leads to node 1 (100%, 5/5).

The 'Tree Viewer' window also includes a 'Display' section with settings for Zoom, Width, Depth (Unlimited), Edge width (Relative to parent), and Target class (None).

# Decision Tree in Orange

Build your model in 10  
seconds

- Add the random forest in to the canvas

The screenshot displays the Orange3 data mining software interface. On the left, the 'Model' widget palette is visible, containing various machine learning models such as Constant, CN2 Rule Induction, Calibrated Learner, kNN, Tree, Random Forest, SVM, Linear Regression, Logistic Regression, Naive Bayes, AdaBoost, Neural Network, Stochastic Gradient Descent, Stacking, Save Model, and Load Model. The 'Evaluate' section includes options for Unsupervised, Image Analytics, and Text Mining. The 'Test and Score' section shows cross-validation accuracy estimation.

The main canvas shows a workflow diagram. It starts with a 'File' widget connected to a 'Data' widget. The 'Data' widget is connected to a 'Random Forest' widget (labeled 'Learner'). The 'Random Forest' widget is connected to a 'Tree Data' widget (labeled 'Model'). The 'Tree Data' widget is connected to a 'Tree Viewer' widget. The 'Random Forest' widget is also connected to a 'Test and Score' widget (labeled 'Learner'). The 'Test and Score' widget is connected to an 'Evaluation Results' widget, which is further connected to 'ROC Analysis' and 'Confusion Matrix' widgets.

The 'Test and Score' widget window is open, showing the 'Evaluation Results' table. The table displays performance metrics for the 'Tree' and 'Random Forest' models across different sampling methods.

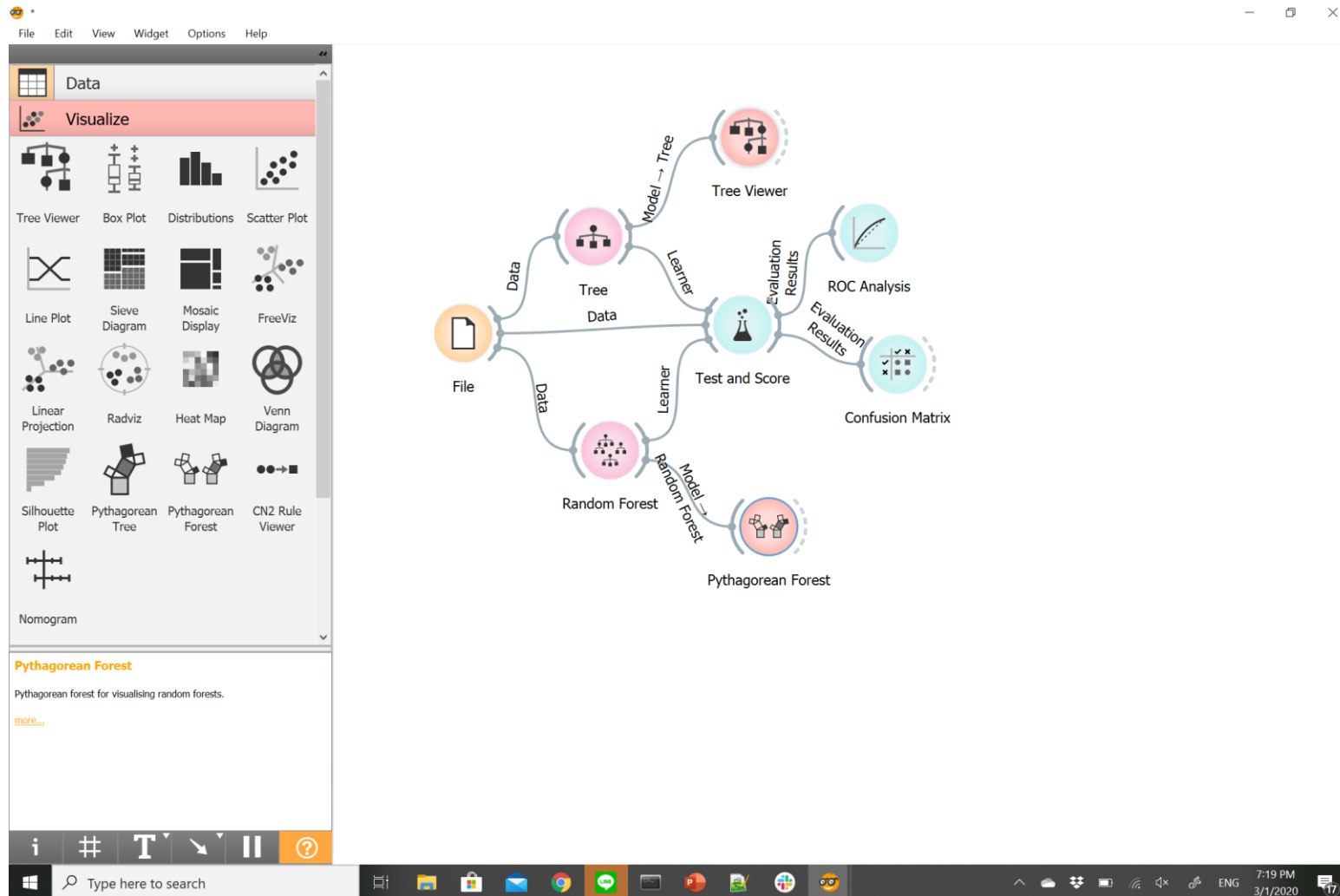
Model	AUC	CA	F1	Precision	Recall
Tree	0.986	0.917	0.916	0.929	0.917
Random Forest	1.000	1.000	1.000	1.000	1.000

# Random Forest in Orange

Build your model in 10 seconds

You can train multiple models together

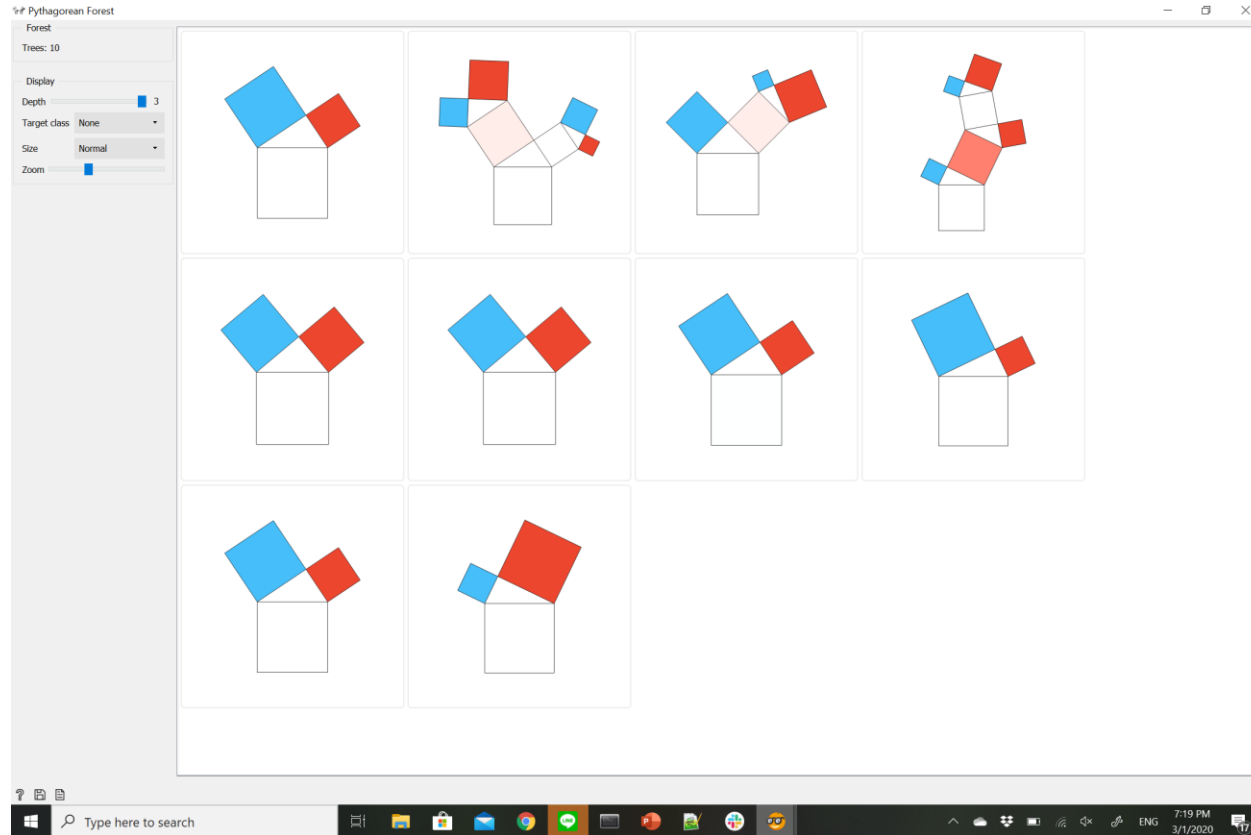
- Visualize your random forest with the Pythagorean forest



# Random Forest in Orange

Build your model in 10 seconds

- Visualize your random forest with the Pythagorean forest



# Random Forest in Orange

Build your model in 10  
seconds



- Build your model using Kaggle dataset

## Exercise

Our sample data is very small for demonstration purpose

Now it is the time to work with the dataset churn prediction of telecom in Kaggle Competition