data science for (physical) scientists VIII

Tree methods

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

this slide deck: http://bit.ly/dspsVIII

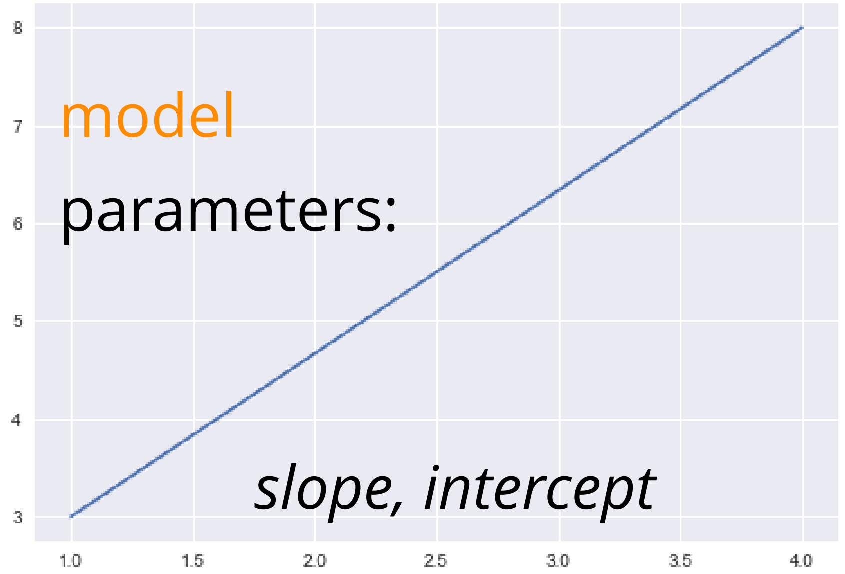
- Machine Learning basic concepts
  - interpretability
  - parameters vs hyperparameters
  - supervised/unsupervised
- Tree methods
  - single trees
  - hyperparameters
  - weaknesses
- Tree ensembles
- Feature importance

recap

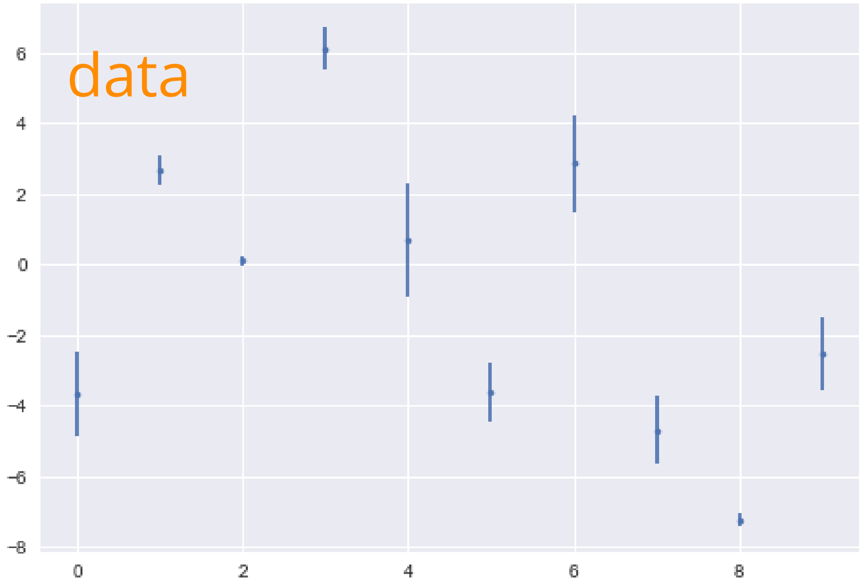
what is machine learning

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. Arthur Samuel, 1959



ML: any model with parameters learnt from the data



what is machine learning?

supervised learning

classification
prediction
feature selection

unsupervised learning

understanding structure
organizing/compressing data
anomaly detection
dimensionality reduction

what is machine learning?

supervised learning

k-Nearest Neighbors
Regression
Support Vector Machines
Classification/Regression Trees
Neural networks

classification
prediction
feature selection

unsupervised learning

understanding structure
organizing/compressing data
anomaly detection
dimensionality reduction

clustering
PCA
Apriori

general ML points

used to:

understand structure of feature space

classify based on examples,

regression (classification with infinitely small classes)

general ML points

should be interpretable: why?

ethical implication,

prective policing,

selection of conference participants.

general ML points

ML model have *parameters* and *hyperparameters*

parameters: the model optimizes based on the data

hyperparameters: chosen by the model author, could be based on domain knowledge, other data, guessed (?!). e.g. the shape of the polynomial

general ML points

should be interpretable: why?

ethical implication,

prective policing,

selection of conference participants.

connect to causality

why themodel made a choice?

which feature mattered

classification vs clustering

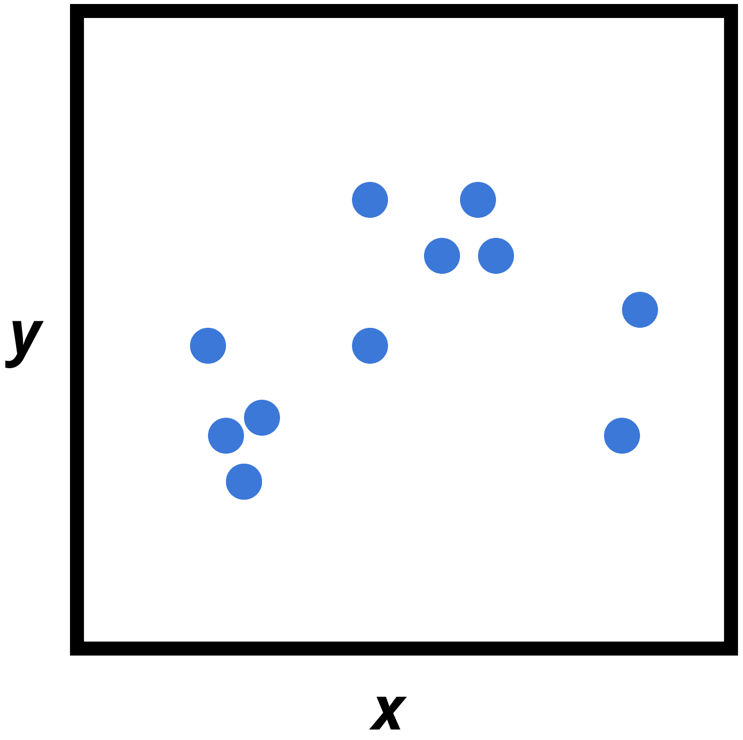
1

clustering vs classifying

unsupervised

observed features:

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

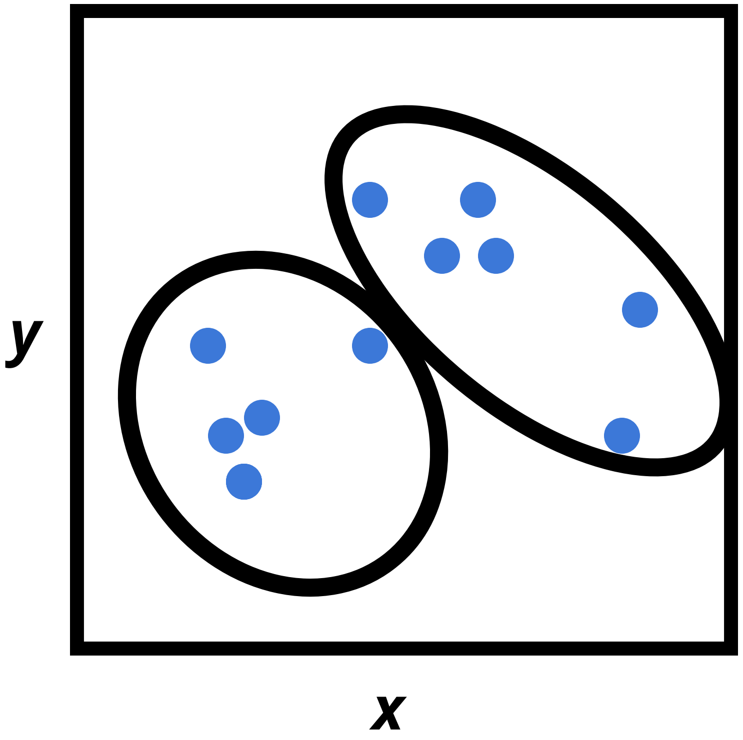


clustering vs classifying

unsupervised

observed **features:**

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



clustering vs classifying

unsupervised

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

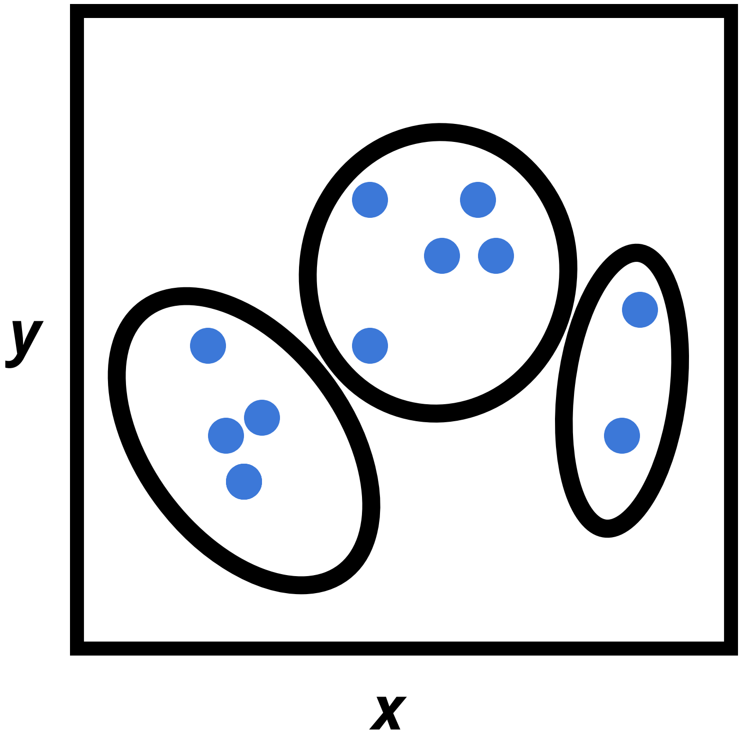
separated into

maximally homogeneous

maximally distinguishable groups

observed features:

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



models typically return a cluster label by object

clustering vs classifying

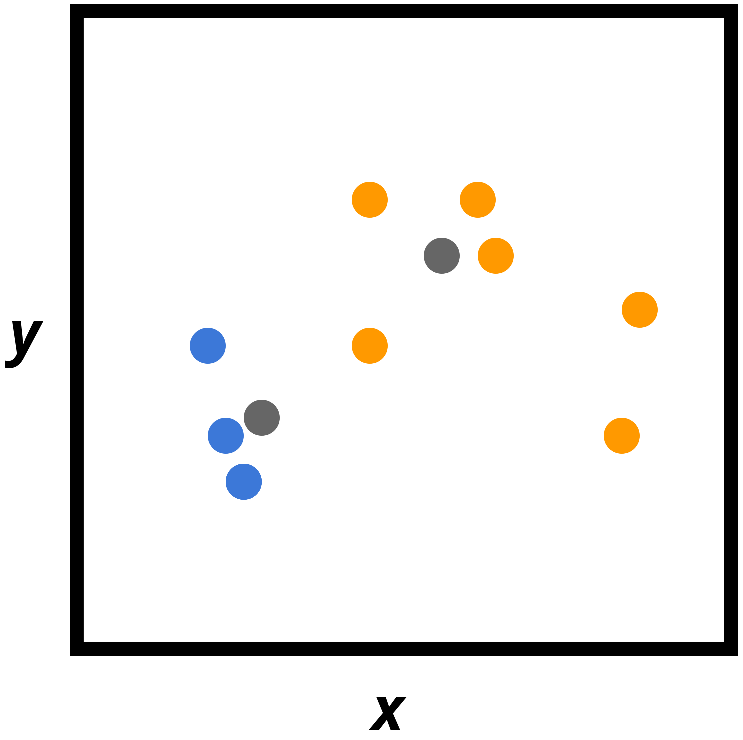
unsupervised *supervised*

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:

$(\overrightarrow{color})$

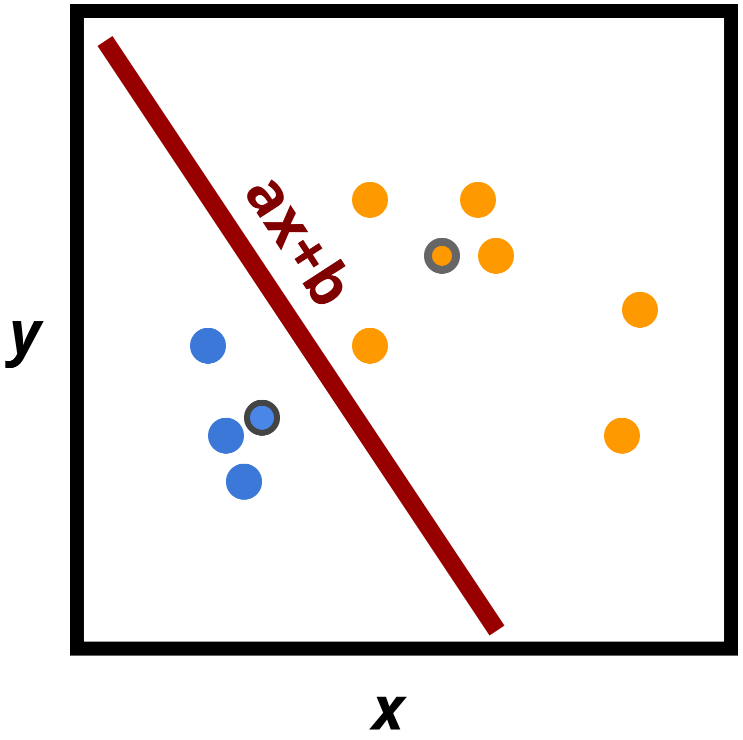
models typically return a partition of the space

clustering vs classifying

unsupervised *supervised*

observed features:

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



target features:

$\overrightarrow{(color)}$

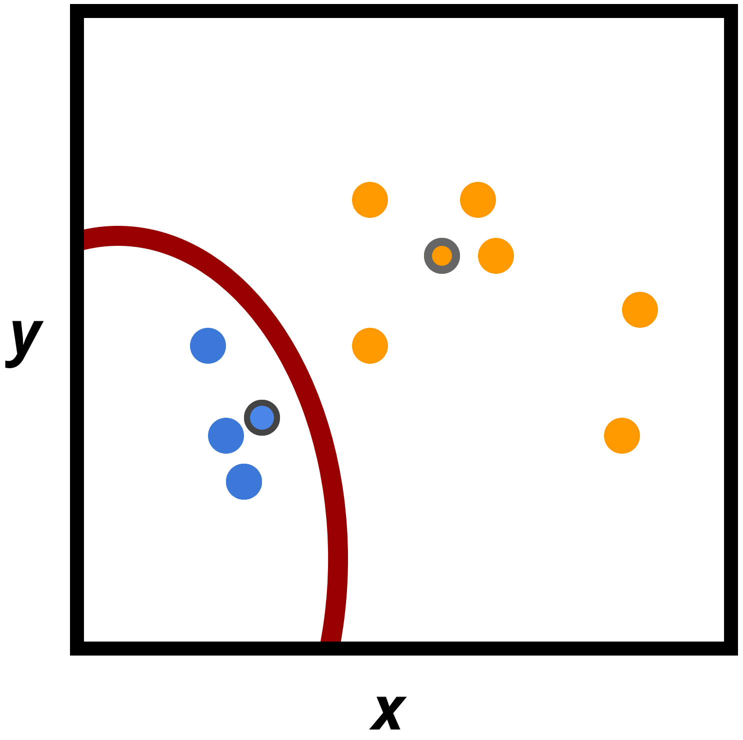
```
if y >= a*x + b :
        return blue
else:
        return orange
```

clustering vs classifying

unsupervised *supervised*

observed **features:**

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



target **features:**

(color)

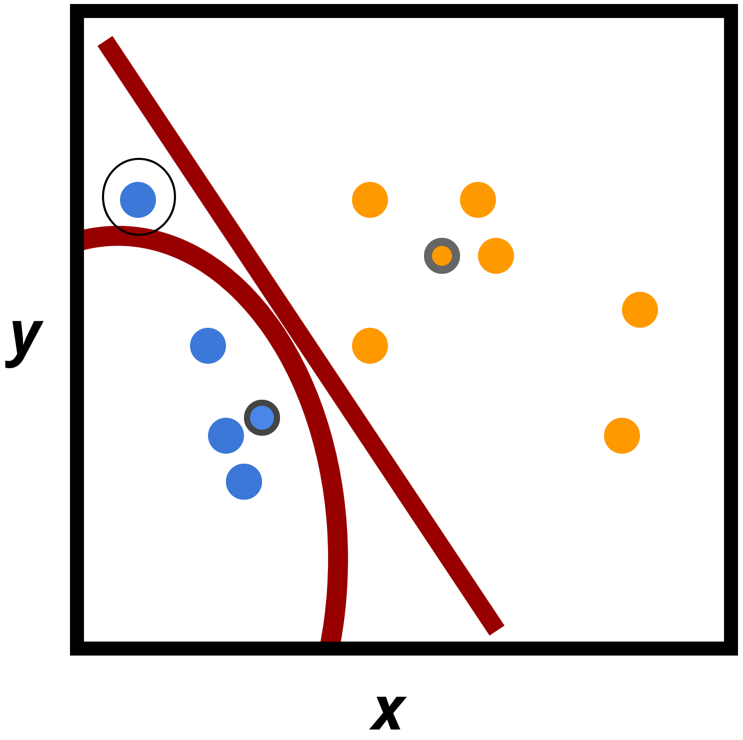
```
if x**2 + y**2 >= (x-a)**2 + (y-b)**2 :
        return blue
else:
        return orange
```

clustering vs classifying

unsupervised *supervised*

observed features:

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



target features:

$(\overrightarrow{color})$

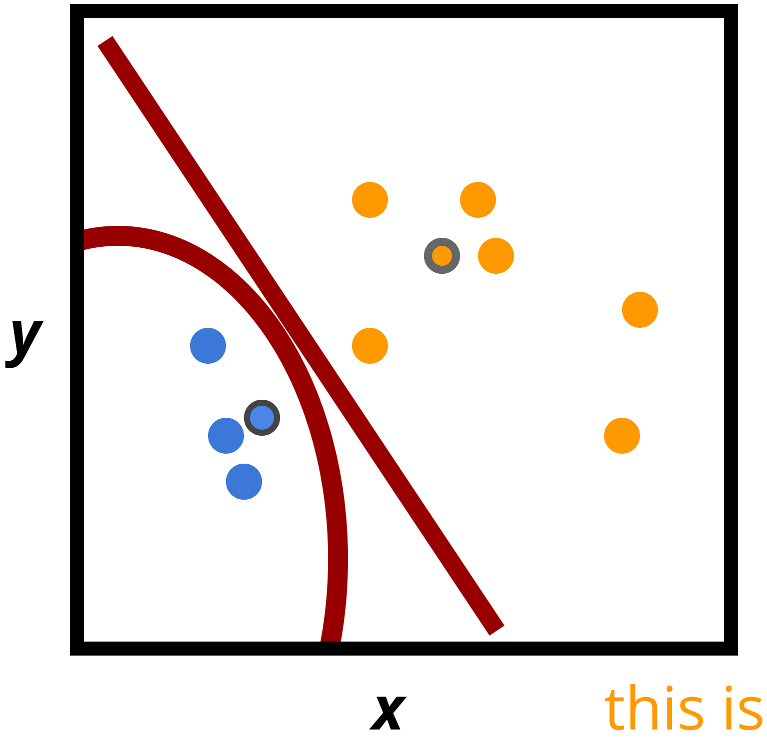
```
if x**2 + y**2 >= (x-a)**2 + (y-b)**2 :
        return blue
else:
        return orange
```

clustering vs classifying

unsupervised *supervised*

observed **features:**

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



target **features:**

$(\overrightarrow{color})$

this is a solution SVM would provide:

Support Vector Machine

supervised ML: classification

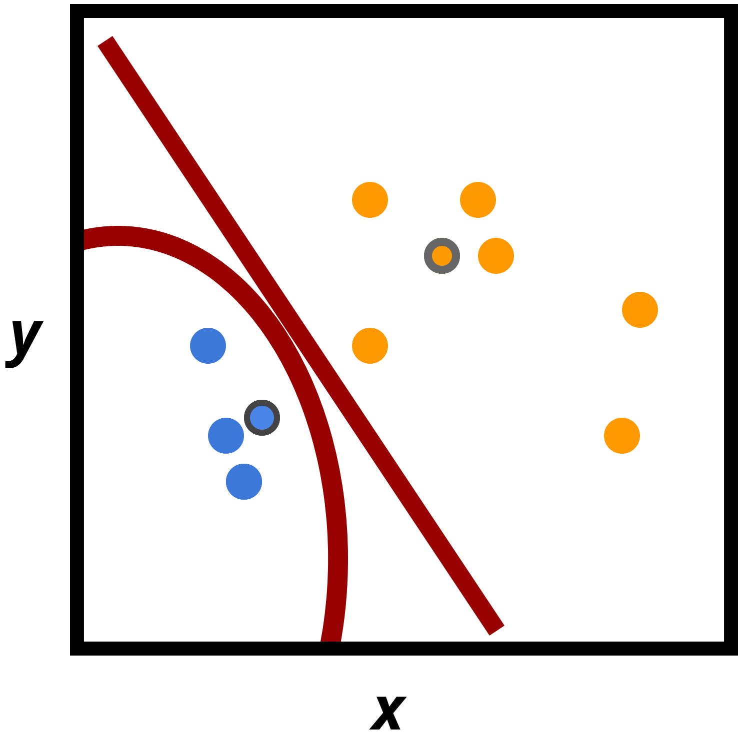
A subset of variables has class labels.
Guess the label for the other variables

Support Vector Machine:

finds a hyperplane that partitions the space

observed **features:**

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



target **features:**

$(\overrightarrow{color})$

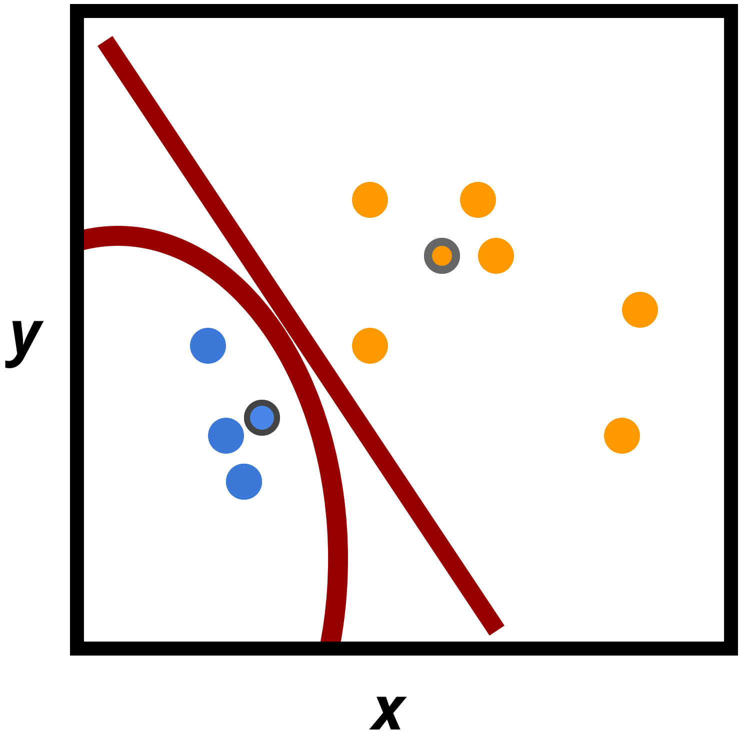
supervised ML: classification

A subset of variables has class labels.

Guess the label for the other variables

Support Vector Machine:

finds a hyperplane that partitions the space



2d hyperplane: line (curve)

3d hyperplane: surface

4d hyperplane: volume

...

observed features:

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

target features:

$(\overrightarrow{color})$

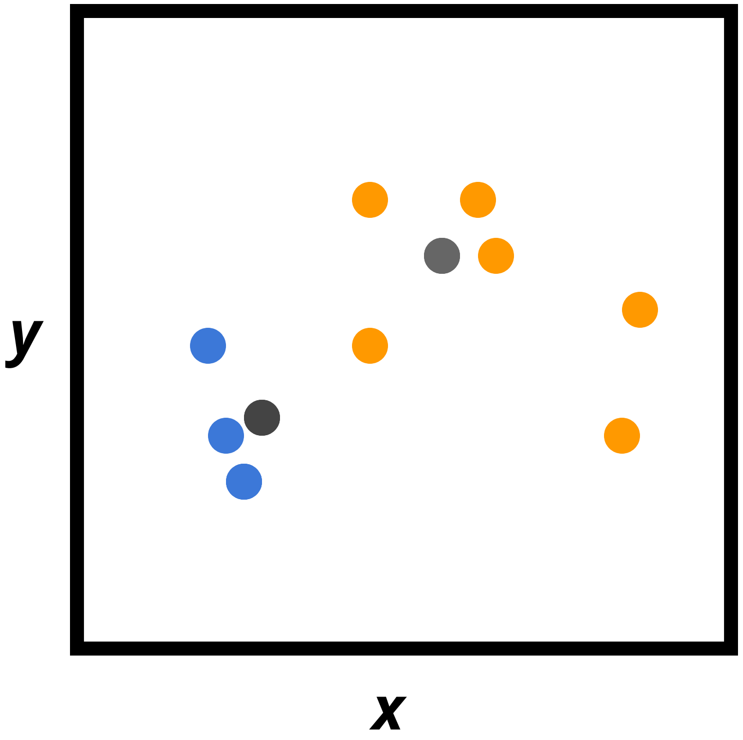
supervised ML: classification

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:
$(\overrightarrow{color})$

supervised ML: classification

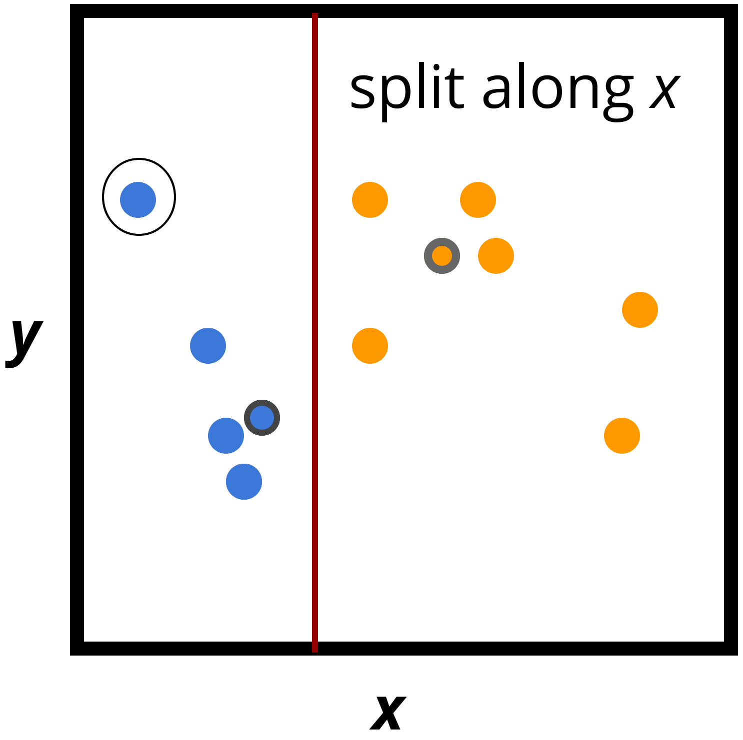
A subset of variables has class labels.
Guess the label for the other variables

Tree Methods

split spaces along each axis separately

observed features:

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



target features:

$\overrightarrow{(color)}$

```
if x~<=~a :
        return blue
else:
        return orange
```

supervised ML: classification

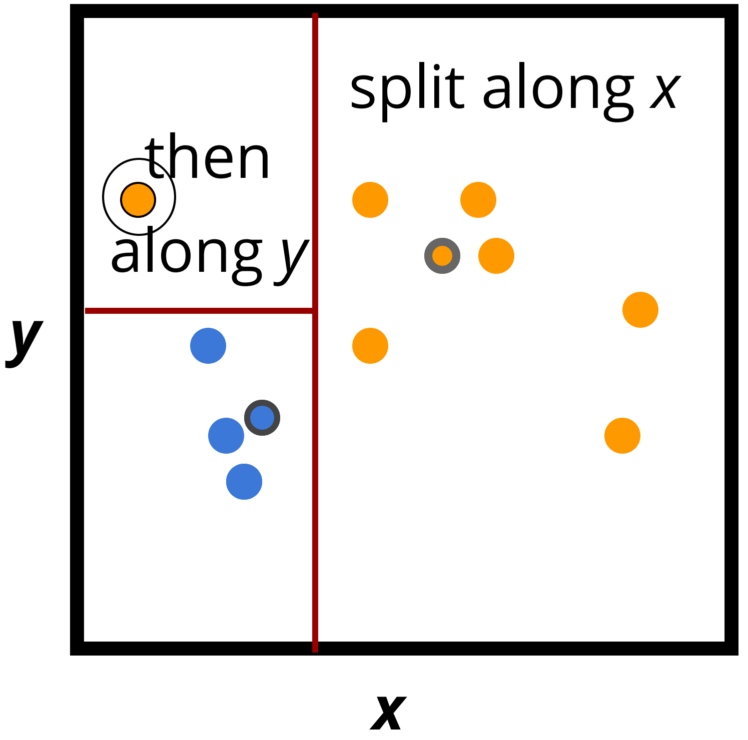
A subset of variables has class labels.
Guess the label for the other variables

Tree Methods

split spaces along each axis separately

observed features:

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



target features:

(color)

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

Tree Methods
supervised learning method
partitions feature space along each feature separately

The good

- Non-Parametric
- White-box: can be easily interpreted
- Works with any feature type and mixed feature types
- Works with missing data
- Robust to outliers

The bad

- High variability (-> use ensamble methods)
- Tendency to overfit
- (not really easily interpretable after all...)

single tree

1

Application:

a robot to predict surviving the Titanic

(Kaggle)

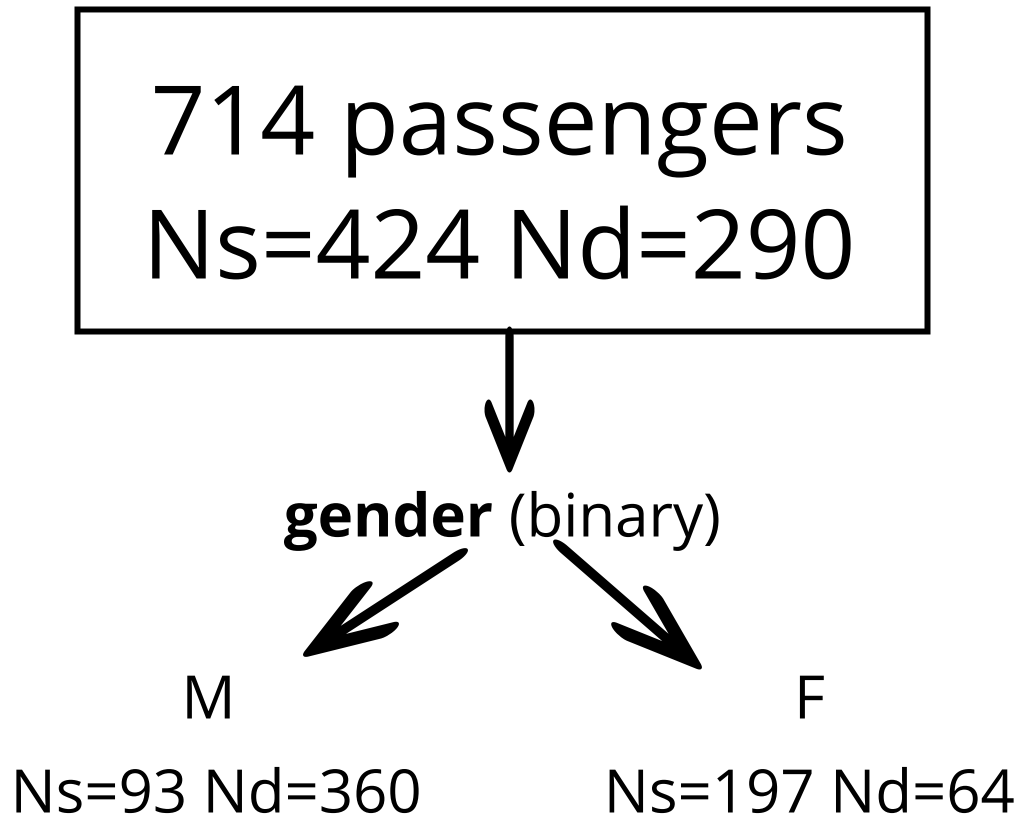
https://www.kaggle.com/c/titanic

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

F
Ns=197 Nd=64

optimize over purity:

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

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)

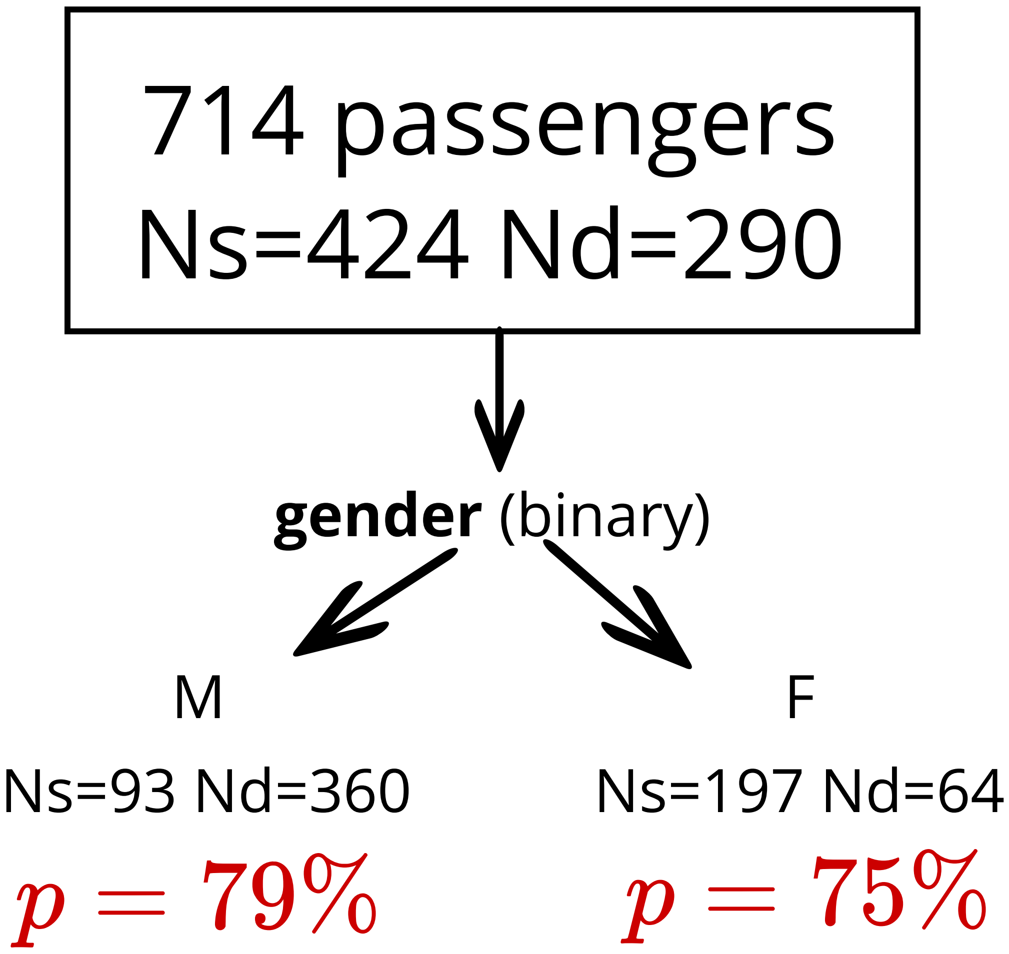
https://www.kaggle.com/c/titanic

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)



optimize over purity:

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

Application:

a robot to predict surviving the Titanic

(Kaggle)

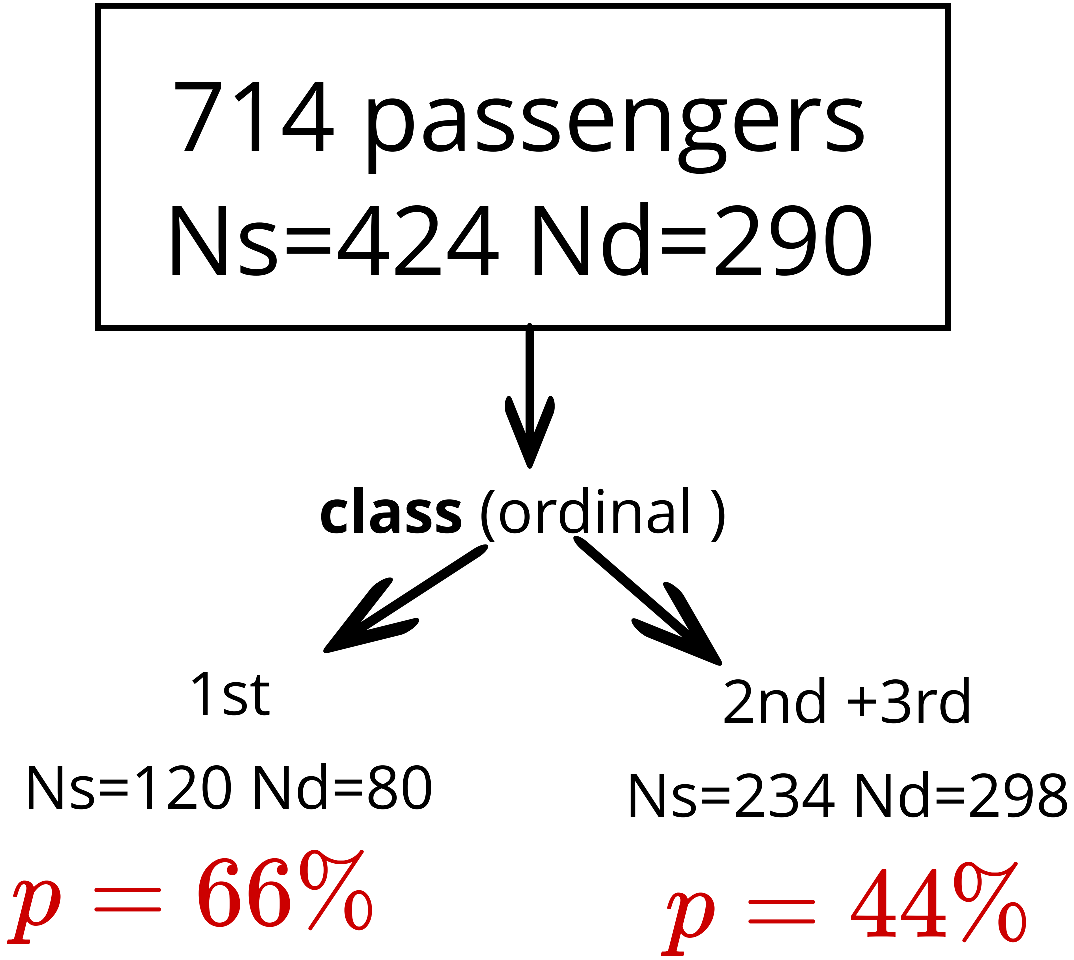
https://www.kaggle.com/c/titanic

features:

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

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

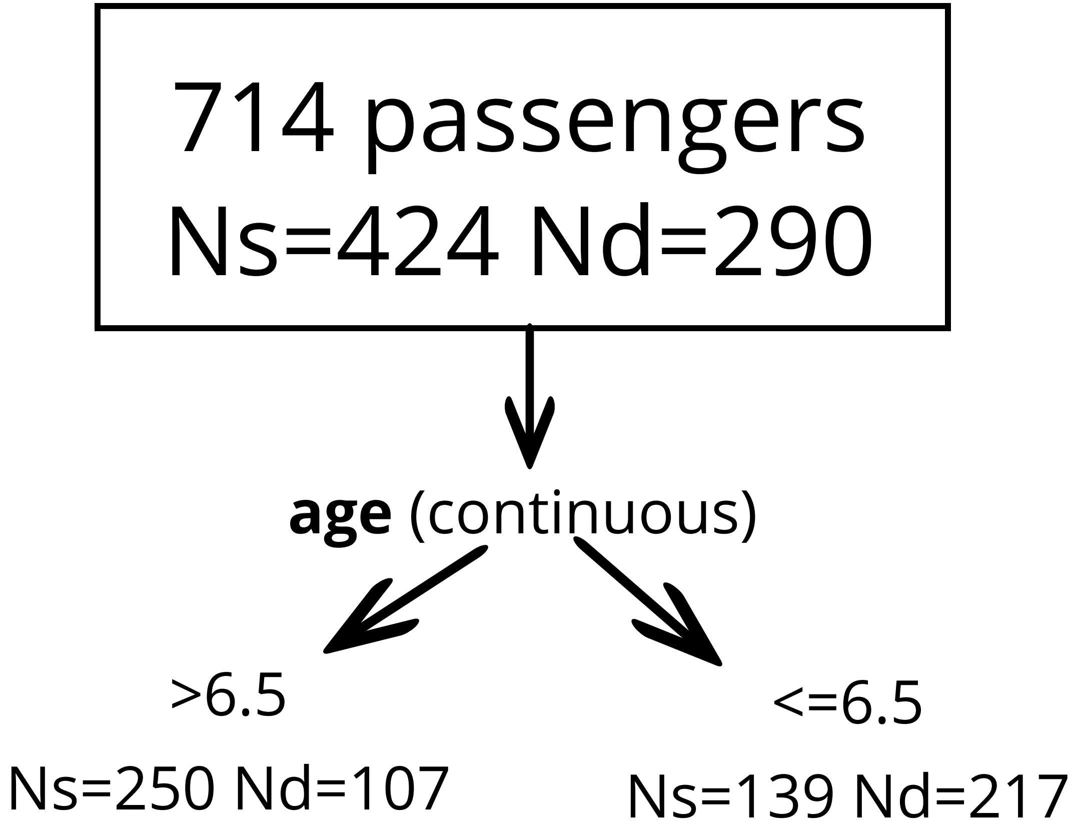
https://www.kaggle.com/c/titanic

features:

- gender 79%|75%
- ticket class 66% | 44%
- age 66% | 61%

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

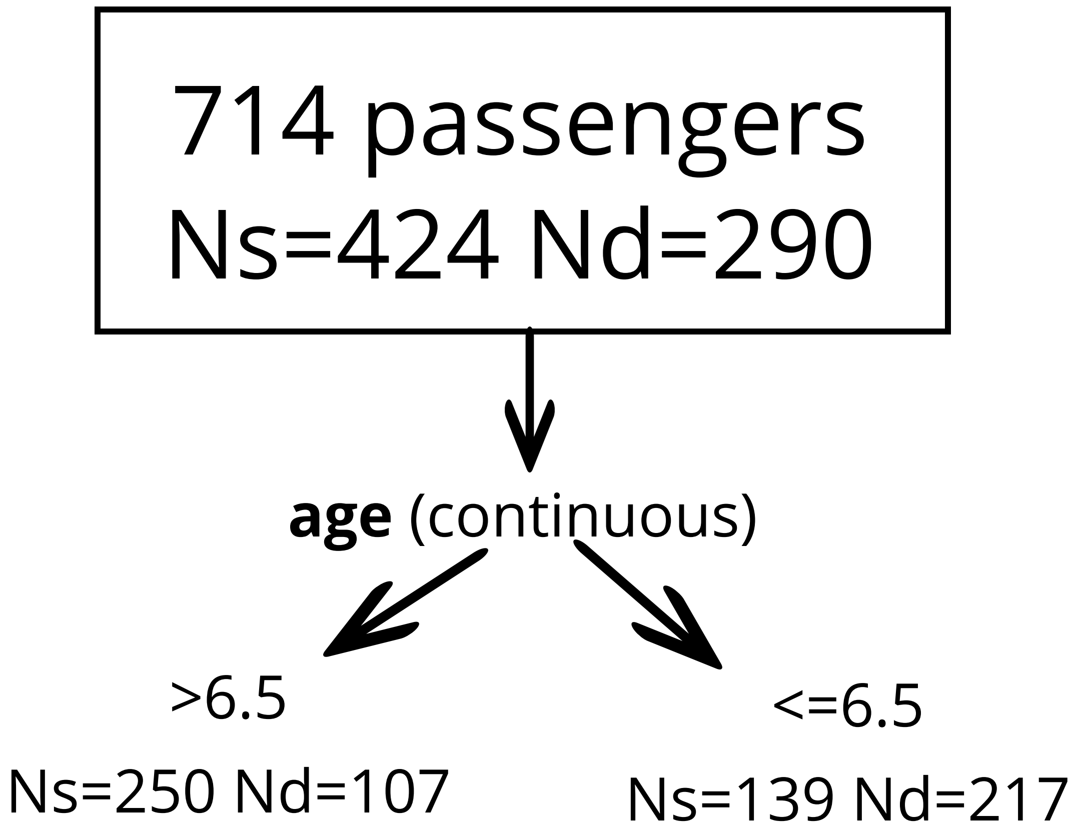
https://www.kaggle.com/c/titanic

features:

- gender 79%|75%
- ticket class 66% | 44%
- age 66% | 61%

target variable:

-> survival (y/n)

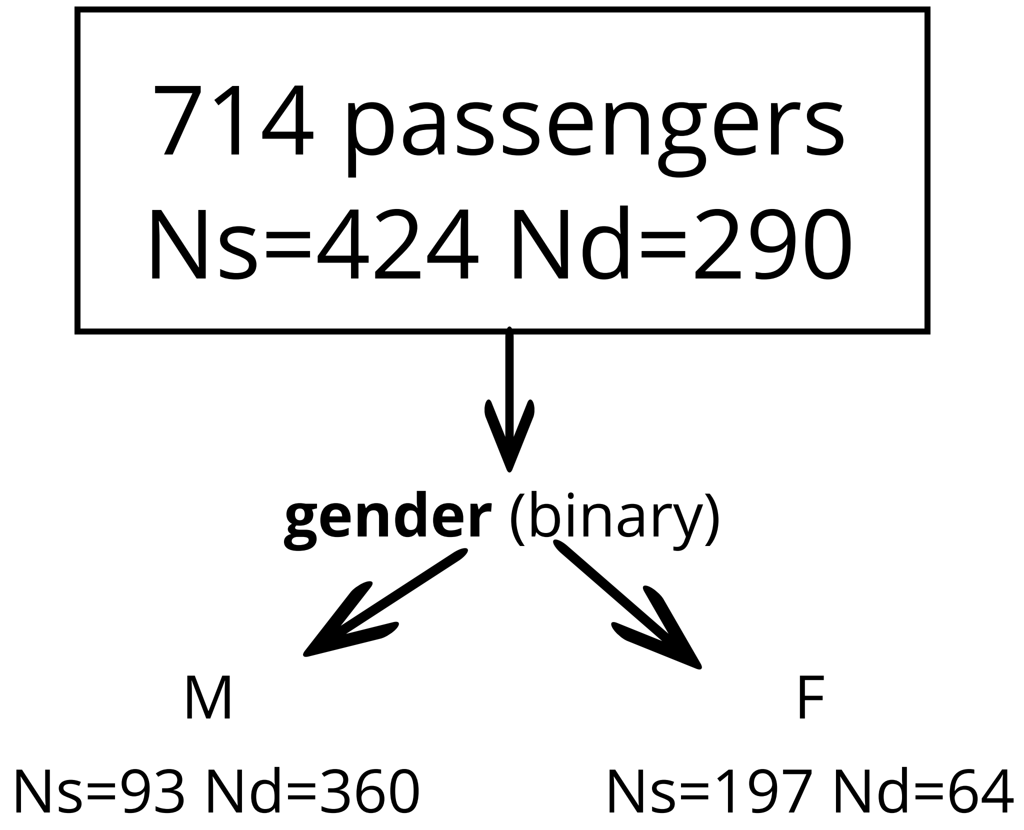


Application:

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:

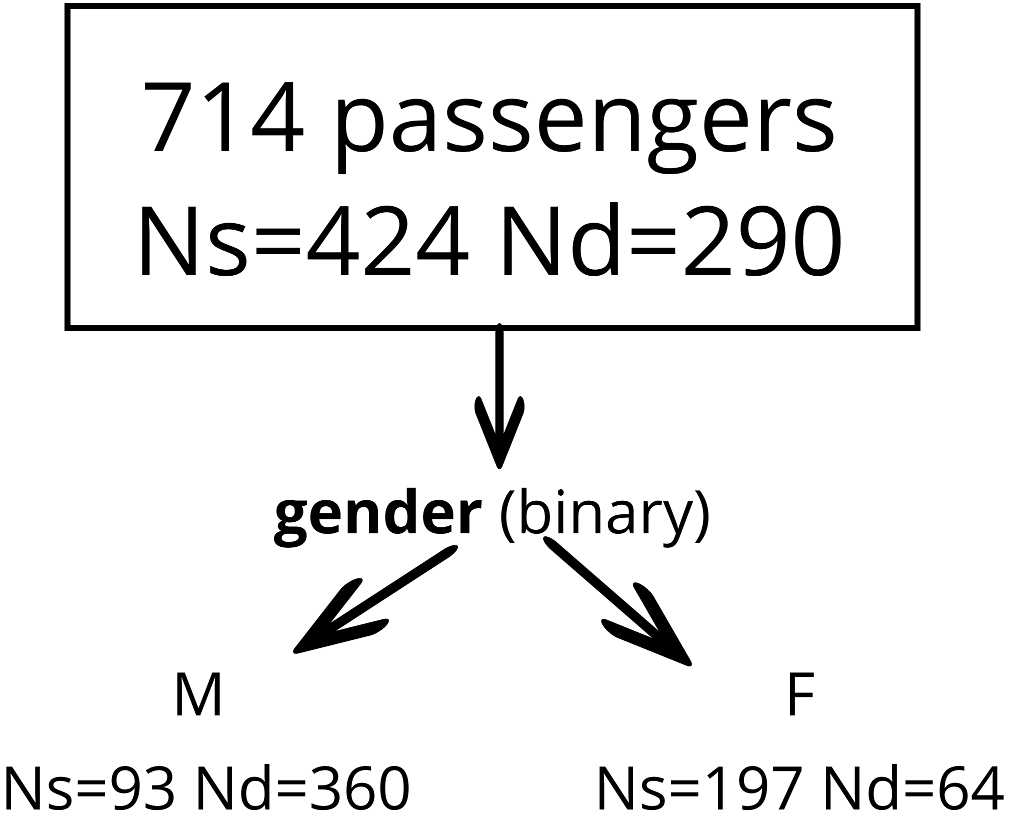
-> survival (y/n)

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)

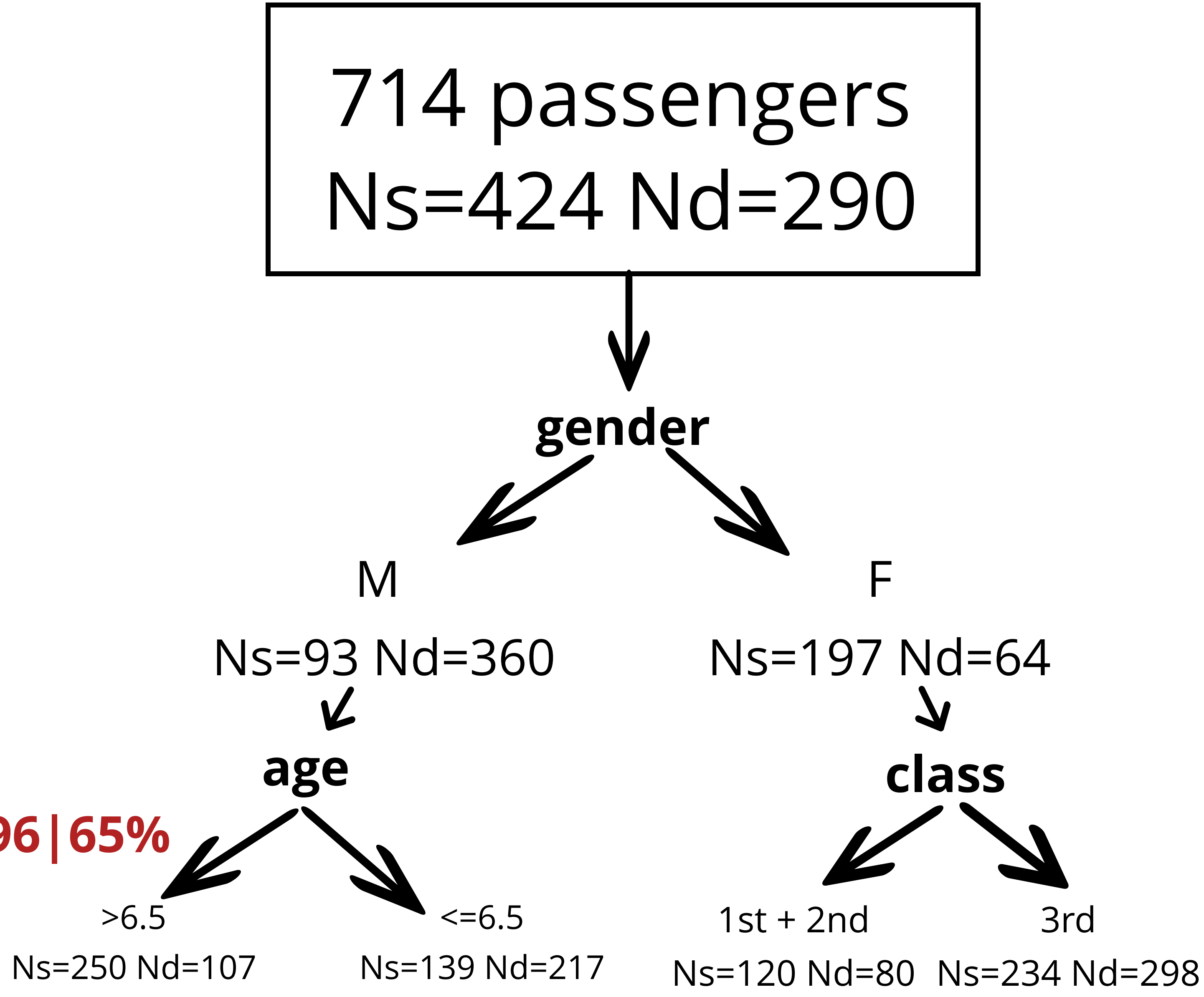
https://www.kaggle.com/c/titanic

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)

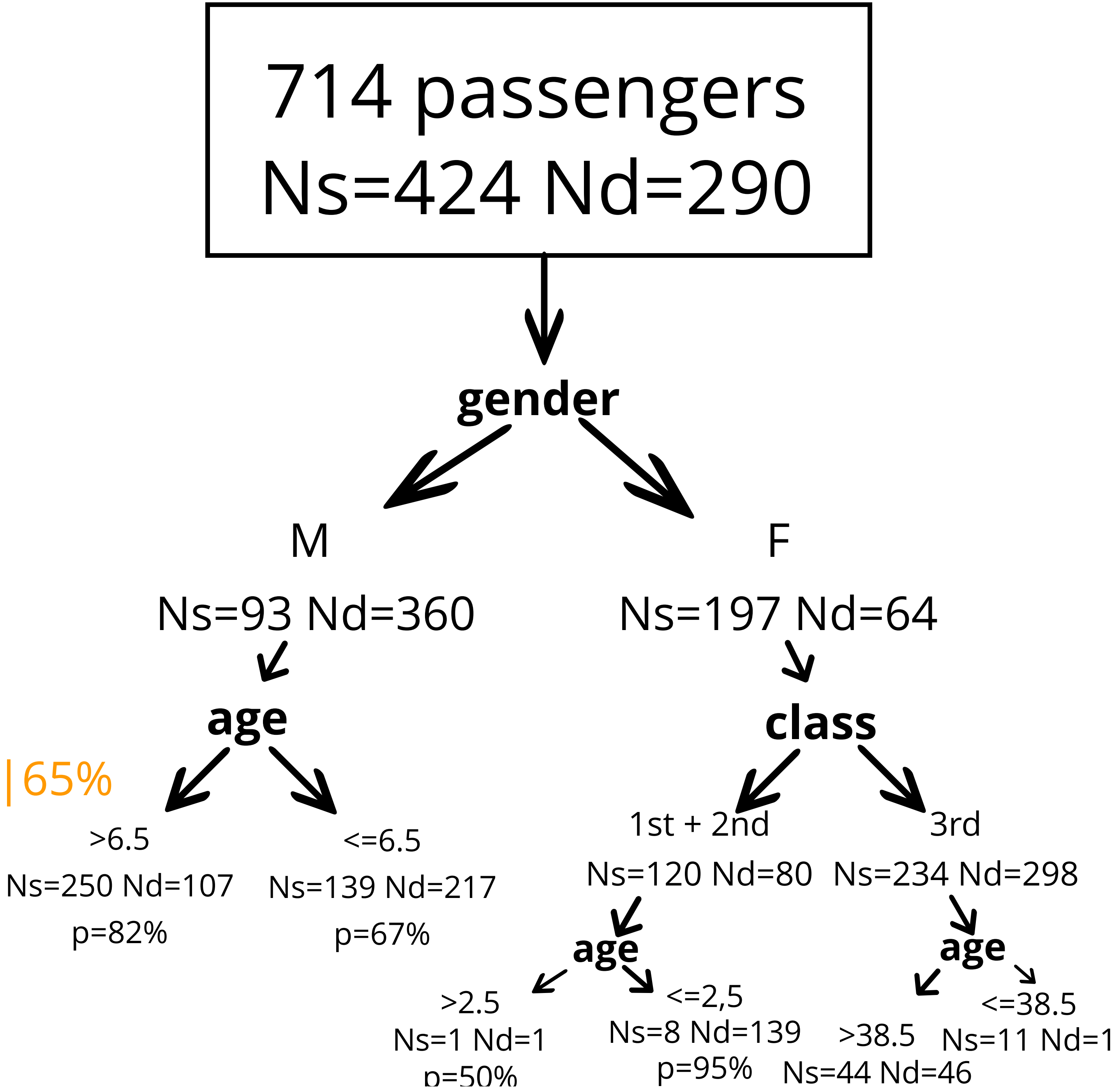
https://www.kaggle.com/c/titanic

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)

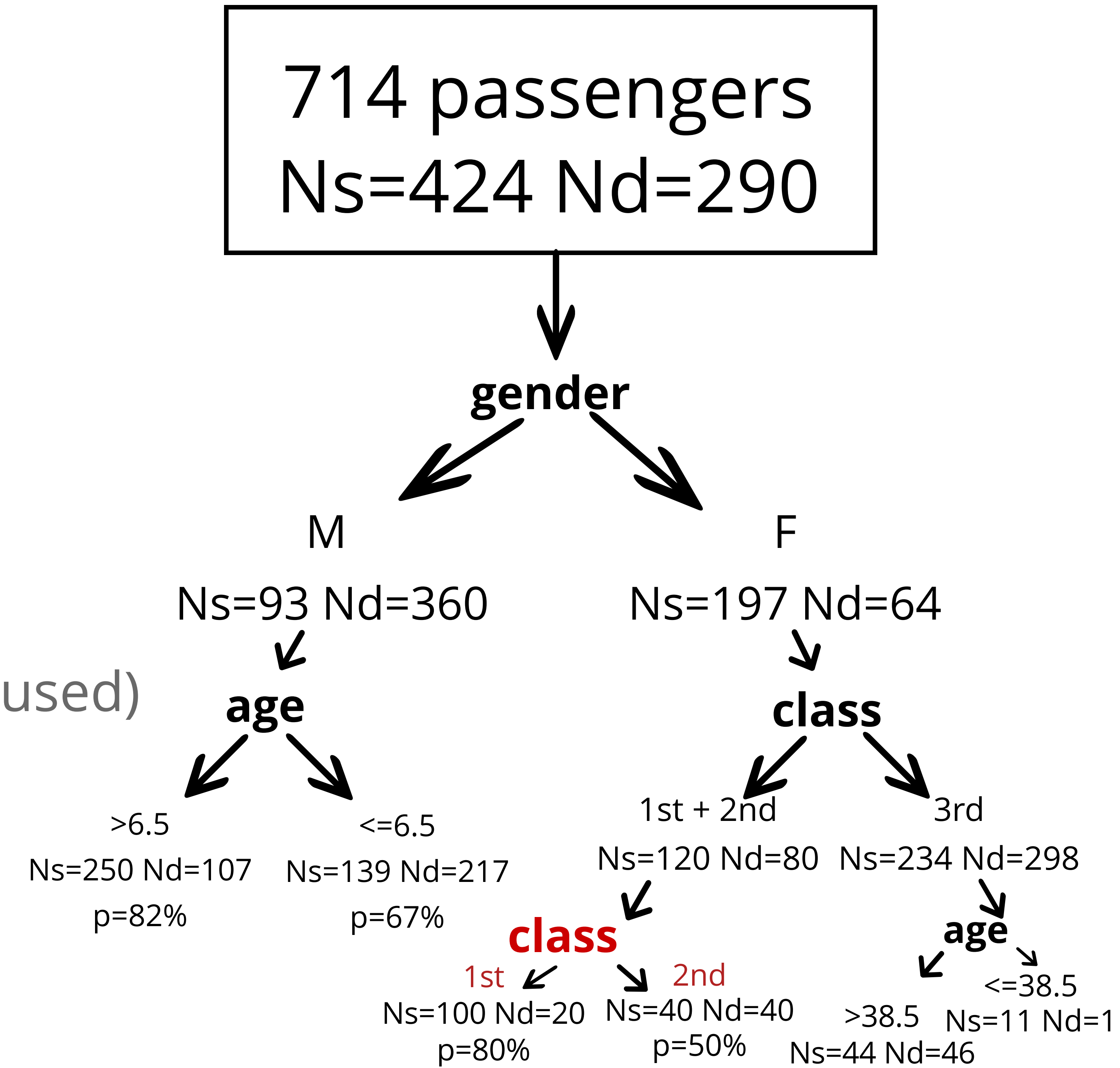
https://www.kaggle.com/c/titanic

features:

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

target variable:

-> survival (y/n)

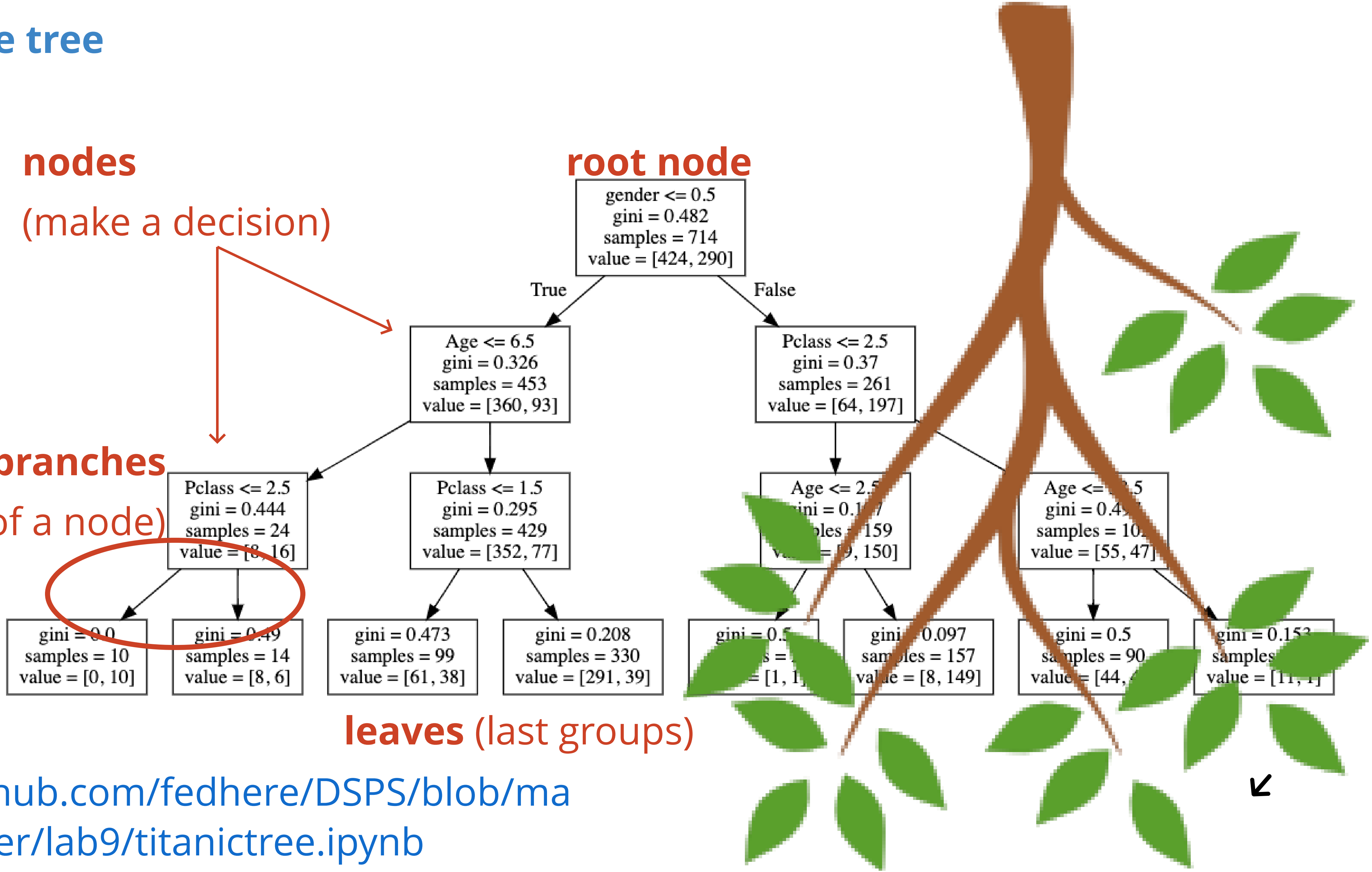


A single tree

nodes

(make a decision)

root node



branches

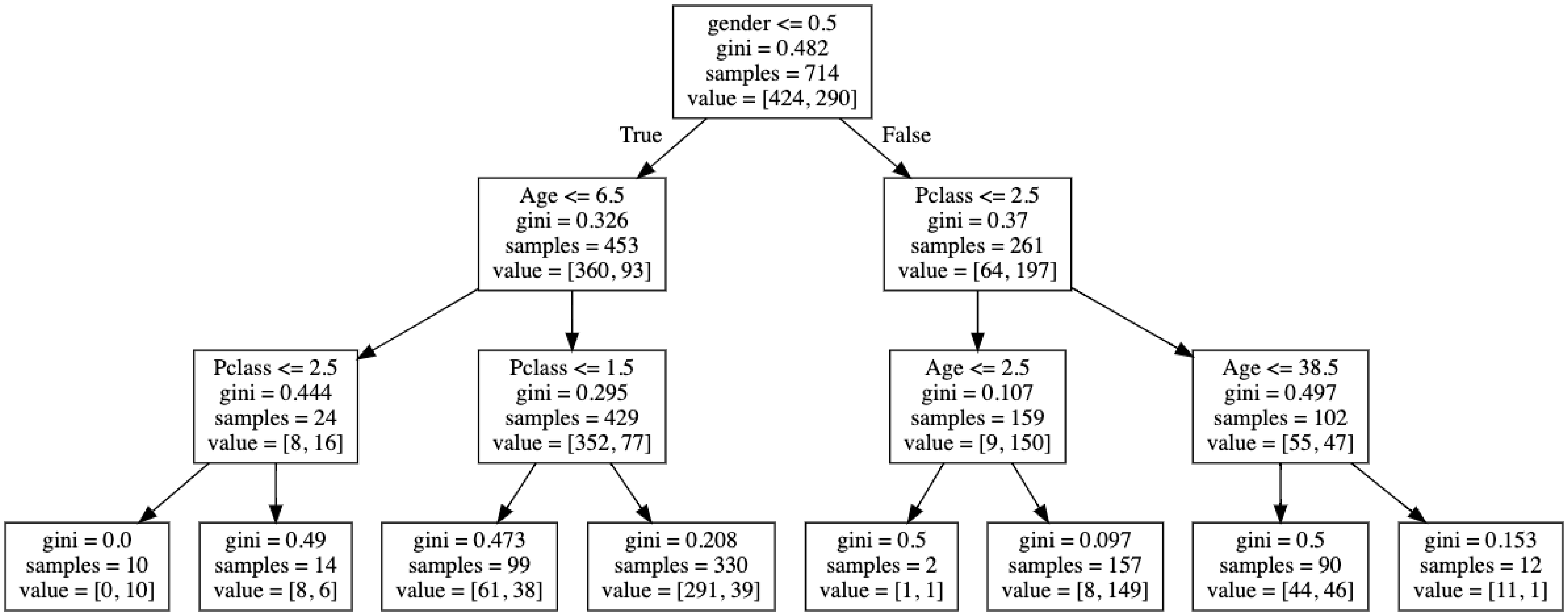
(split off of a node)

leaves (last groups)

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

A single tree

this visualization is called a "dendrogram"



tree hyperparameters

2

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.

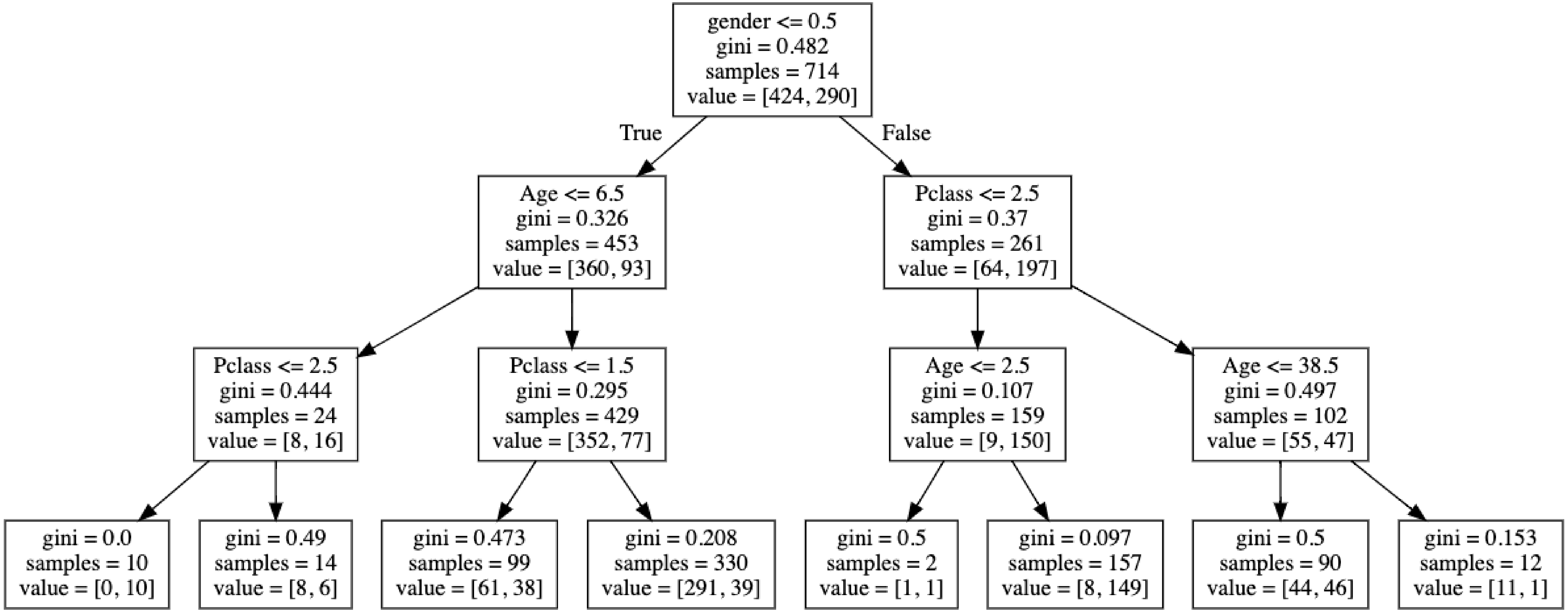
gini impurity

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

information gain (entropy)

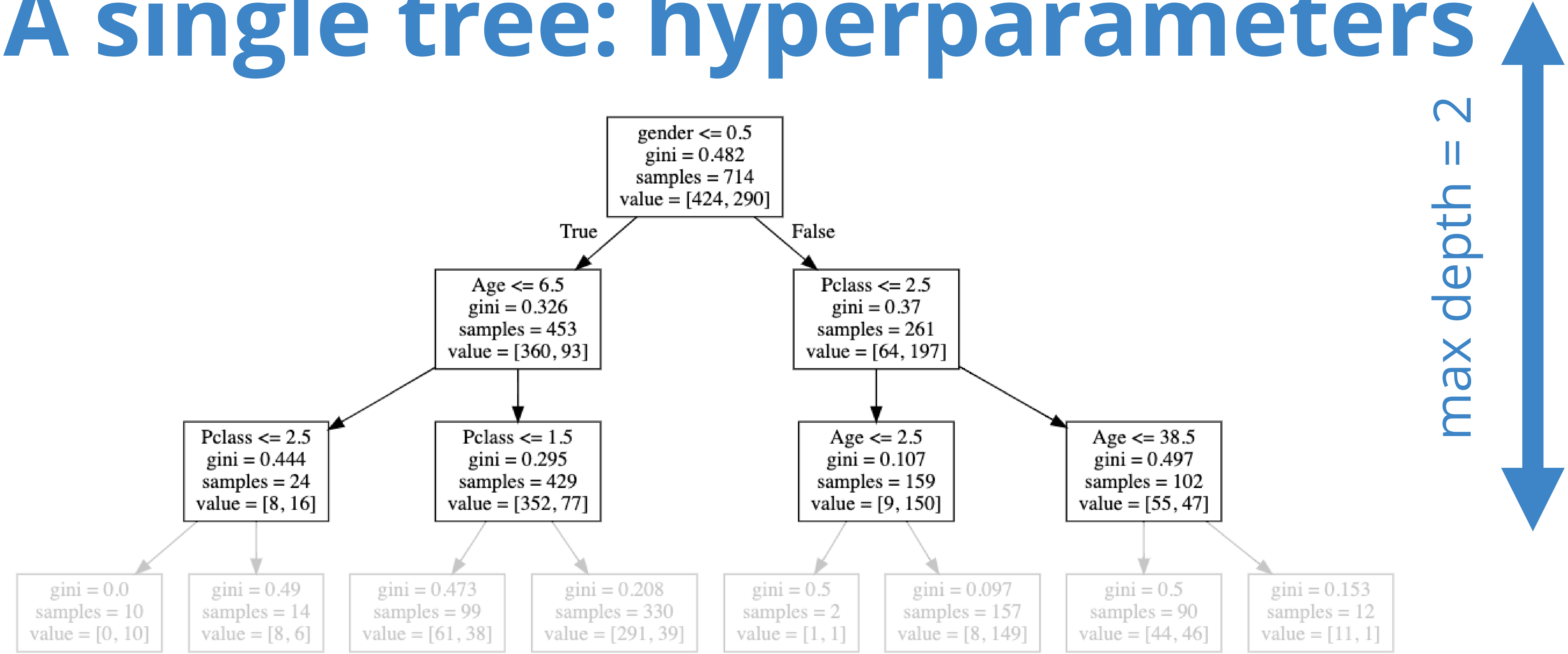
$$\mathrm{H}(T) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

A single tree: hyperparameters

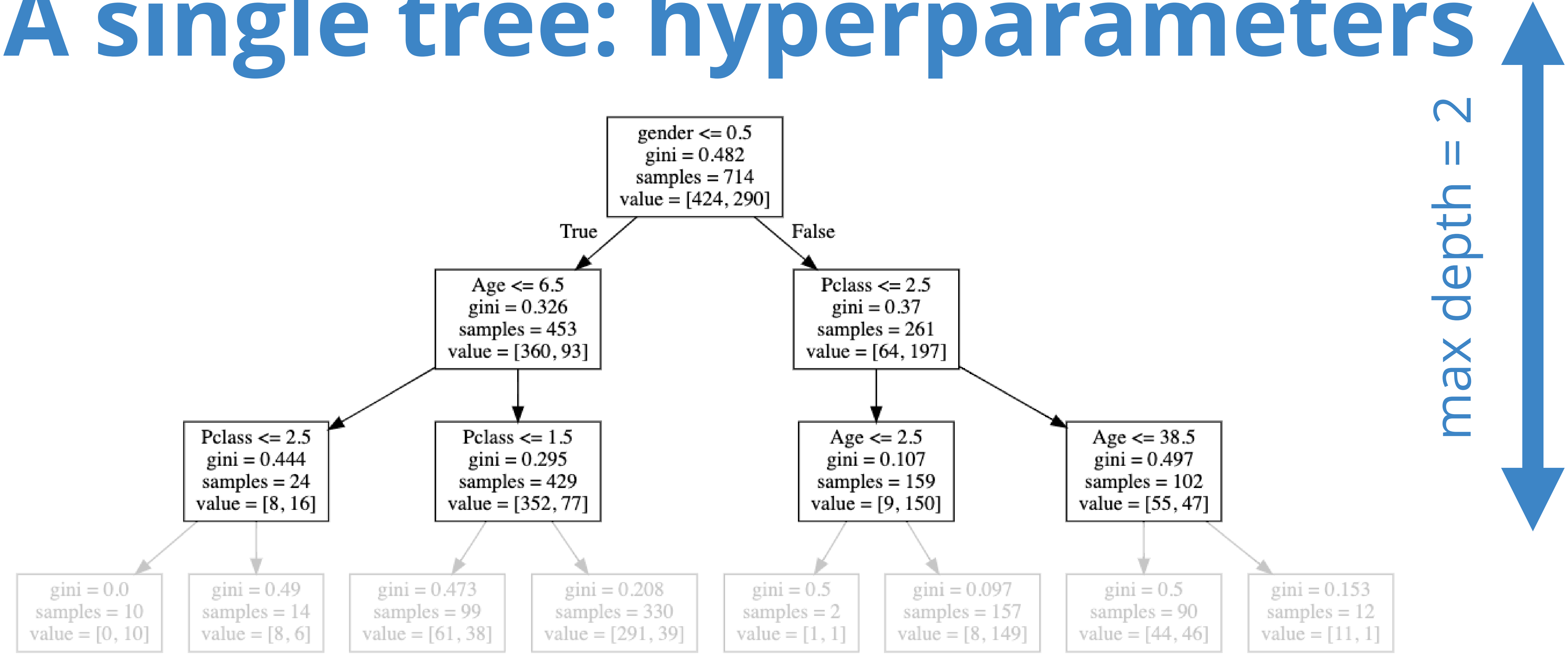


depth

A single tree: hyperparameters

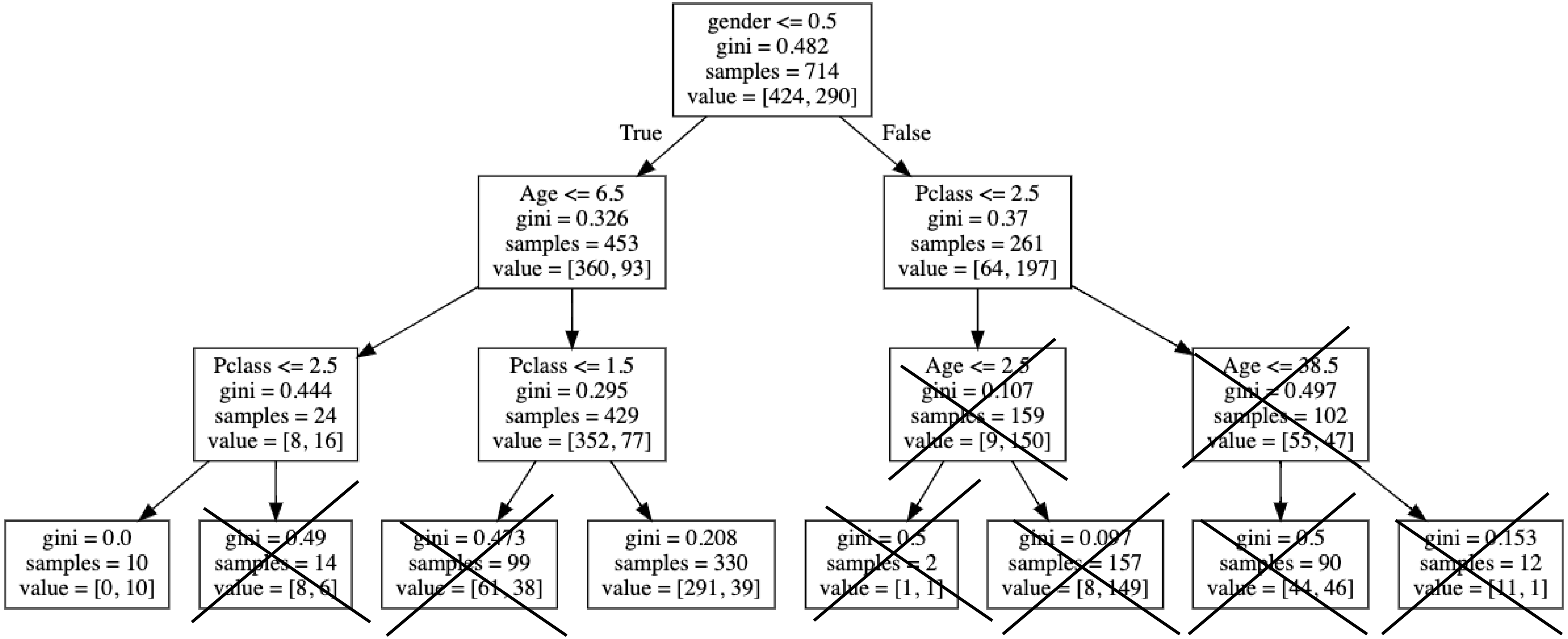


A single tree: hyperparameters



PREVENTS OVERGFITTING

A single tree: hyperparameters

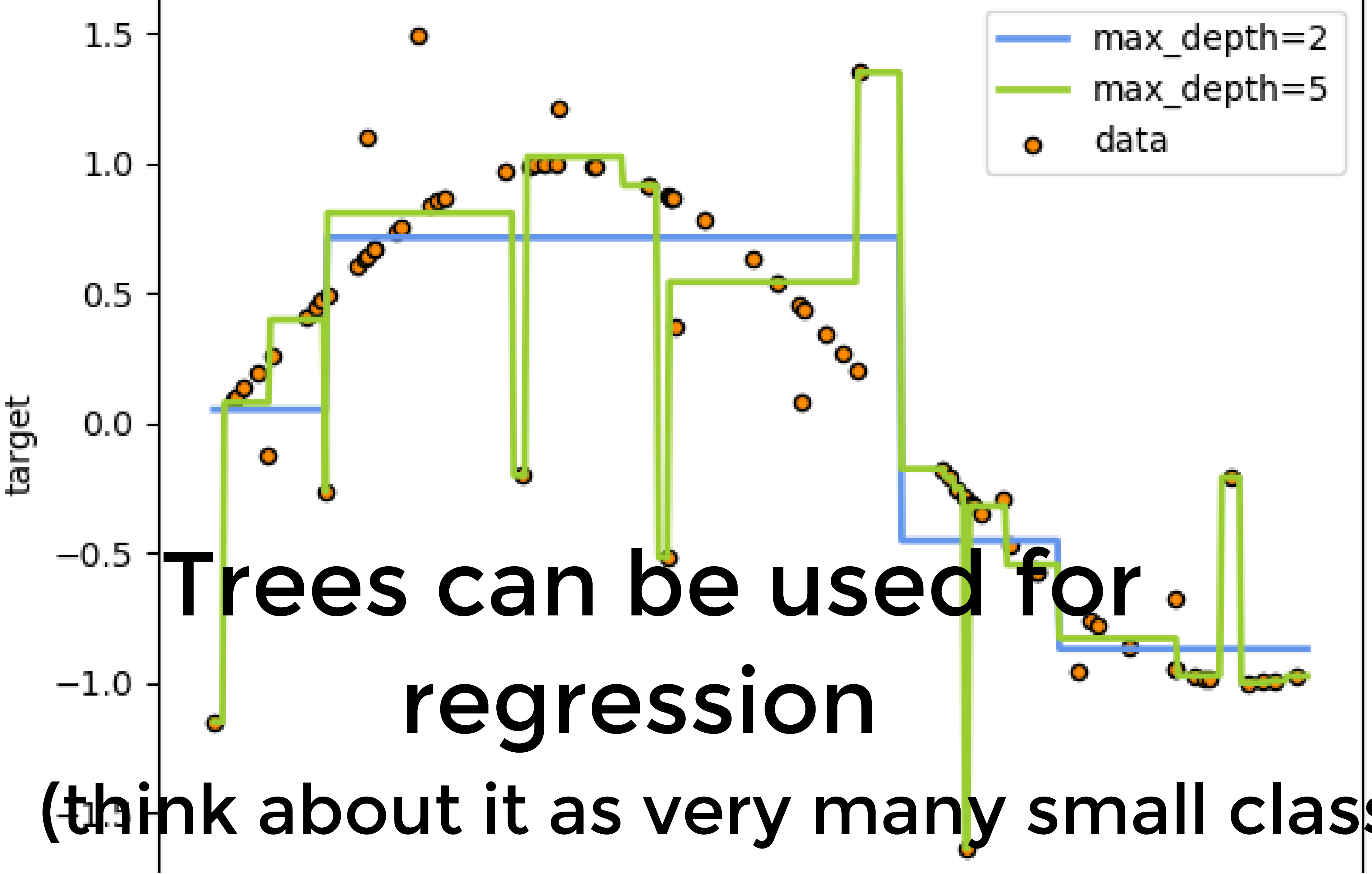


alternative: tree pruning

regression with trees

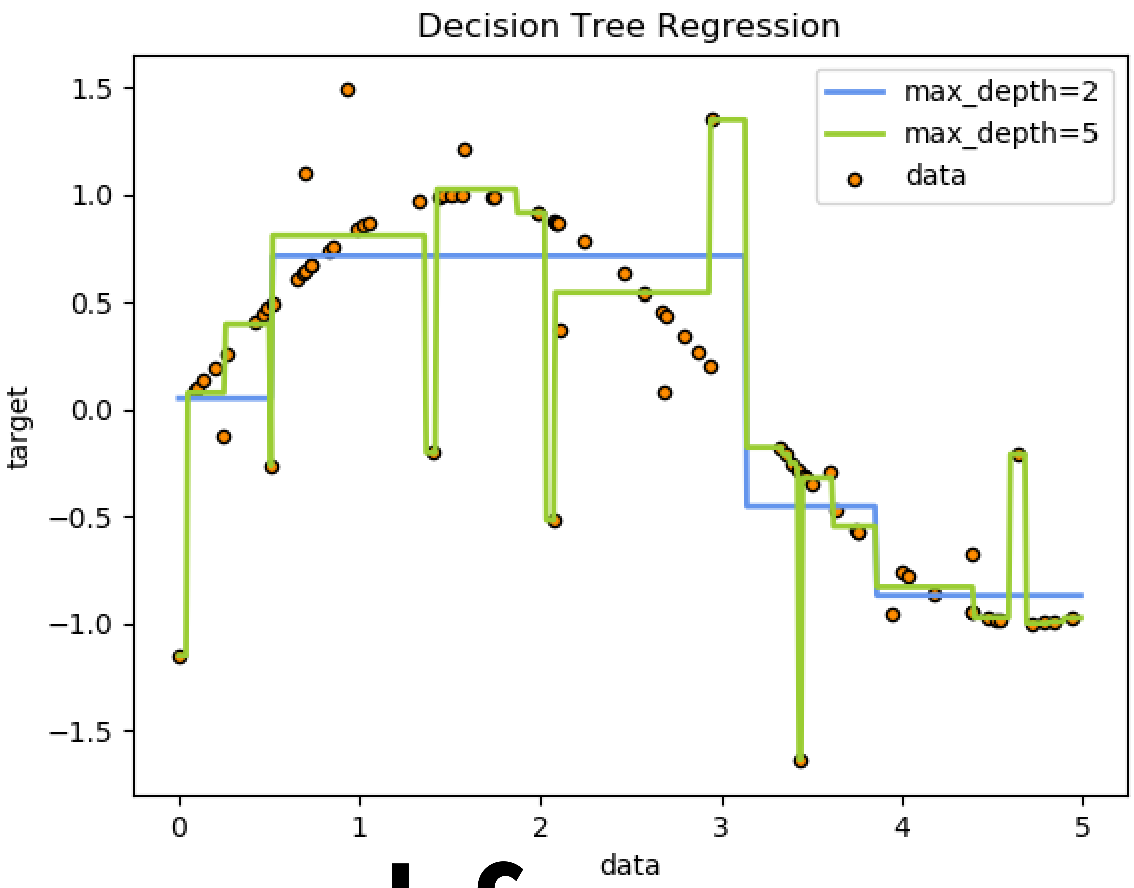
3

CART: Classification and Regression Trees



Trees can be used for regression

(think about it as very many small classes) https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html



Trees can be used for regression

(think about it as very many small classes)

sklearn.tree.DecisionTreeRegressor

class `sklearn.tree.` **`DecisionTreeRegressor`** (*criterion='mse', 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, presort=False*) ¶ [source]

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

A single tree: hyperparameters

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

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, "friedman_mse", which uses mean squared error with Friedman's improvement score for potential splits, and "mae" for the mean absolute error, which minimizes the L1 loss using the median of each terminal node.

mean square error

$$L_2 = \left(y_{true} - y_{predicted}\right)^2$$

mean absolute error

$$L1 = \left|y_{true} - y_{predicted}\right|$$

issues with trees

4

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

ensemble methods

5

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

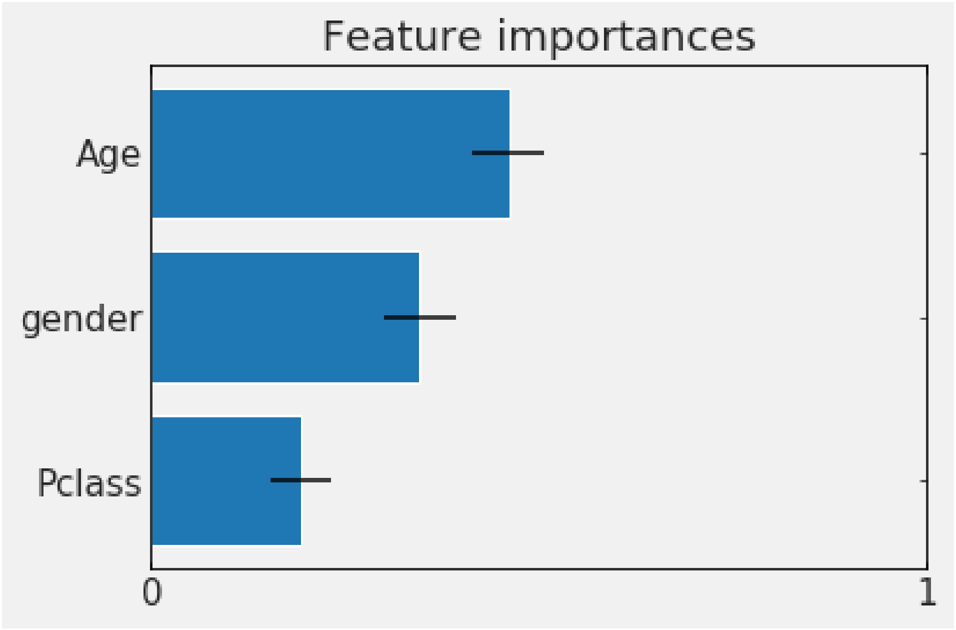
feature importance

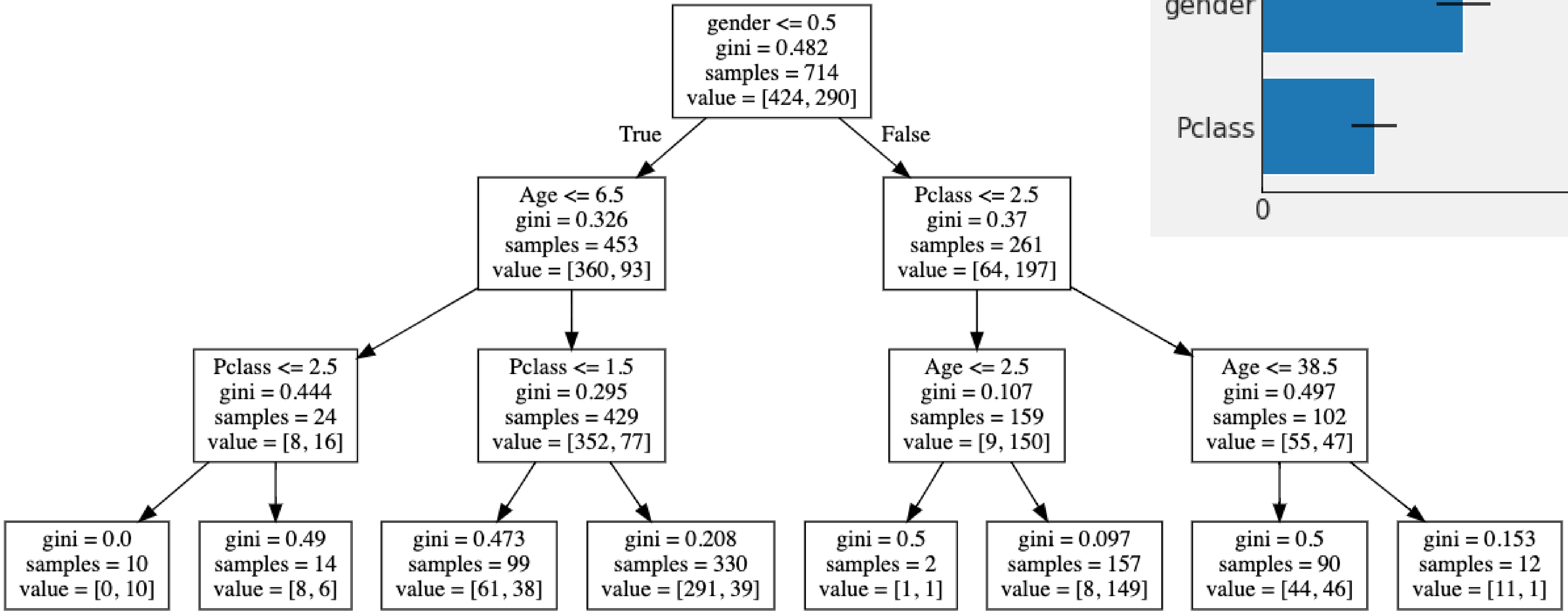
6

feature importance

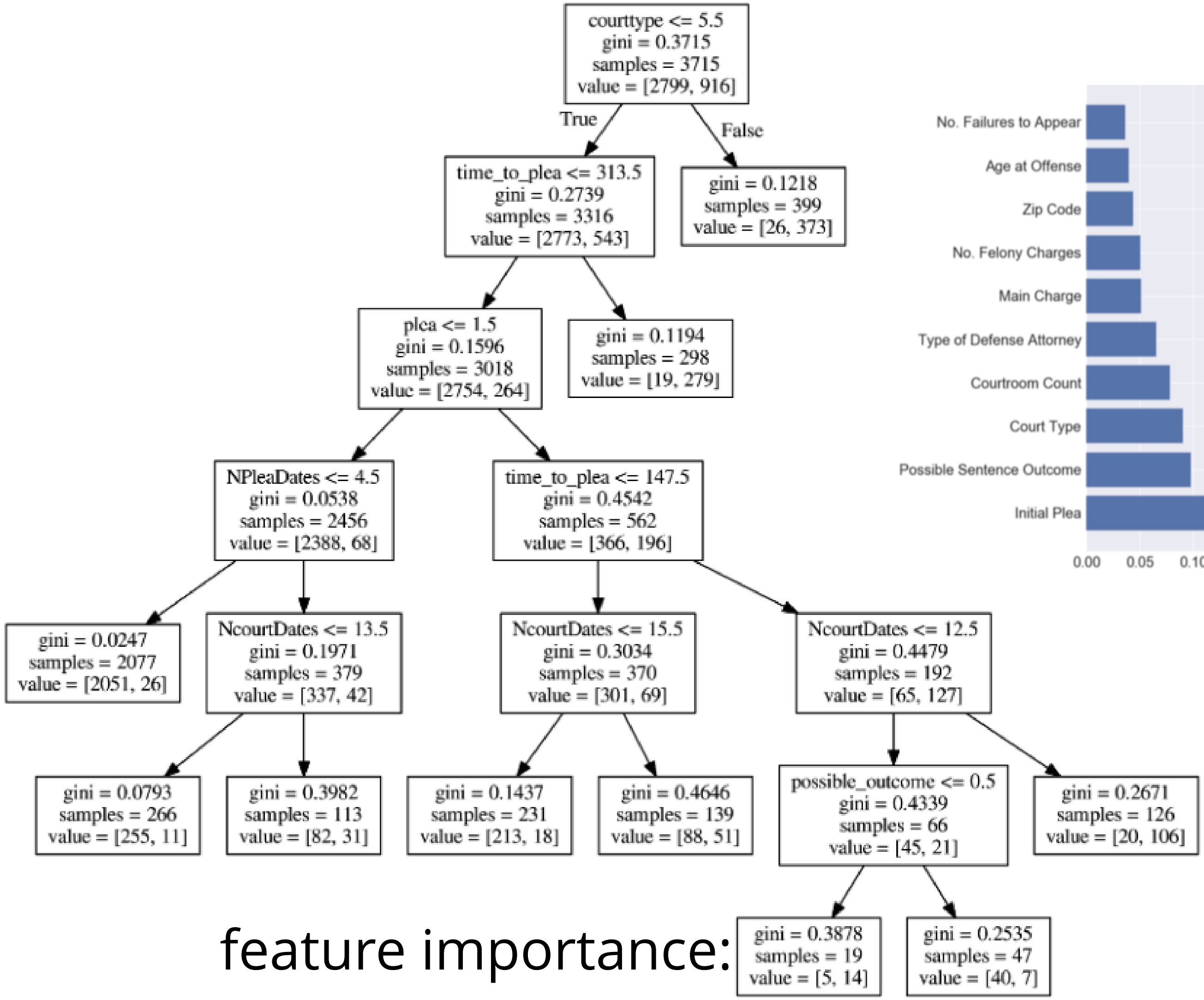
In principle CART methods are interpretable

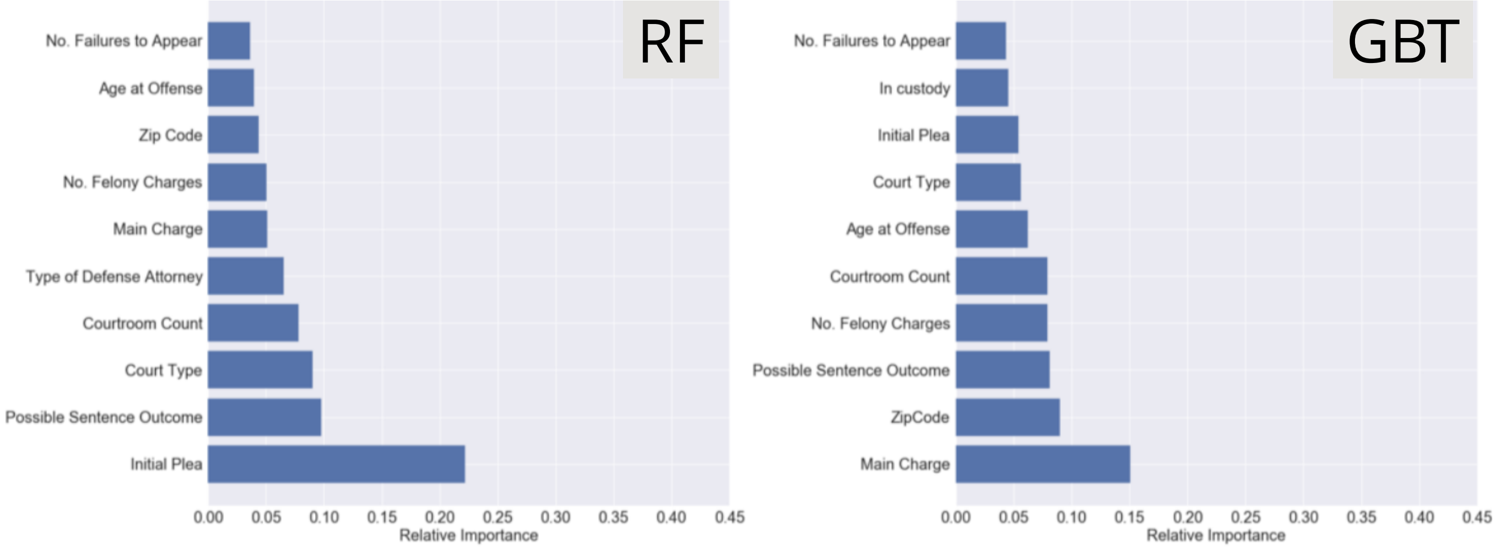
you can measure the influence that each
feature has on the decision : feature importance





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





A Data-Driven Evaluation of Delays in Criminal Prosecution

https://doi.org/10.22541/au.155535549.97131926

feature importance:

how soon was a feature chosen,

how many times was it used...

https://explained.ai/rf-importance/

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

Machine Learning includes models that learn parameters from data

ML models have parameters learned from the data and **hyperparameters** assigned by the user.

Unsupervised learning:

- all variables observed for all data points
- learns the structure of the features space from the data
- predicts a label (group of belonging) based on similarity of all features

Supervised learning:

- a target feature is observed only for a subset of the data
- learns target feature for data where it is not observed based on similarity of the other features
- predicts a class/value for each datum without observed label

Tree methods:

- partition the space one feature at a time with binary choices
- prone to overfitting
- can be used for regression

key concepts

single trees have high variance as the optimization has to be local

ensemble methods solve variance issue by running multiple trees and making an ensemble decision

random forest: trees run in parallel with a random subset of features and the decision scheme is "majority" decision

gradient boosted trees: trees run in series with feature weighted learning the weights from the outcome of the previous tree. The last tree has the division

feature importance: the importance of each feature can be extracted. In presence of covariance the feature importance may be hard to interpret 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

actually a video: watching not reading (~1 hour)

https://www.youtube.com/watch?v=Trar7GZOPl8&feature=youtu.be&utm_medium=email&utm_source=intercom&utm_campaign=modular-code-event

reading

Create a plot, of whatever data (and models if you want) you choose from open data (if you have doubt about whether your dataset is relevant for this homework please email me.)

You can make the plot in any coding language you want (e.g. python, javascript, R...), as long as you upload the code that generates the plot onto your repo (which means no tableau, or any other non reproducible).

Create a directory HW8_<firstLast> in your DSPS repo. **The plot neads to be uploaded onto the HW8 folder in your github DSPS repo and be embedded in the README.md.** That means: when I click on the HW8 link the plot must be rendered in the front page of the repo. Your readme must contain the plot, and a brief caption. If it is an interactive graphic, upload a static image of it in the README and provide a link to the interactive version.

Please make an effor to make it a good, compelling graphic. Put though into the esthetic of the plot, how clearly the content is communicated, avoid clutter, avoid misleading elements, mind your choice of colors accordingly to what was discussed in class.

Each of you needs to upload their own plot, no group submissions.

If your plot shows up as I described above in the repo and the code is also uploaded you will get 100% of the HW points. (Next week you will be tasked to review 3 plots of your classmates and you will be graded on the quality of the review.)

Follow scheleton notebook to create an H-R diagram visualization with datapoints and contours

EC: make your visualization interactive so that rolling on any datapoint provides information about the data

homework 2