ML for natural and physical scientists 2023 6

CART

dr.federica bianco | fbb.space | fedhere | fedhere

this slide deck:

https://slides.com/federicabianco/mlpn23_6

1

Supervise learning

what is machine learning?

supervised learning

classification
prediction
feature selection

unsupervised learning

understanding structure
organizing/compressing data
anomaly detection
dimensionality reduction

supervised learning methods
(nearly all other methods you heard of)
learns by example

used to:

classify, predict (regression)

- Similarity can be used in conjunction to parametric or non-parametric methods

- Need labels, in some cases a lot of labels
- Dependent on the definition of similarity

supervised learning methods
(nearly all other methods you heard of)
learns by example

used to:

classify, predict (regression)

- Similarity can be used in conjunction to parametric or non-parametric methods

- Need labels, in some cases a lot of labels
- Dependent on the definition of similarity

clustering vs classifying

unsupervised *supervised*

goal is to partition the space so that the **unobserved** variables are

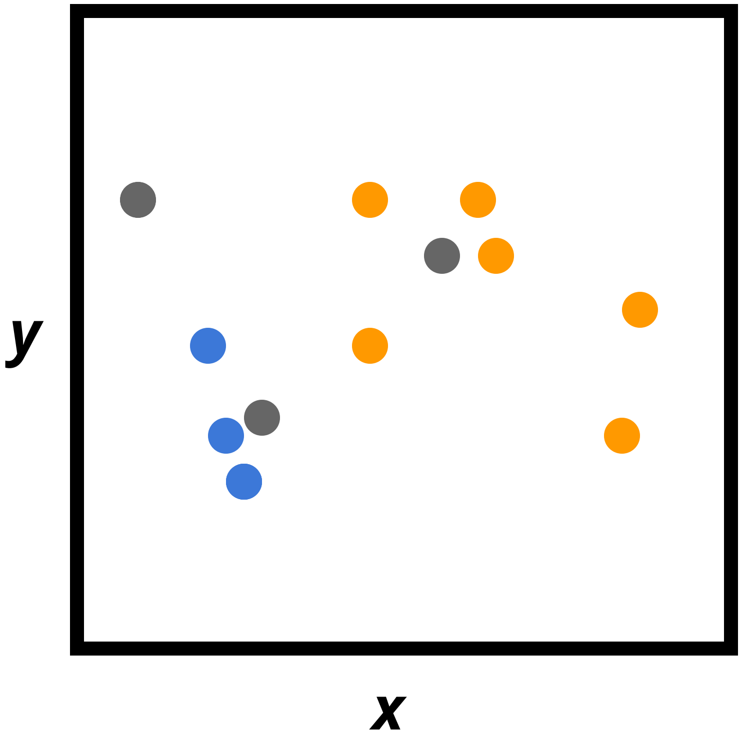
separated in groups

consistently with

an observed subset

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

models typically return a partition of the space

supervised ML: classification

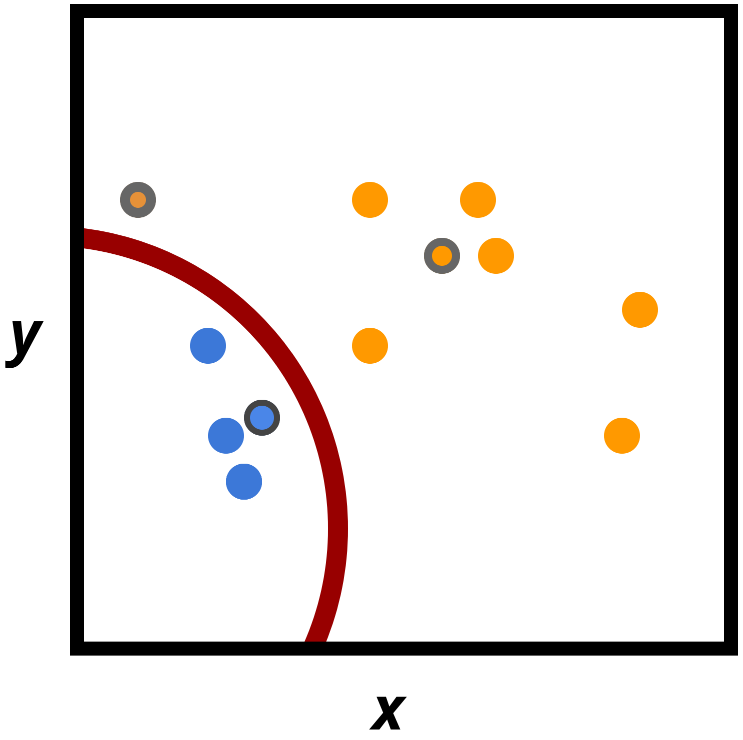
A subset of variables has class labels.
Guess the label for the other variables

SVM

finds a hyperplane that optimally separates observations

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

```
if x**2 + y**2 <= (x-a)**2 + (y-b)**2 :
        return blue
else:
        return orange
```

supervised ML: classification

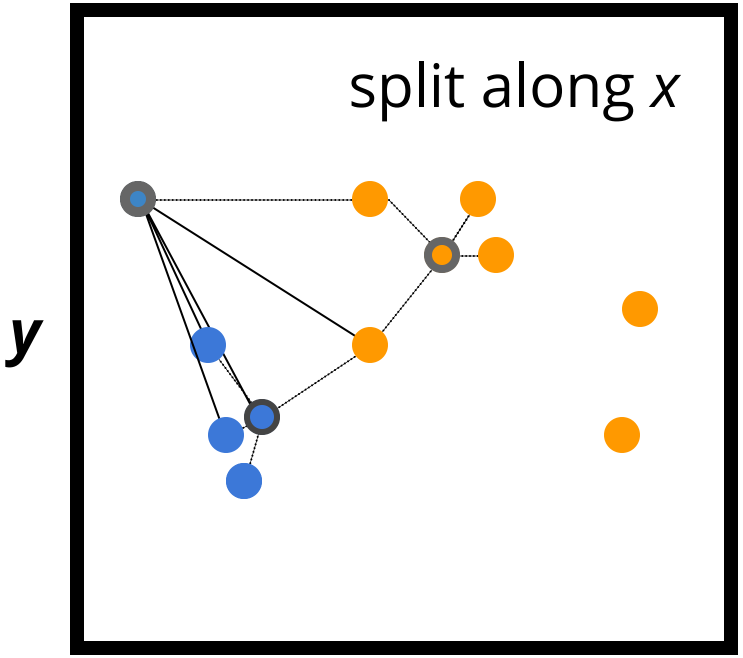
A subset of variables has class labels.
Guess the label for the other variables

KNearest Neighbors

Assigns the class of closest neighbors

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

```
k = 4
if (label[argsort(distance((x,y), trainingset))][:k] == "blue").sum() > (labels[argsort(distance((x,y), trainingset))][:k] == "orange").sum():
        return blue
return orange
```

supervised ML: classification

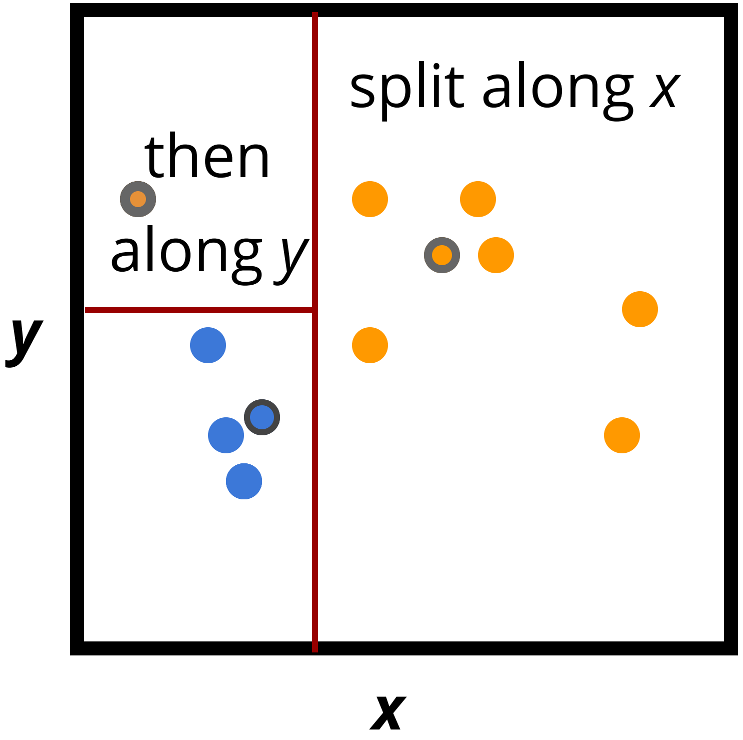
A subset of variables has class labels.
Guess the label for the other variables

Tree Methods

split spaces along each axis separately

observed features:

$(\vec{x}, \vec{y})$



target features:

$(\overrightarrow{color})$

```
if x <= a :
        if y <= b:
                return blue
return orange
```

2

CART

classification and regression trees

single tree

4

Application:

a robot to predict surviving the Titanic

(Kaggle)

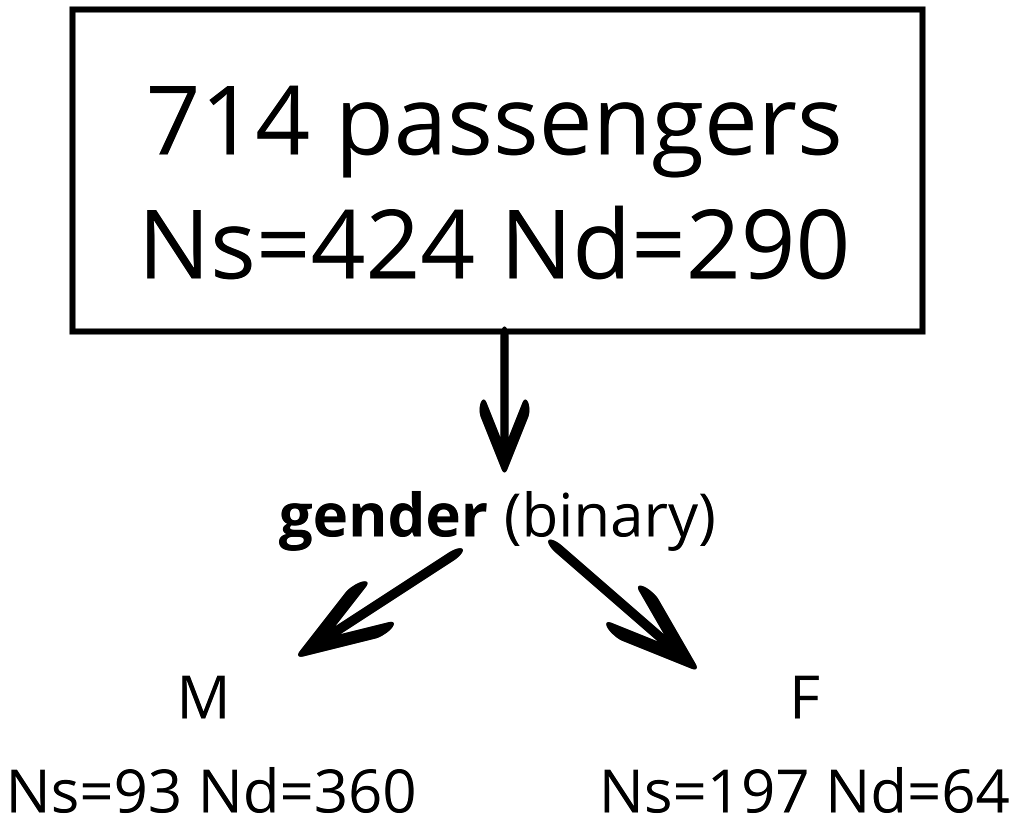
https://www.kaggle.com/c/titanic

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

https://www.kaggle.com/c/titanic

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)

714 passengers

Ns=424 Nd=290

gender (binary)

M

Ns=93 Nd=360

$$p = \frac{360}{360+93}$$

F

Ns=197 Nd=64

$$p = \frac{197}{197+64}$$

optimize over purity:

$$p = \frac{N_{largest\ class}}{N_{totalset}}$$

Application:

a robot to predict surviving the Titanic

(Kaggle)

https://www.kaggle.com/c/titanic

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)

714 passengers
Ns=424 Nd=290

gender (binary)

M

Ns=93 Nd=360

$p = 79\%$

F

Ns=197 Nd=64

$p = 75\%$

optimize over purity:

$$p = \frac{N_{largest\ class}}{N_{totalset}}$$

Application:

a robot to predict surviving the Titanic

(Kaggle)

https://www.kaggle.com/c/titanic

features:

- gender 79%|75%
- ticket class 66 | 54%
- age

target variable:

-> survival (y/n)

714 passengers
Ns=424 Nd=290

class (ordinal)

1st

Ns=120 Nd=80

$p = 66\%$

2nd +3rd

Ns=234 Nd=298

$p = 54\%$

Application:

a robot to predict surviving the Titanic

(Kaggle)

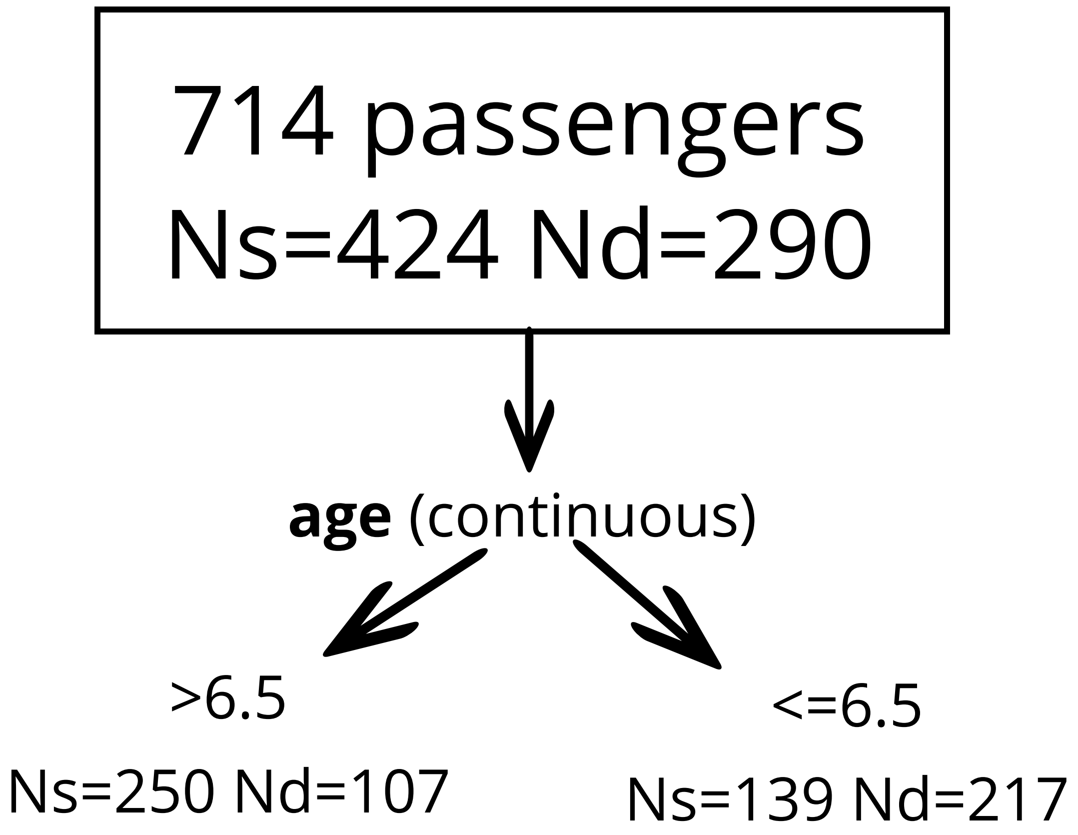
https://www.kaggle.com/c/titanic

features:

- gender 79%|75%
- ticket class 66% | 54%
- age 66% | 61%

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

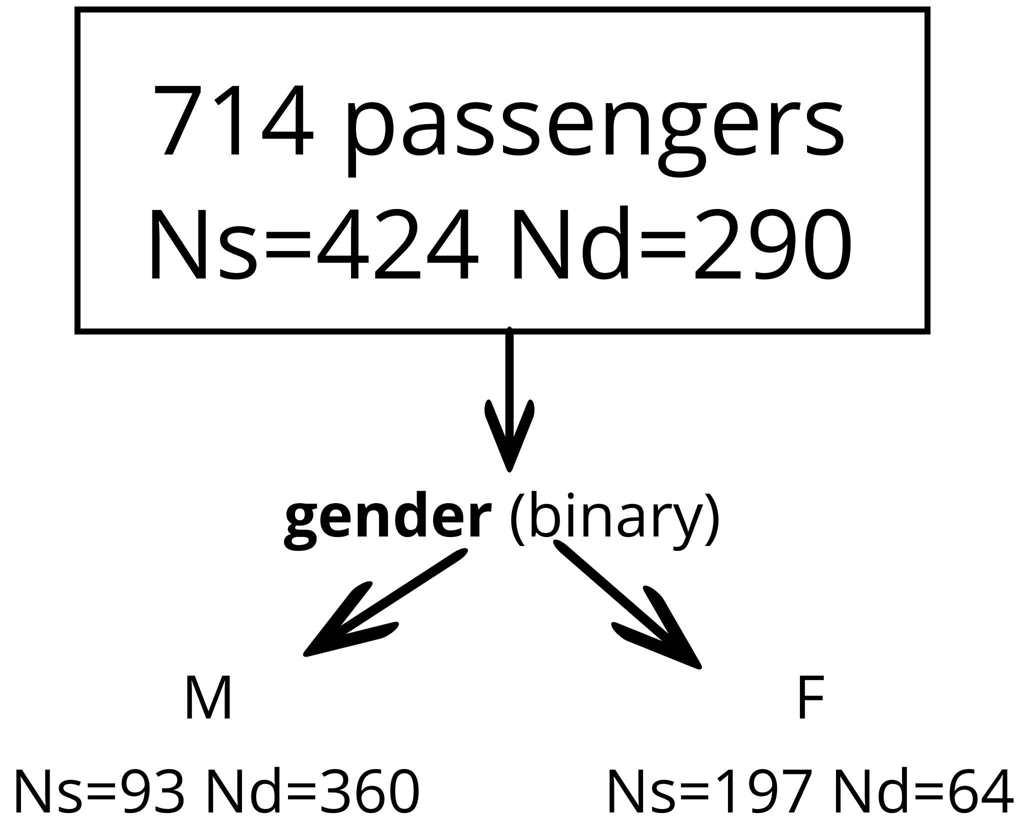
https://www.kaggle.com/c/titanic

features:

- gender 79|75%
- ticket class *M* 60|85% *F* 96|65%
- age *M* 74|67% *F* 66|60%

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

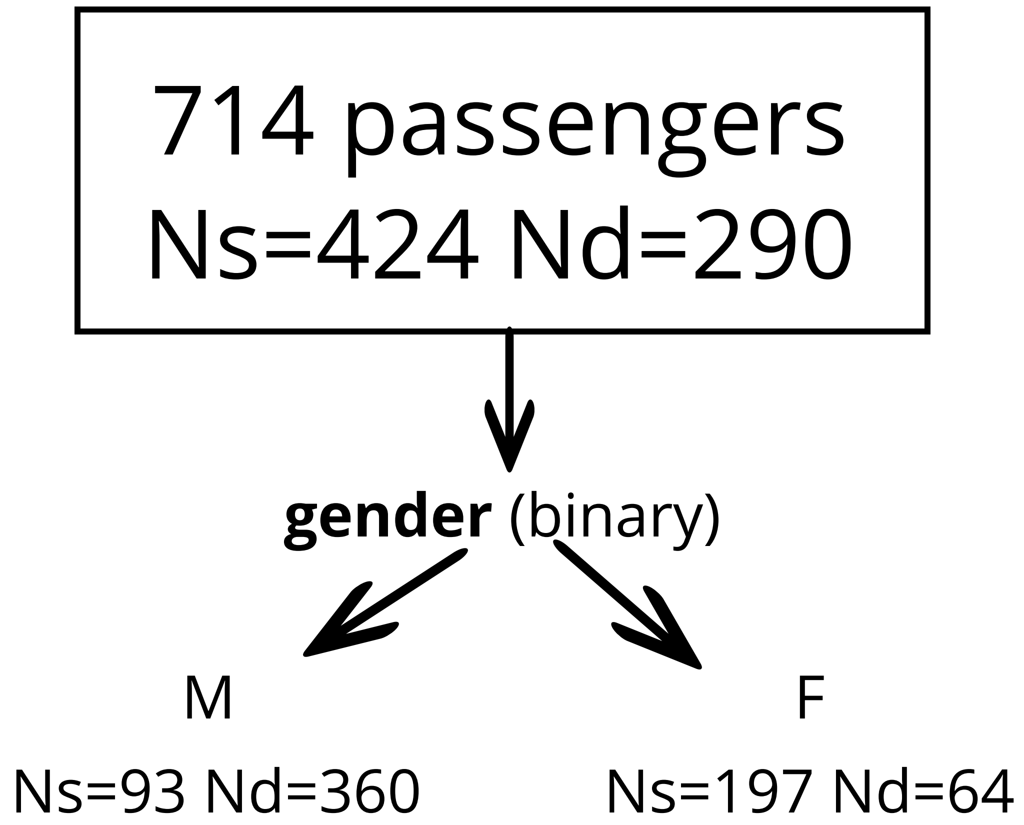
https://www.kaggle.com/c/titanic

features:

- gender 79|75%
- ticket class *M* 60|85% ***F* 96|65%**
- age ***M* 74|67%** *F* 66|60%

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

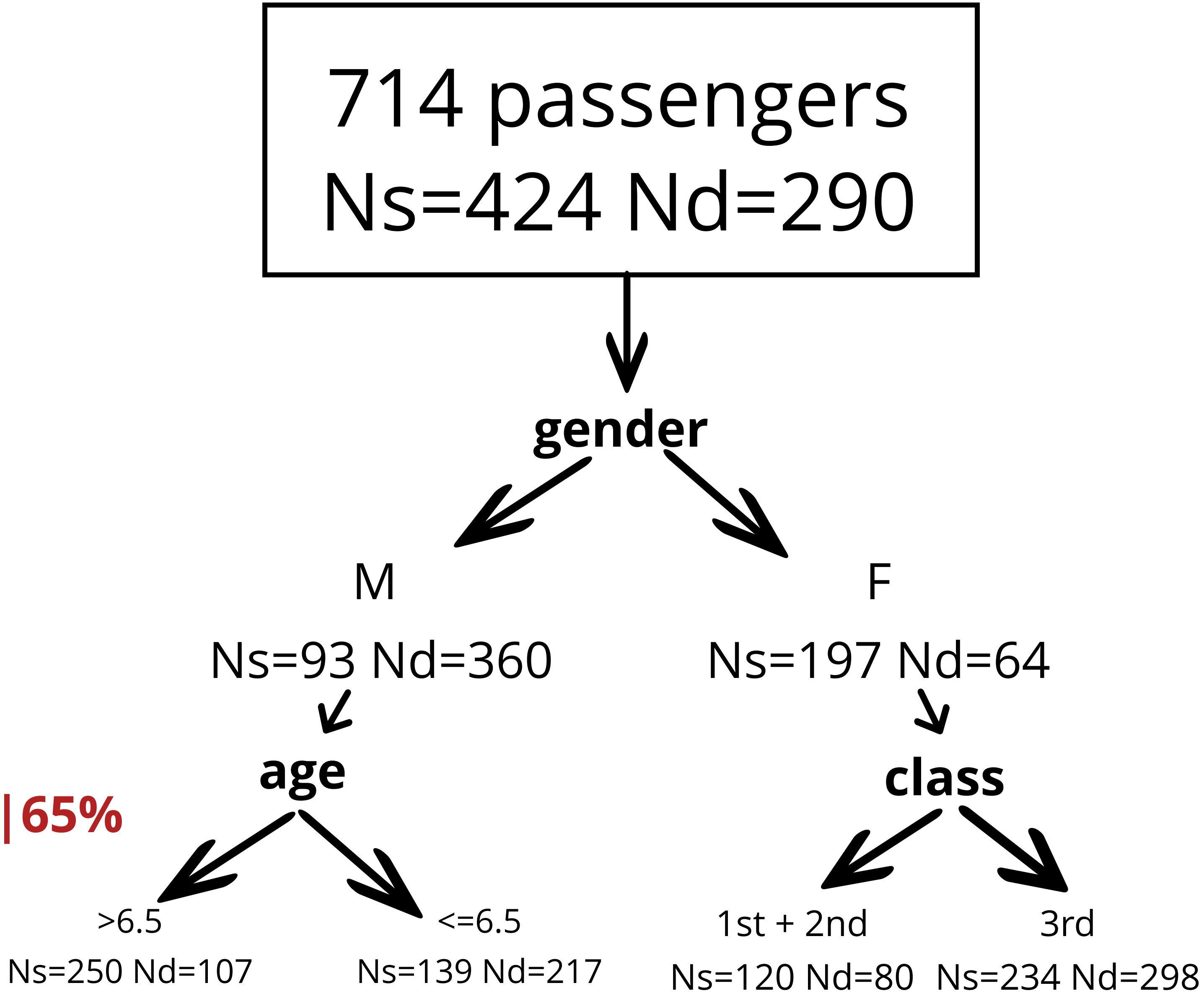
https://www.kaggle.com/c/titanic

features:

- gender 79|75%
- ticket class *M* 60|85% ***F* 96|65%**
- age ***M* 74|67%** *F* 66|60%

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

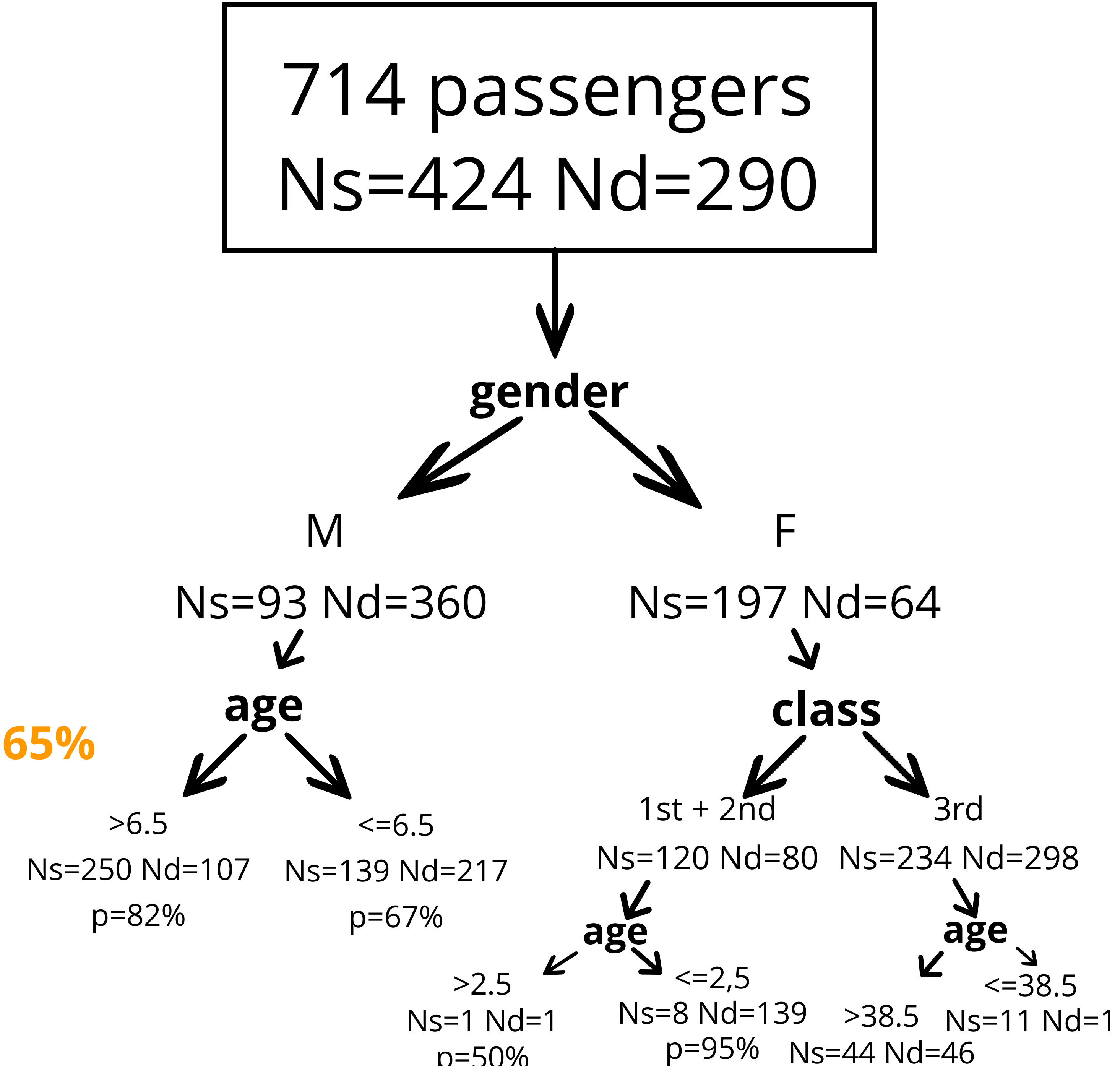
https://www.kaggle.com/c/titanic

features:

- gender 79|75%
- ticket class *M* 60|85% ***F* 96|65%**
- age ***M* 74|67%** *F* 66|60%

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

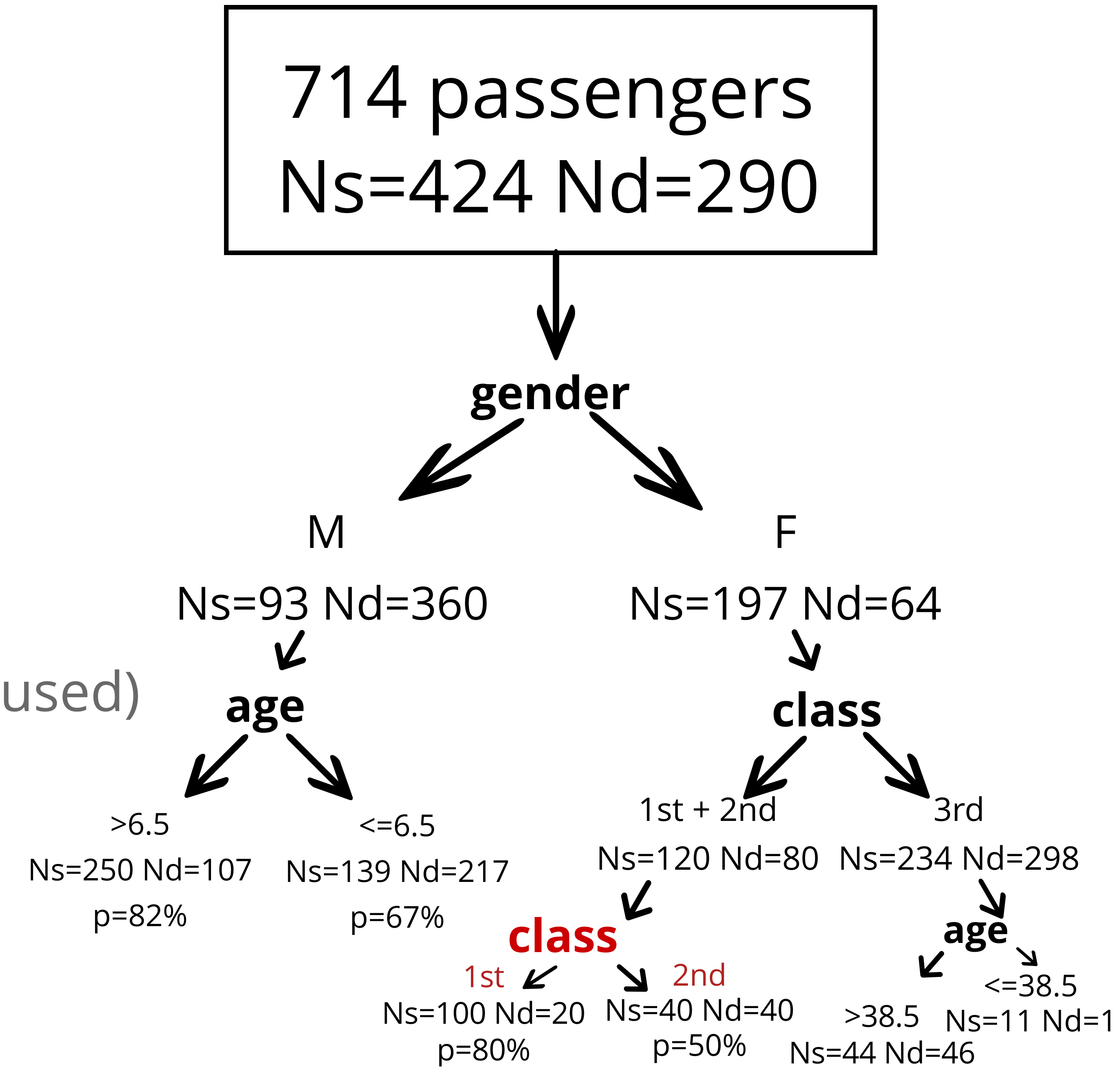
https://www.kaggle.com/c/titanic

features:

- gender (binary already used)
- ticket class (*ordinal*)
- age (continuous)

target variable:

-> survival (y/n)



A single tree

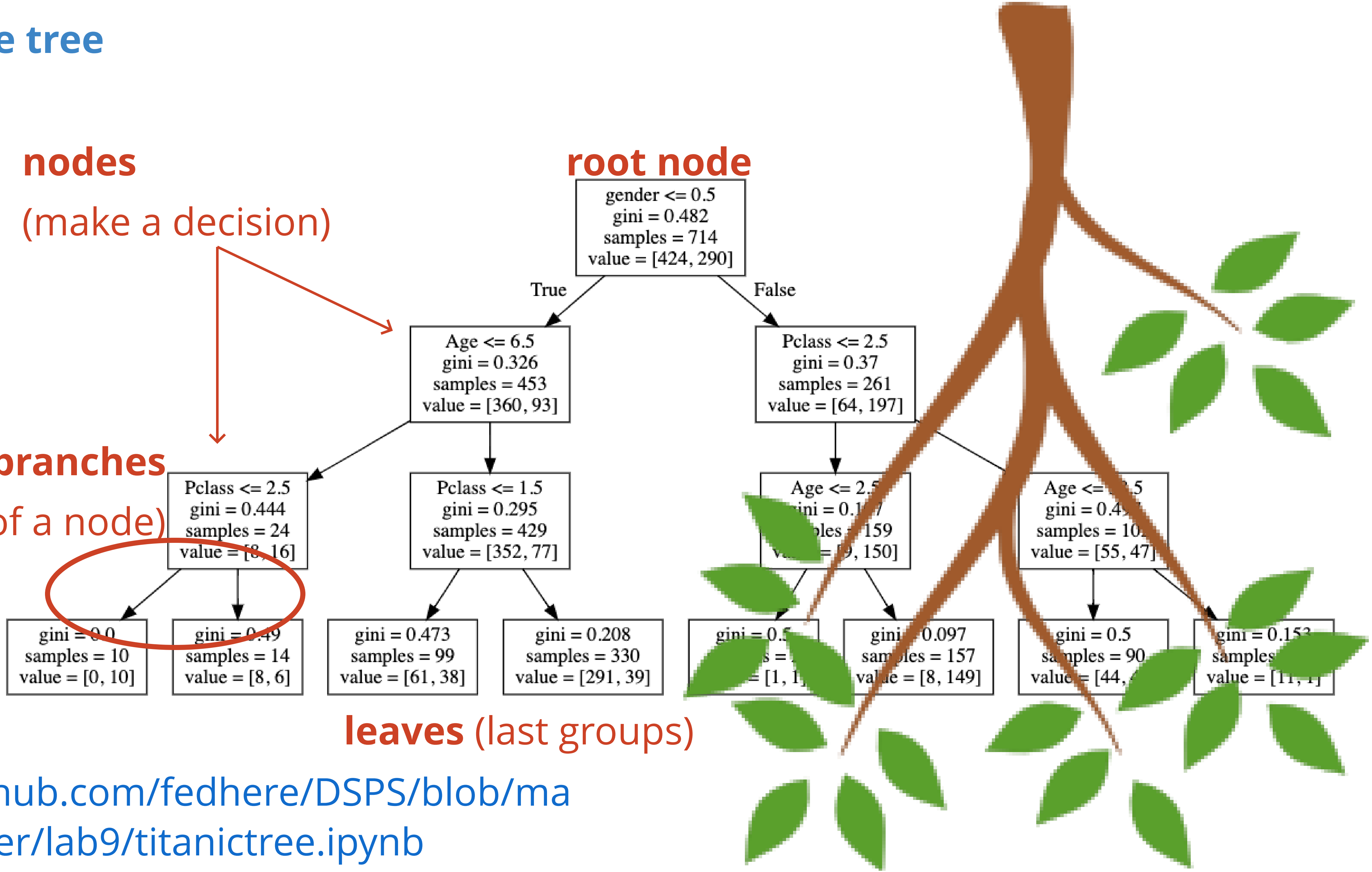
nodes

(make a decision)

branches

(split off of a node)

root node



leaves (last groups)

https://github.com/fedhere/DSPS/blob/master/lab9/titanictree.ipynb

tree hyperparameters

sklearn.tree.DecisionTreeClassifier¶

class `sklearn.tree.` **`DecisionTreeClassifier`**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*)

[source]

A single tree: hyperparameters

criterion : *string, optional (default="gini")*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

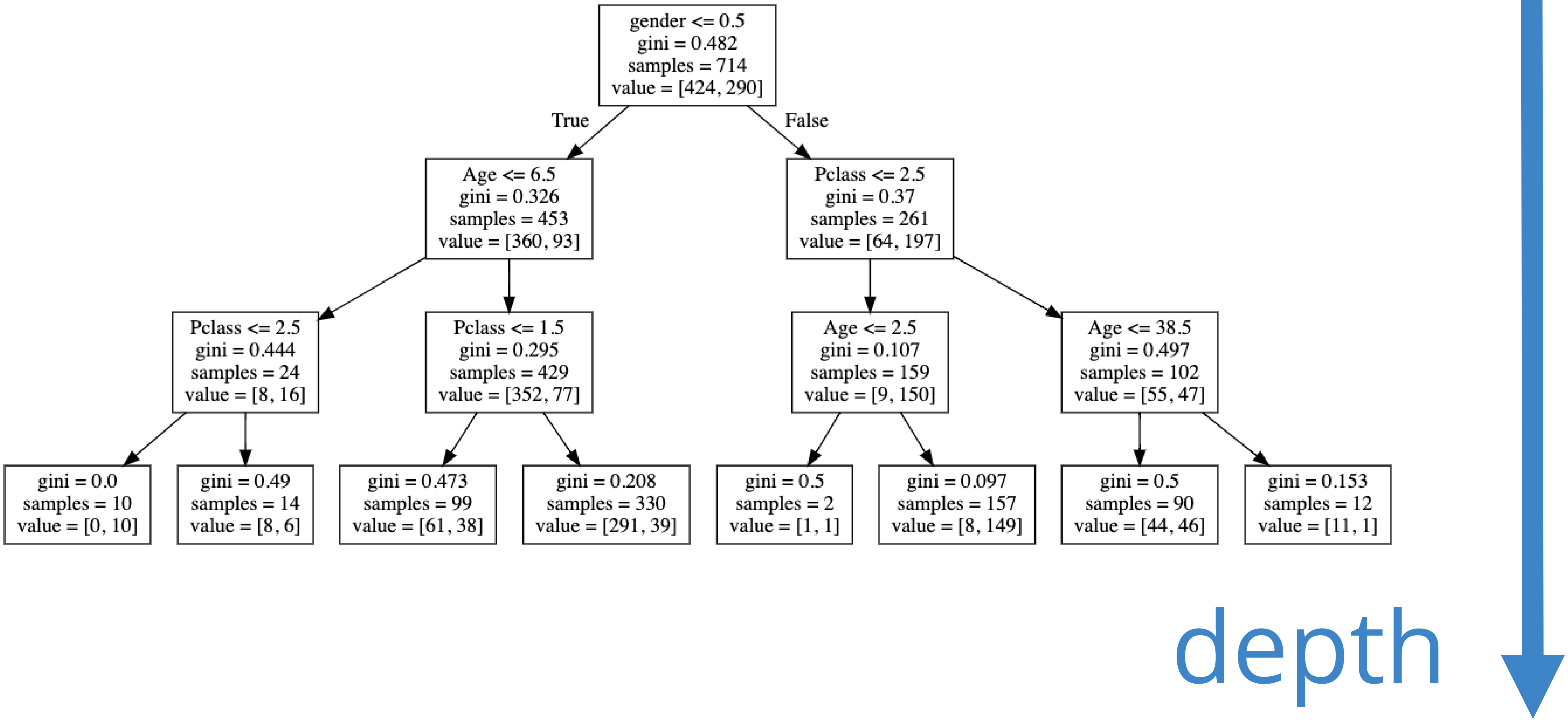
gini impurity

$$\mathrm{I}_G(p) = 1 - \sum_{i=1}^{J} {p_i}^2$$

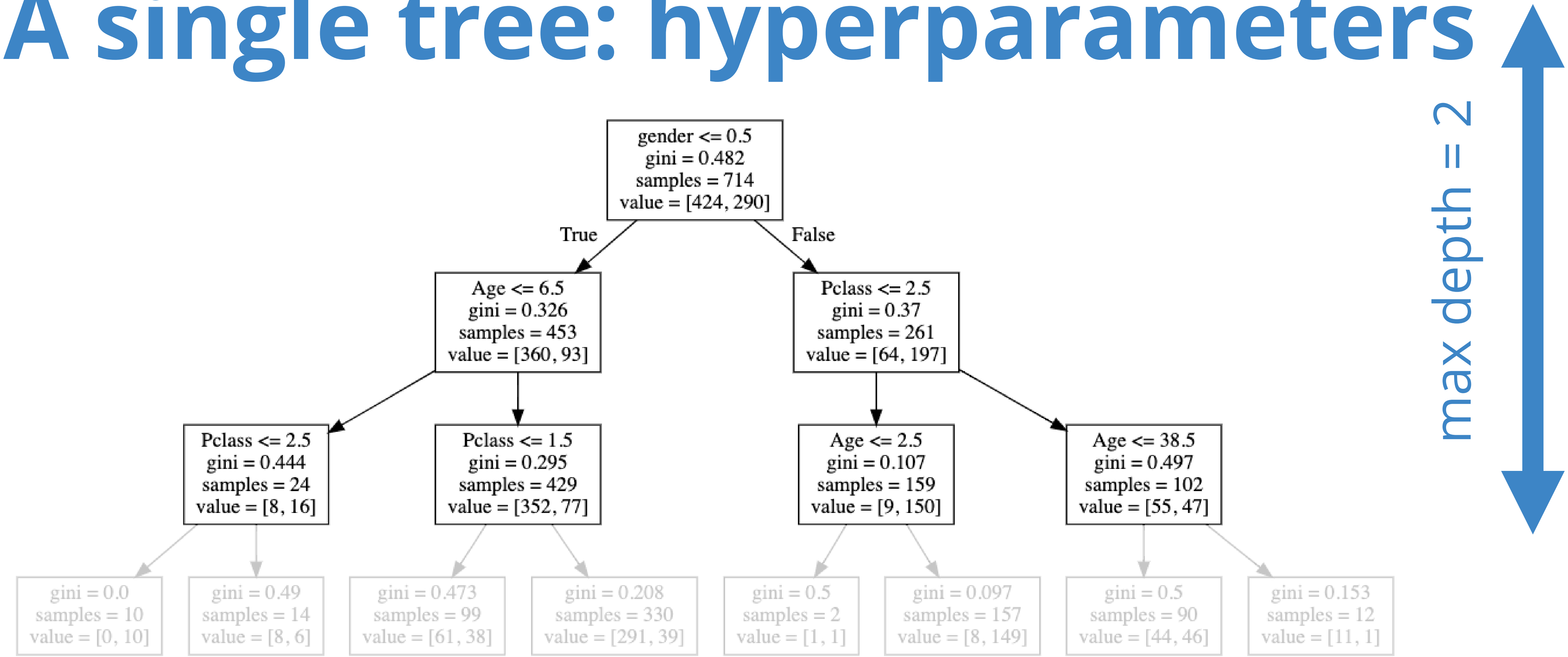
information gain (entropy)

$$\mathrm{H}(T) = - \sum_{i=1}^{J} p_i \log_2 p_i$$

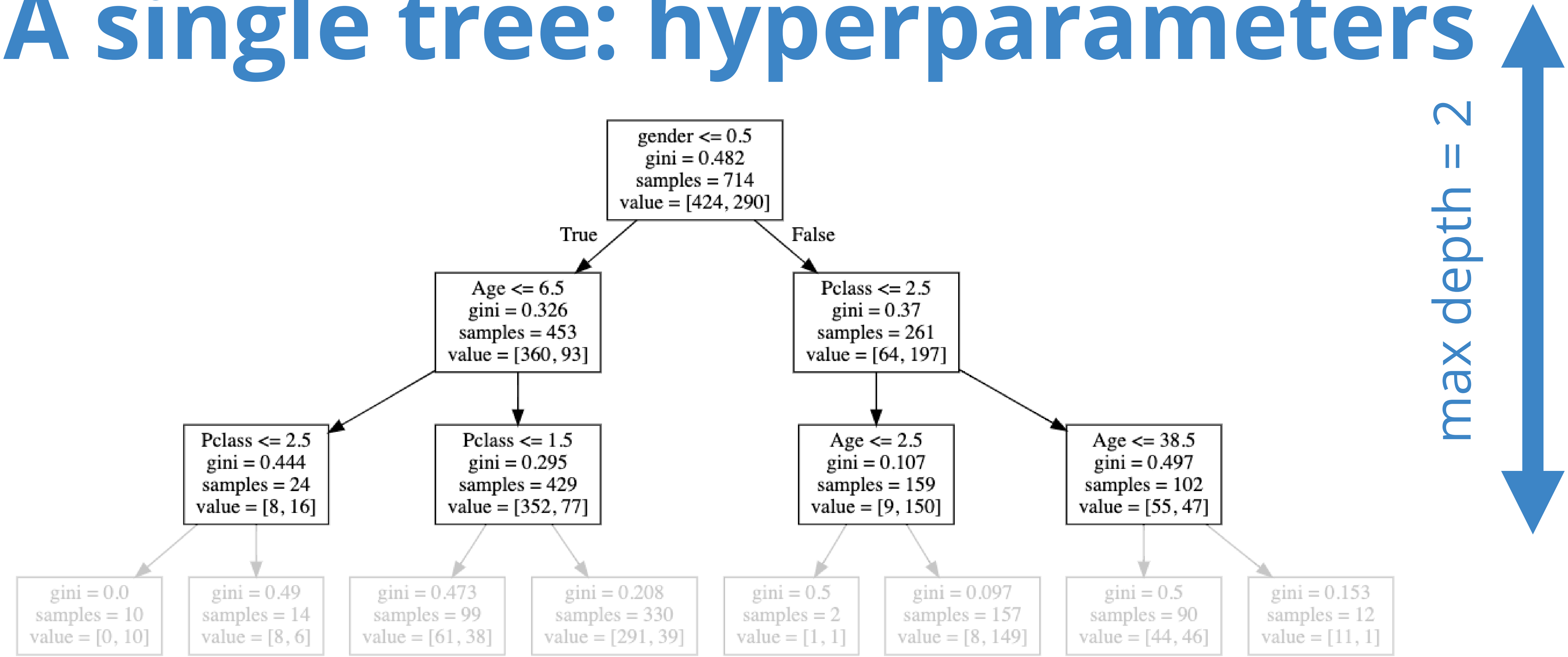
A single tree: hyperparameters



A single tree: hyperparameters

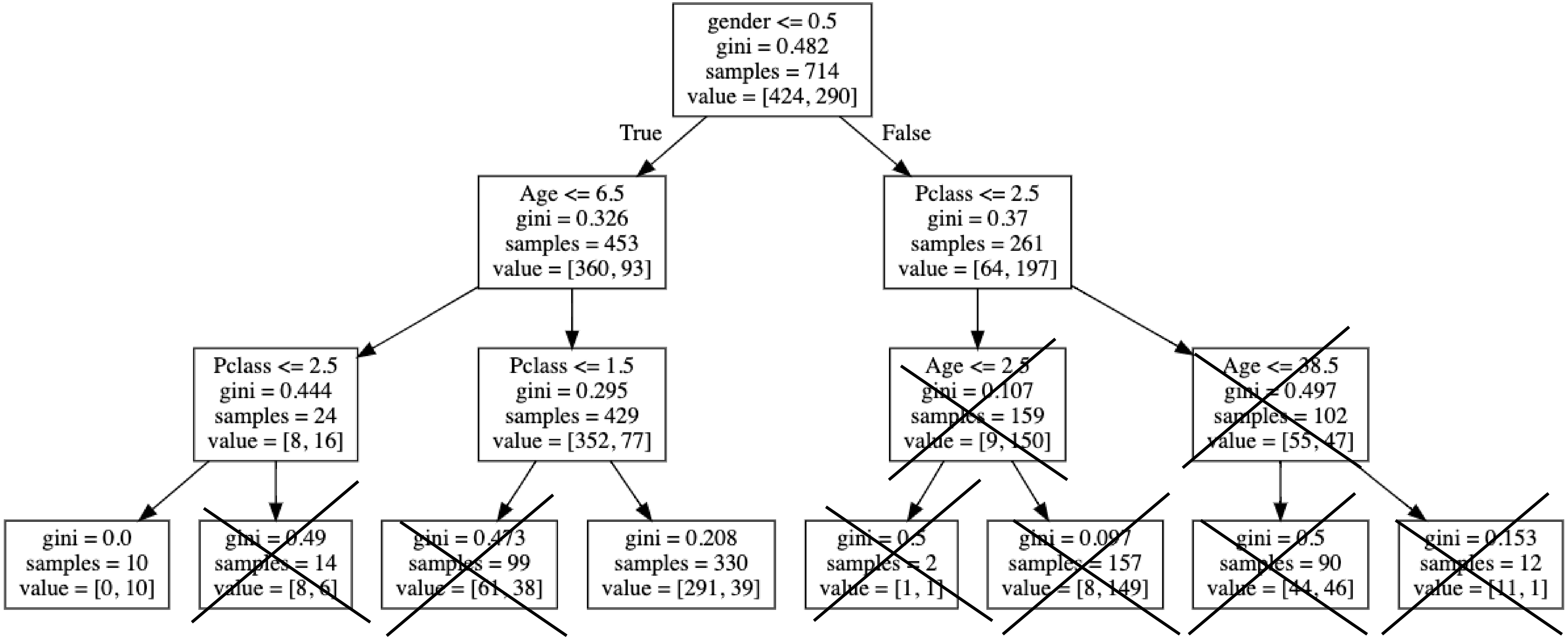


A single tree: hyperparameters



PREVENTS OVERGFITTING

A single tree: hyperparameters



alternative: tree pruning

regression with trees

42

CART: Classification and Regression Trees

Trees can be used for regression

(think about it as very many small classes)

tree ensambles

43

issues with trees

variance:

different trees lead to different results

issues with trees

variance:

different trees lead to different results

why?

because calculating the criterion for every split and every mote is an untractable problem!

e.g. 2 coutinuous variables would be a problem of order ∞^2

issues with trees

variance:

different trees lead to different results

solution

run many trees and take an "ensamble" decision!

Random Forests

a bunch of parallel trees

Gradient Boosted Trees

a series of trees

44

Random Forest and Gradient Boosted Trees

ensemble methods

run multiple versions of the same model with some small (stochastic or progressive) variation and learn from the emsemble of methods

tree ensemble methods

Random forest:

trees run in parallel (independently of each other)

each tree uses a random subset of observations/features (boostrap - bagging)

class predicted by majority vote:

what class do most trees think a point belong to

Gradient boosted trees:

trees run in series (one after the other)

each tree uses different *weights* for the features learning the weighs from the previous tree

the last tree has the prediction

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) ¶ [source]

sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) ¶ [source]

sklearn.ensemble.GradientBoostingClassifier

class sklearn.ensemble.**GradientBoostingClassifier**(*, *loss='log_loss'*, *learning_rate=0.1*, *n_estimators=100*, *subsample=1.0*, *criterion='friedman_mse'*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_depth=3*, *min_impurity_decrease=0.0*, *init=None*, *random_state=None*, *max_features=None*, *verbose=0*, *max_leaf_nodes=None*, *warm_start=False*, *validation_fraction=0.1*, *n_iter_no_change=None*, *tol=0.0001*, *ccp_alpha=0.0*) ¶

[source]

sklearn.ensemble.GradientBoostingRegressor

class `sklearn.ensemble.GradientBoostingRegressor`(*, *loss='squared_error', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) ¶ [source]

ML model performance

6

ML model performance

Accuracy, Recall, Precision

$$LR = \frac{\text{False Negative}}{\text{True Negative}}$$

	H0 is True	H0 is False
H0 is falsified	Type I Error False Positive	True Positive
H0 is not falsified	True Negative	Type II Error False Negative

ML model performance

Accuracy, Recall, Precision

$$LR = \frac{\text{False Negative}}{\text{True Negative}}$$

	H0 is True	H0 is False
H0 is falsified	important message spammed	True Positive
H0 is not falsified	True Negative	spam in your inbox

ML model performance

Accuracy, Recall, Precision

$$\textbf{\textit{Precision}} = \frac{TP}{TP + FP}$$

$$\textbf{\textit{Recall}} = \frac{TP}{TP + FN}$$

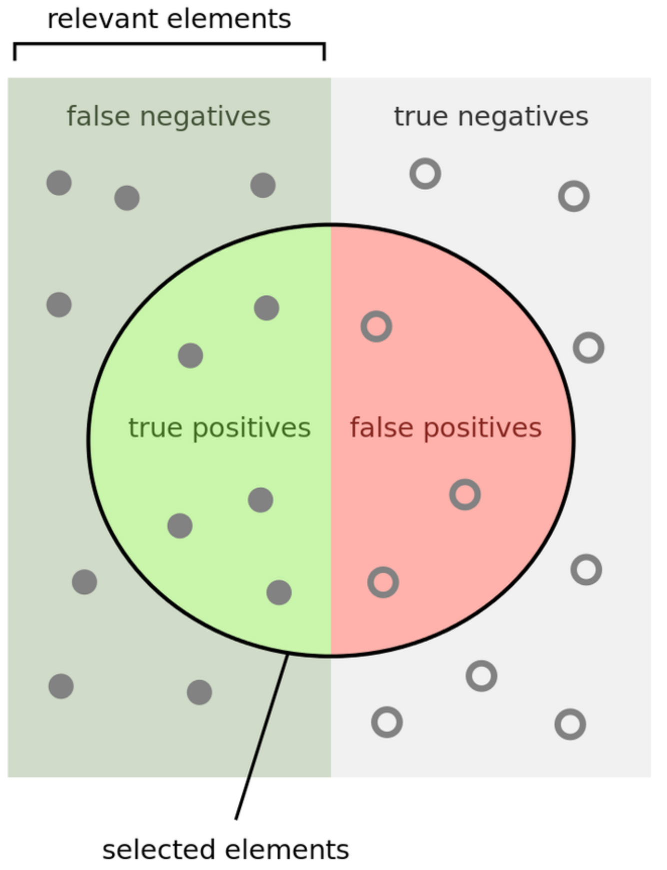
$$\textbf{\textit{Accuracy}} = \frac{TP + TN}{TP + TN + FP + FN}$$

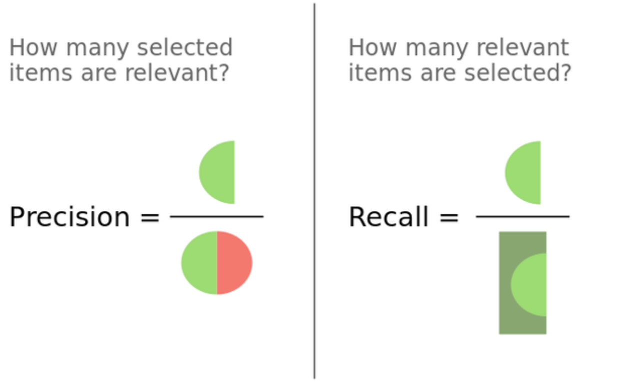
TP=True Positive

FP=False Positive

TN=True Negative

FN=False Positive





https://en.wikipedia.org/wiki/Precision_and_recall

ML model performance

Accuracy, Recall, Precision

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

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

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

TP=True Positive

FP=False Positive

TN=True Negative

FN=False Positive

Formula

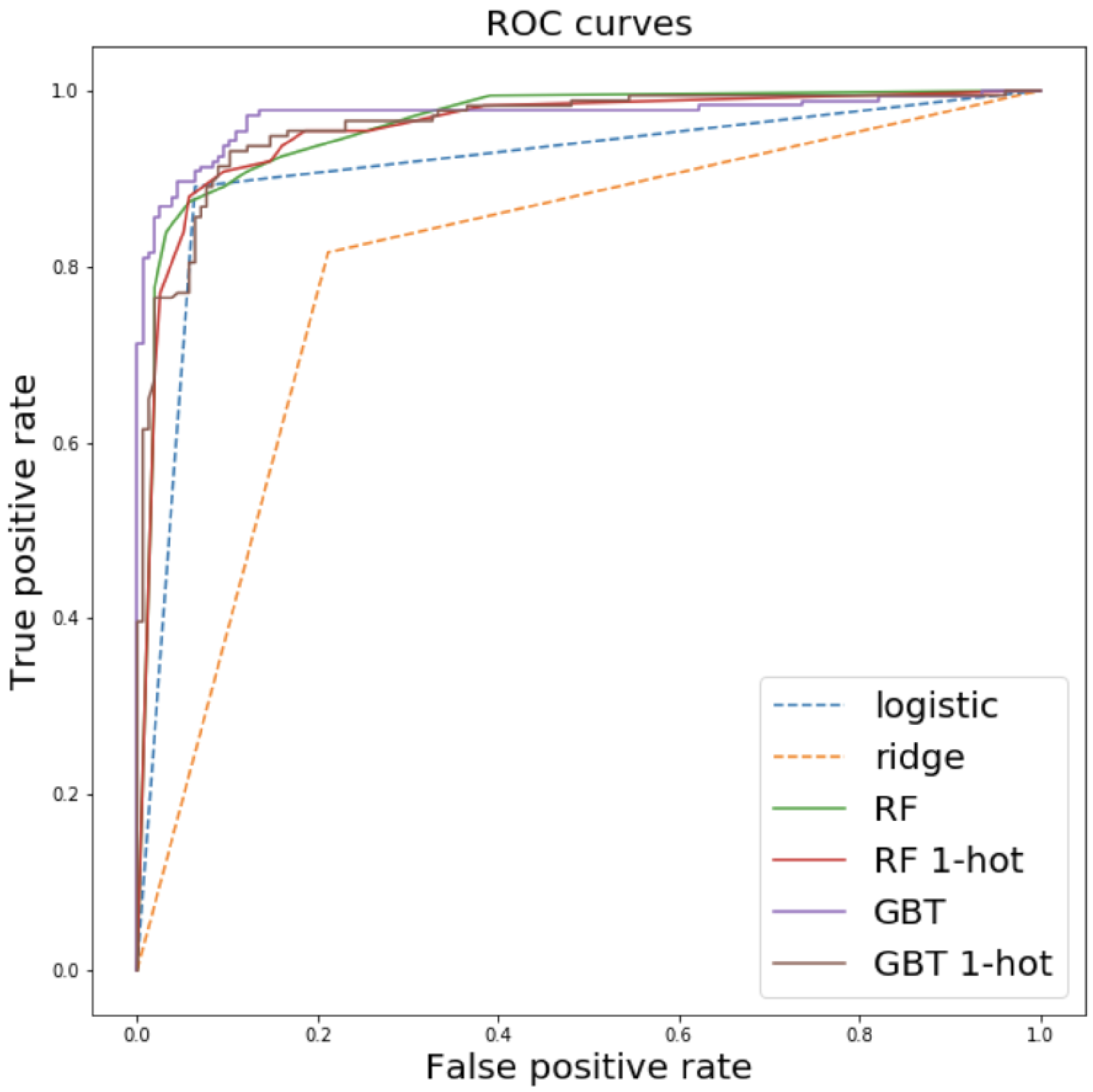
$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

TP = number of true positives

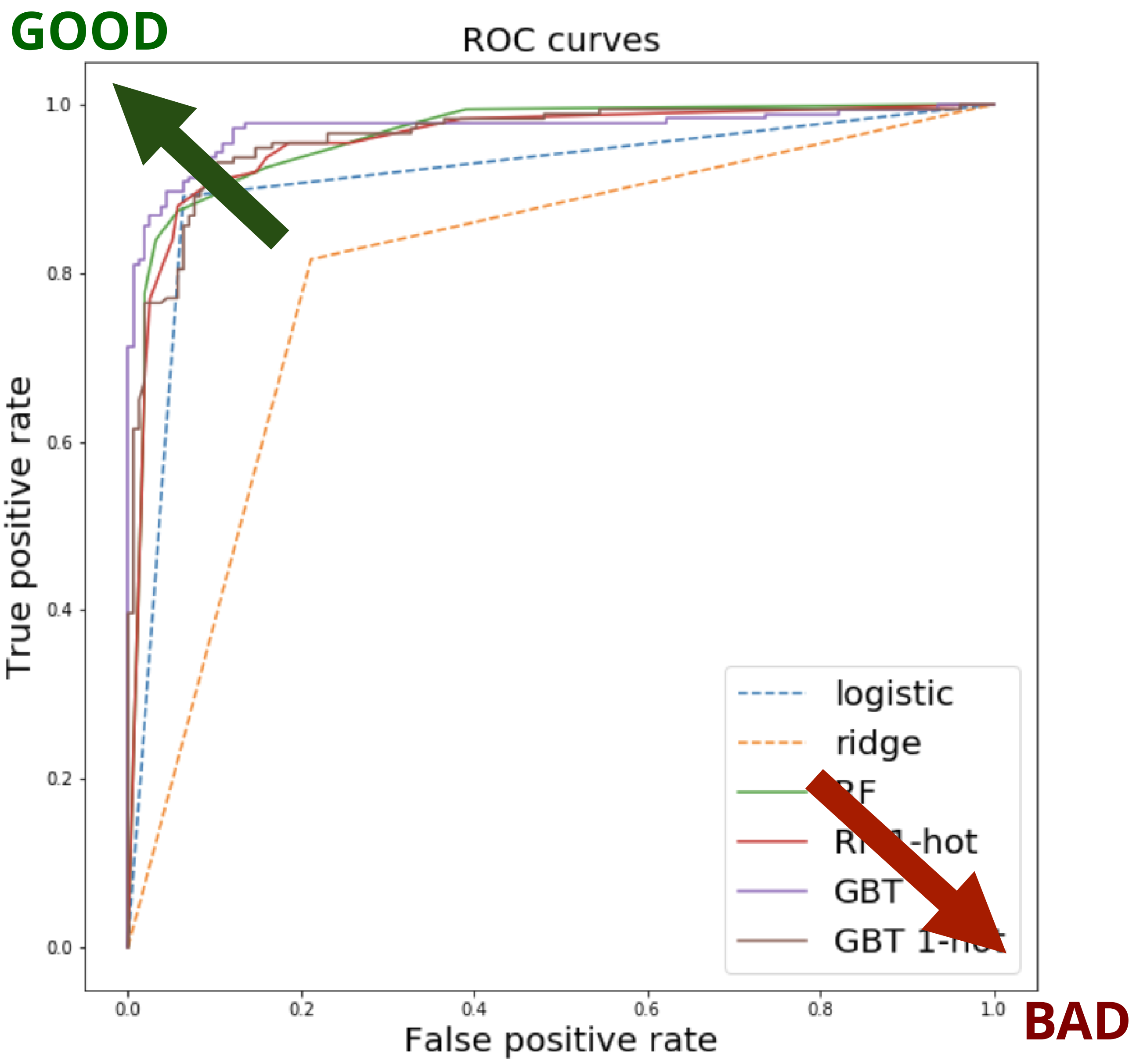
FP = number of false positives

FN = number of false negatives

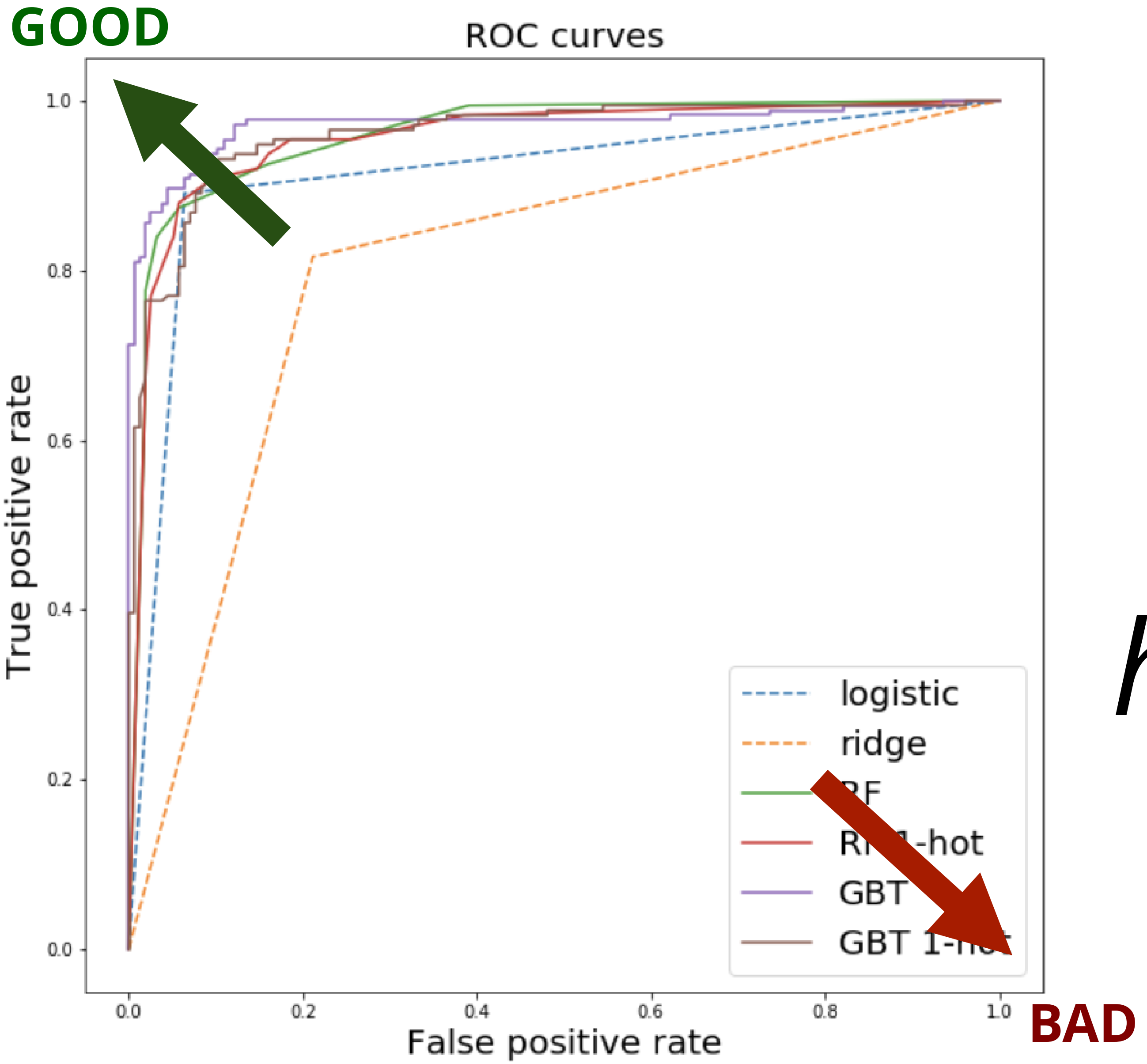
https://en.wikipedia.org/wiki/Precision_and_recall



Receiver operating characteristic



Receiver operating characteristic



tuning by changing *hyperparameters*

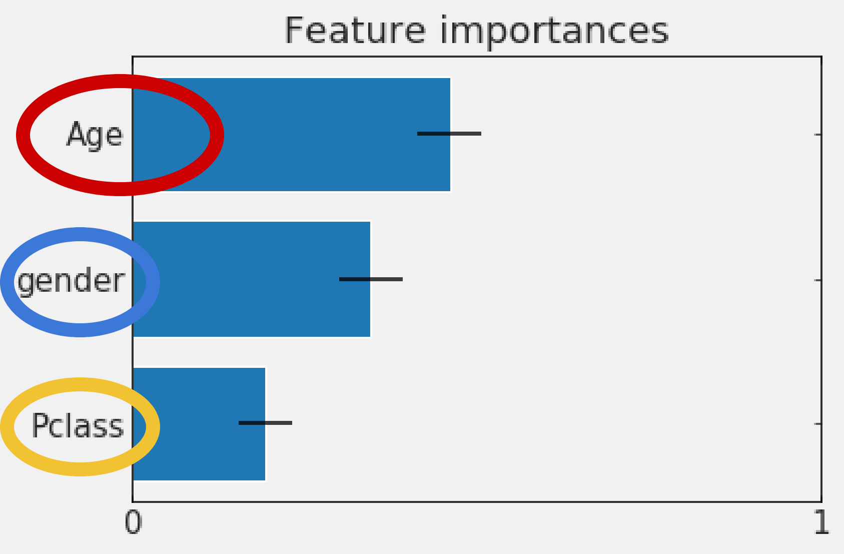
Receiver operating characteristic

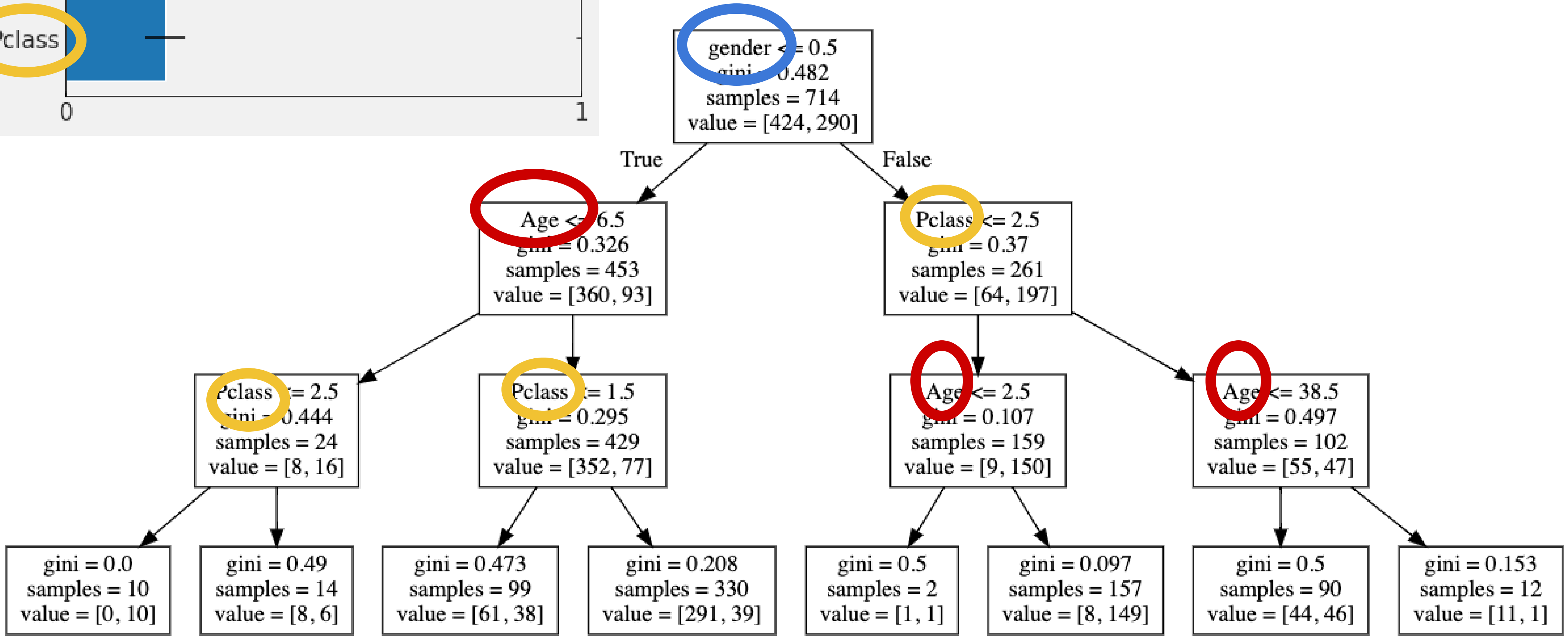
5 Extraction of features

feature importance

In principle CART methods are interpretable

you can measure the influence that each
feature has on the decision : feature importance





https://github.com/fedhere/DSPS/blob/master/lab9/titanictree.ipynb

feature importance

In principle CART methods are interpretable

you can measure the influence that each feature has on the decision : feature importance

In practice the interpretation is complicated by covariance of features

6

encoding categorical variables

Categorical Variable

variable that can take a finite number of values.

spicies	age	weight
dog	7	32.3
bird	1	0.3
cat	3	8.1

Categorical Variable

variable that can take a finite number of values.

spicies	**age**	**weight**
dog	7	32.3
bird	1	0.3
cat	3	8.1

continuous

Categorical Variable

variable that can take a finite number of values.

ordinal

spicies	age	weight
dog	7	32.3
bird	1	0.3
cat	3	8.1

continuous

Categorical Variable

variable that can take a finite number of values.

ordinal

categorical

spicies	age	weight
dog	7	32.3
bird	1	0.3
cat	3	8.1

continuous

numerical encoding

change categorical to (integer) numerical

spicies	age	weight
1	7	32.3
2	1	0.3
3	3	8.1

one-hot encoding

change each category to a binary

cat	bird	dog	age	weight
0	0	1	7	32.3
0	1	0	1	0.3
1	0	0	3	8.1

numerical encoding

change categorical to (integer) numerical

spicies	age	weight
1	7	32.3
2	1	0.3
3	3	8.1

implies an order that does not exist

one-hot encoding

change each category to a binary

cat	bird	dog	age	weight
0	0	1	7	32.3
0	1	0	1	0.3
1	0	0	3	8.1

numerical encoding

change categorical to (integer) numerical

spicies	age	weight
1	7	32.3
2	1	0.3
3	3	8.1

implies an order that does not exist

one-hot encoding

change each category to a binary

cat	bird	dog	age	weight
0	0	1	7	32.3
0	1	0	1	0.3
1	0	0	3	8.1

ignores covariance between features
increases the dimensionality

numerical encoding

change categorical to (integer) numerical

spicies	age	weight
1	7	32.3
2	1	0.3
3	3	8.1

implies an order that does not exist

one-hot encoding

Definitely Preferred!

change each category to a binary

cat	bird	dog	age	weight
0	0	1	7	32.3
0	1	0	1	0.3
1	0	0	3	8.1

ignores covariance between features

increases the dimensionality

problematic if you are interested in feature importance

numerical encoding one-hot encoding

https://github.com/fedhere/MLPNS_FBianco/blob/main/OHE/locationLocationLocation.ipynb

CART

http://what-when-how.com/artificial-intelligence/decision-tree-applications-for-data-modelling-artificial-intelligence/

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/

resources

Feature extractions from time series

Distributed and parallel time series feature extraction for industrial big data applications

Maximilian Christ a , Andreas W. Kempa-Liehrb,c, Michael Fein

https://arxiv.org/pdf/1610.07717.pdf

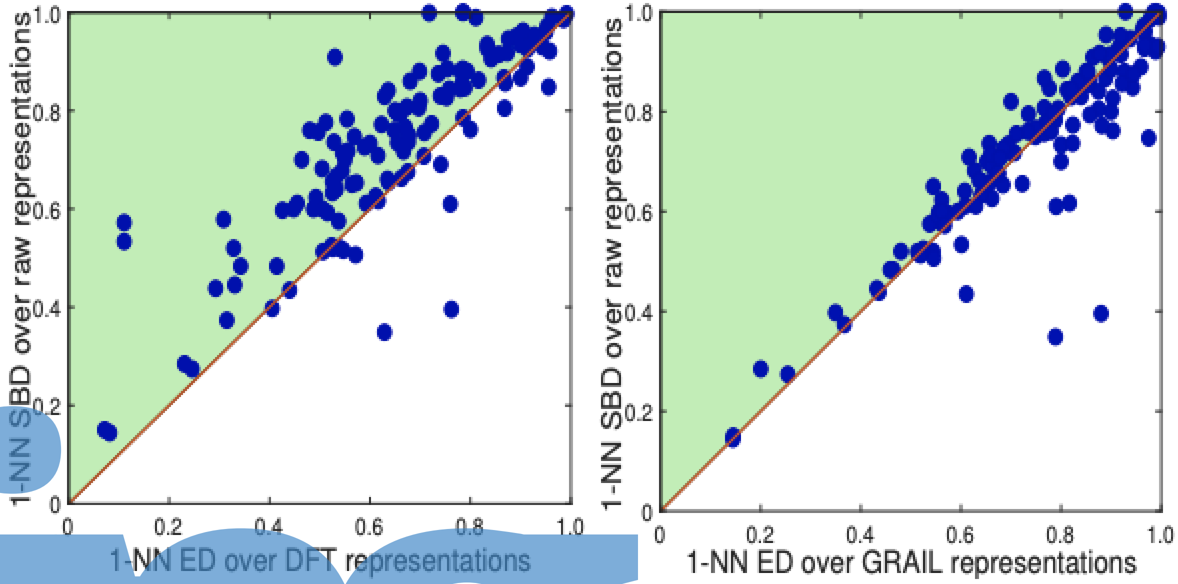
TL;DR:

https://towardsdatascience.com/time-series-feature-extraction-for-industrial-big-data-iiot-applications-5243c84aaf0e

resources

http://www.vldb.org/pvldb/vol12/p1762-paparrizos.pdf

Reading



(a) Raw vs. DFT (b) Raw vs. GRAIL

Figure 1: Comparison of classification accuracies across 128 datasets using time series in their raw representations against compact representations of size 20 computed with DFT and GRAIL. Circles over the diagonal indicate datasets over which raw representations outperform low-dimensional representations.

Kaggle PLAsTICc challenge

Homework