principles of Urban Science 8

random forest

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

this slide deck: https://slides.com/federicabianco/pus2020_8

- Machine Learning basic concepts
  - interpretability
  - parameters vs hyperparameters
  - supervised/unsupervised
- Tree methods
  - single trees
  - hyperparameters
  - weaknesses
- Tree ensembles
- Feature importance
- Categorical feature encoding
- ML models performance evaluation

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

clustering vs classifying

unsupervised *supervised*

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

separated in groups

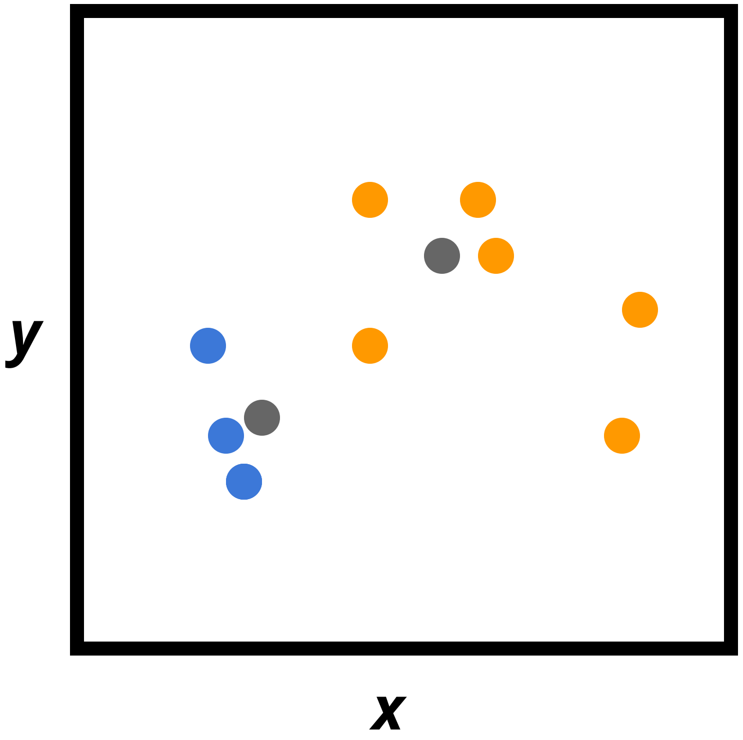
consistently with

1

an observed subset

observed features:

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



target features:

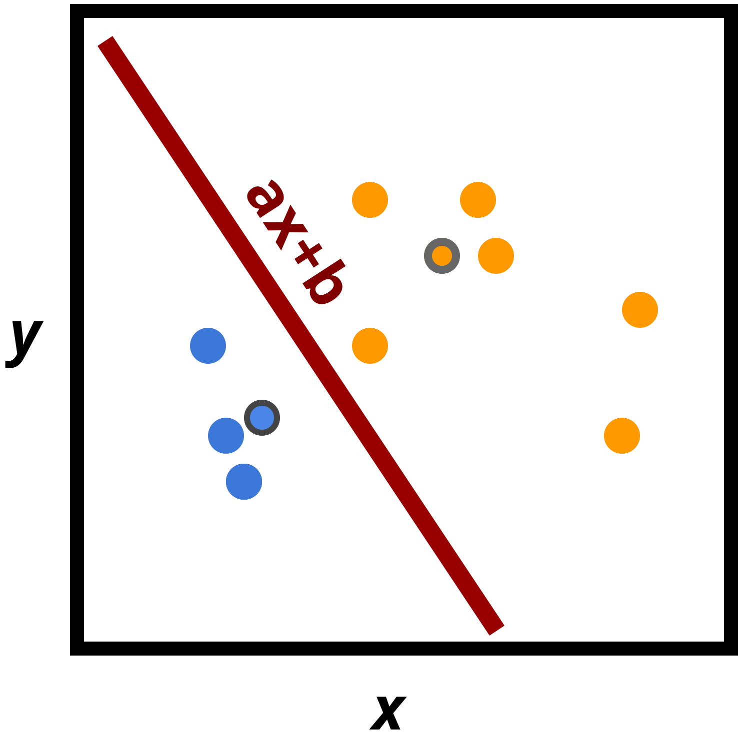
$(\overrightarrow{color})$

models typically return a partition of the space

clustering vs classifying

unsupervised *supervised*

observed features: $(\vec{x}, \vec{y})$



target features: $(\overrightarrow{color})$

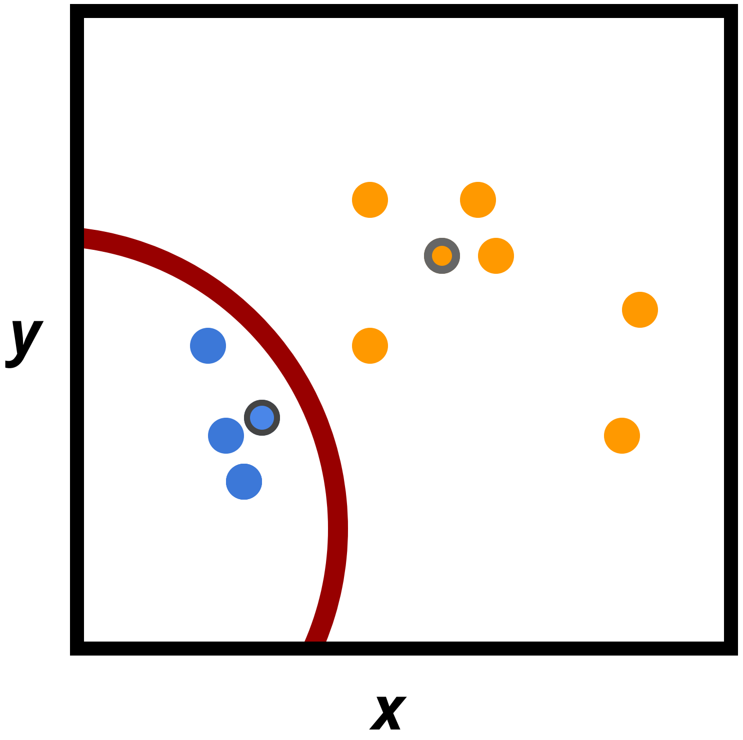
```
if y <= a*x + b :
        return blue
else:
        return orange
```

clustering vs classifying

unsupervised *supervised*

observed **features:**

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



target **features:**

$(\overrightarrow{color})$

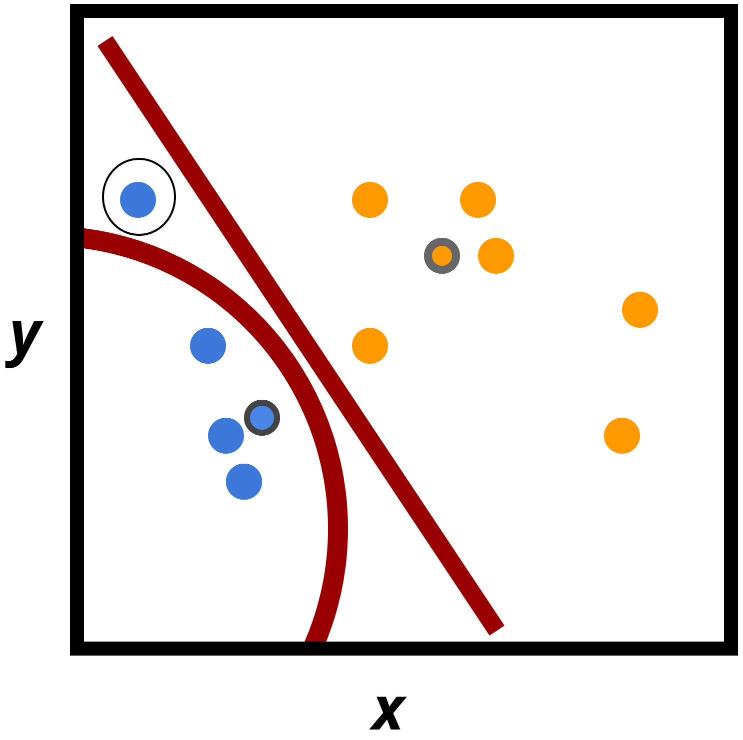
```
if x**2 + y**2 <= (x-a)**2 + (y-b)**2 :
        return blue
else:
        return orange
```

clustering vs classifying

unsupervised *supervised*

observed **features:**

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



target **features:**

$(\overrightarrow{color})$

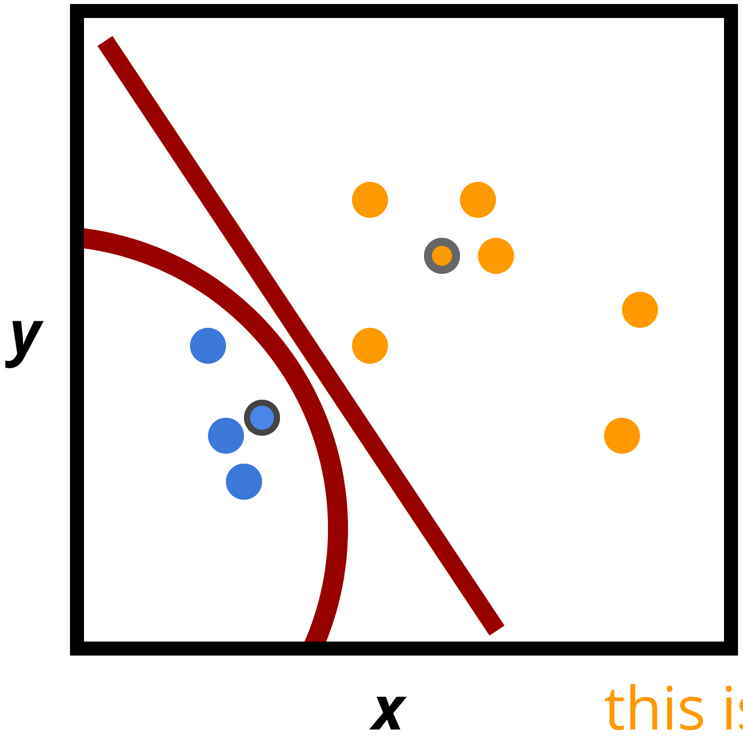
```
if x**2 + y**2 <= (x-a)**2 + (y-b)**2 :
        return blue
else:
        return orange
```

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

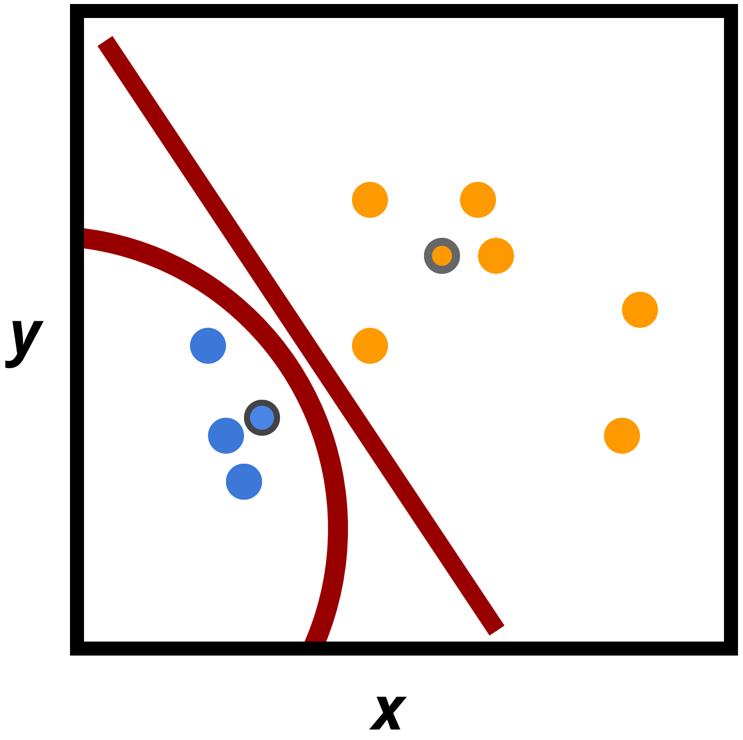
supervised ML: classification

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

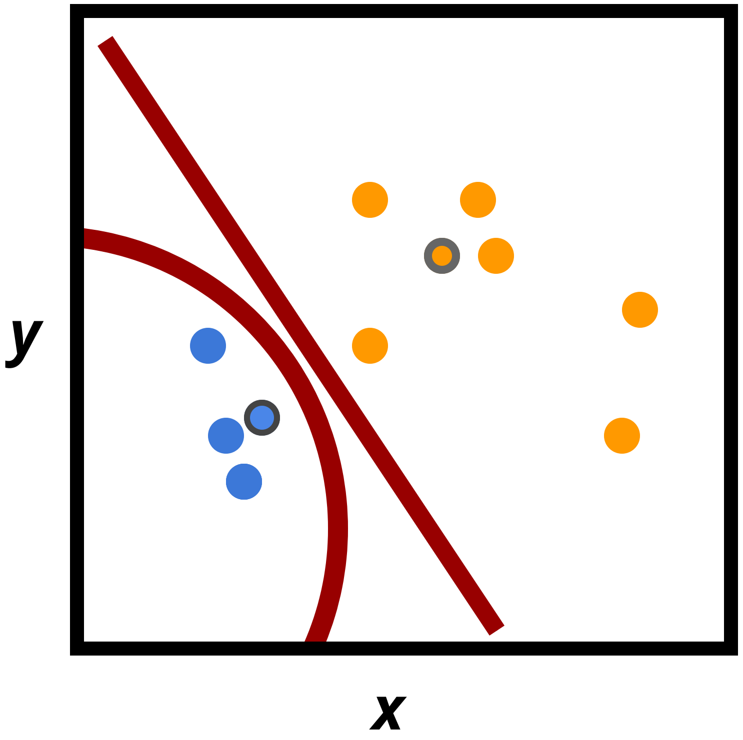
A subset of variables has class labels.
Guess the label for the other variables

Support Vector Machine:

finds a hyperplane that partitions the space

observed features:

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



2d hyperplane: line (curve)

3d hyperplane: surface

4d hyperplane: volume

...

target features:

$(\overrightarrow{color})$

supervised ML: classification

A subset of variables has class labels.

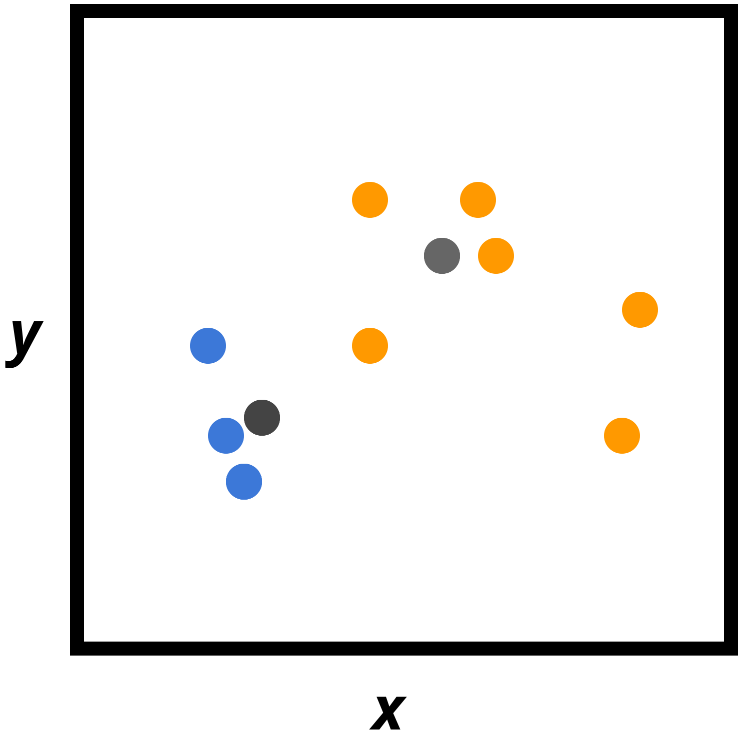
Guess the label for the other variables

Tree Methods

split spaces along each axis separately

observed features:

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



target features:

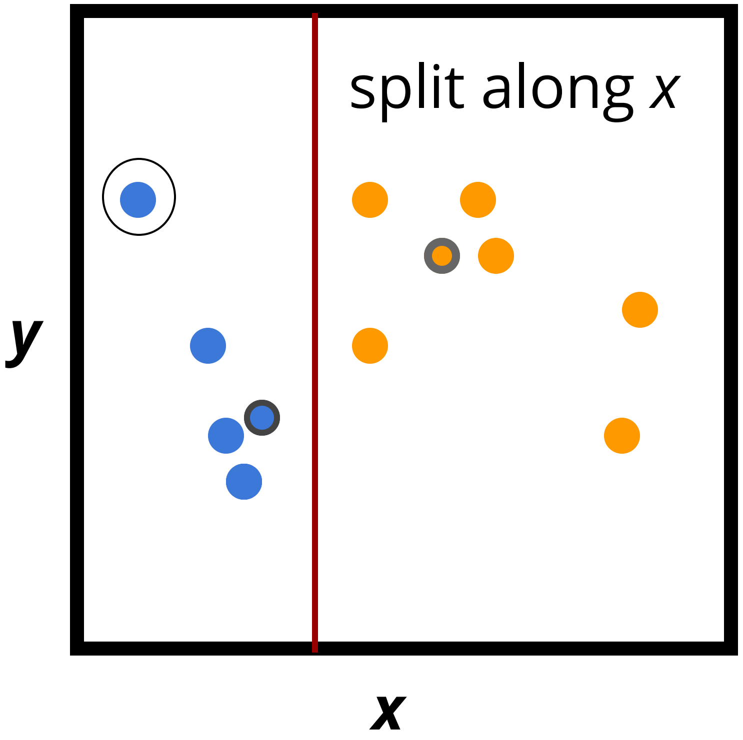
($\overrightarrow{color}$)

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

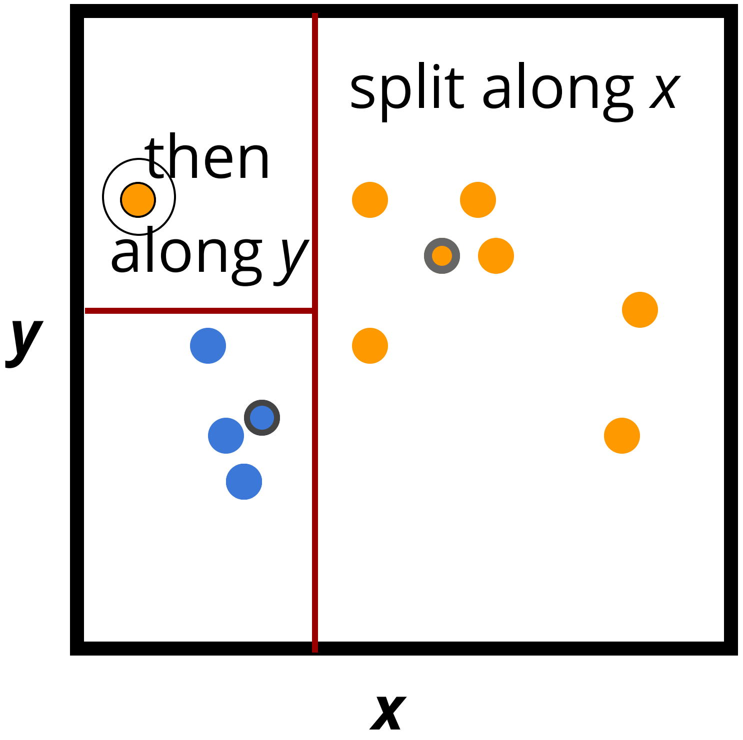
A subset of variables has class labels.
Guess the label for the other variables

Tree Methods

split spaces along each axis separately

observed features:

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



target features:

$(\overrightarrow{color})$

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

single tree

1

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...)

Application:

a robot to predict surviving the Titanic

(Kaggle)

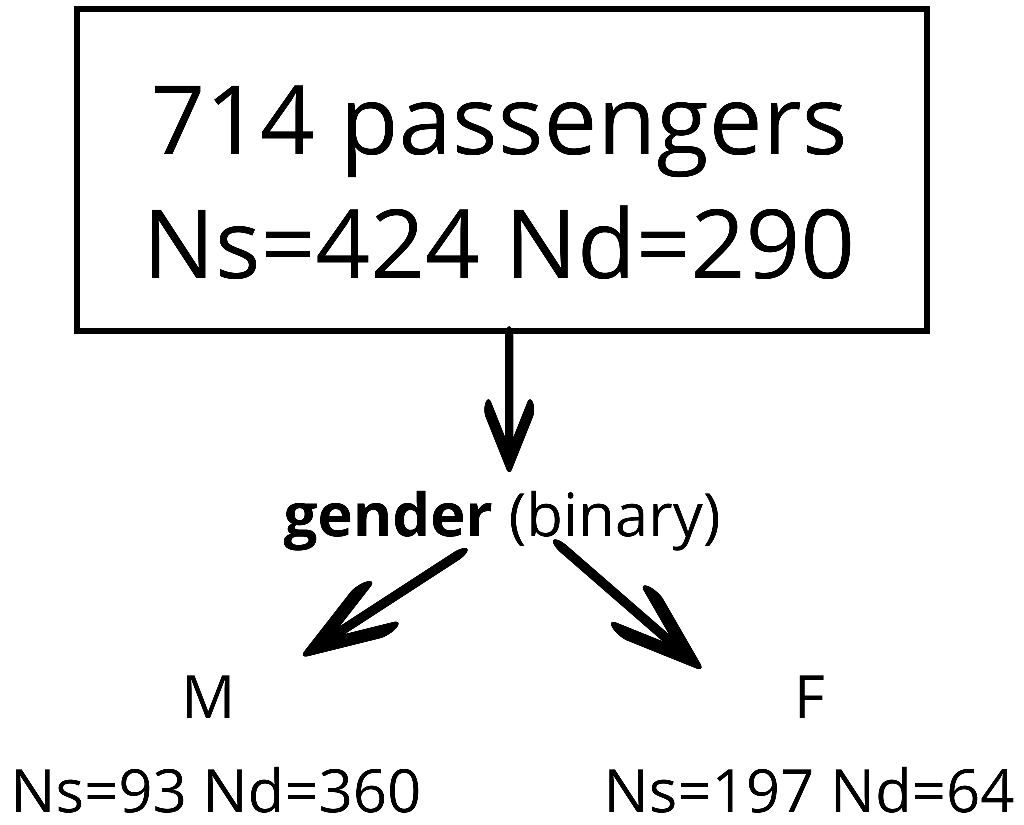
https://www.kaggle.com/c/titanic

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)

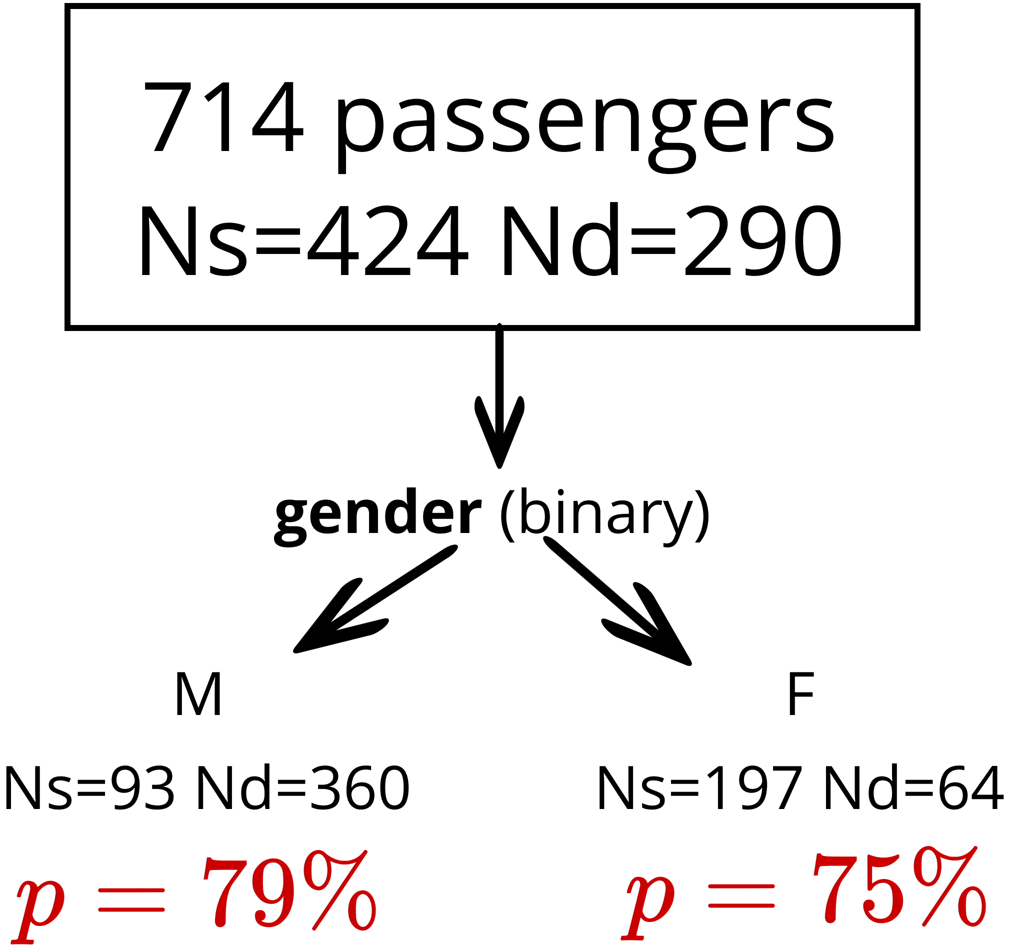
https://www.kaggle.com/c/titanic

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)

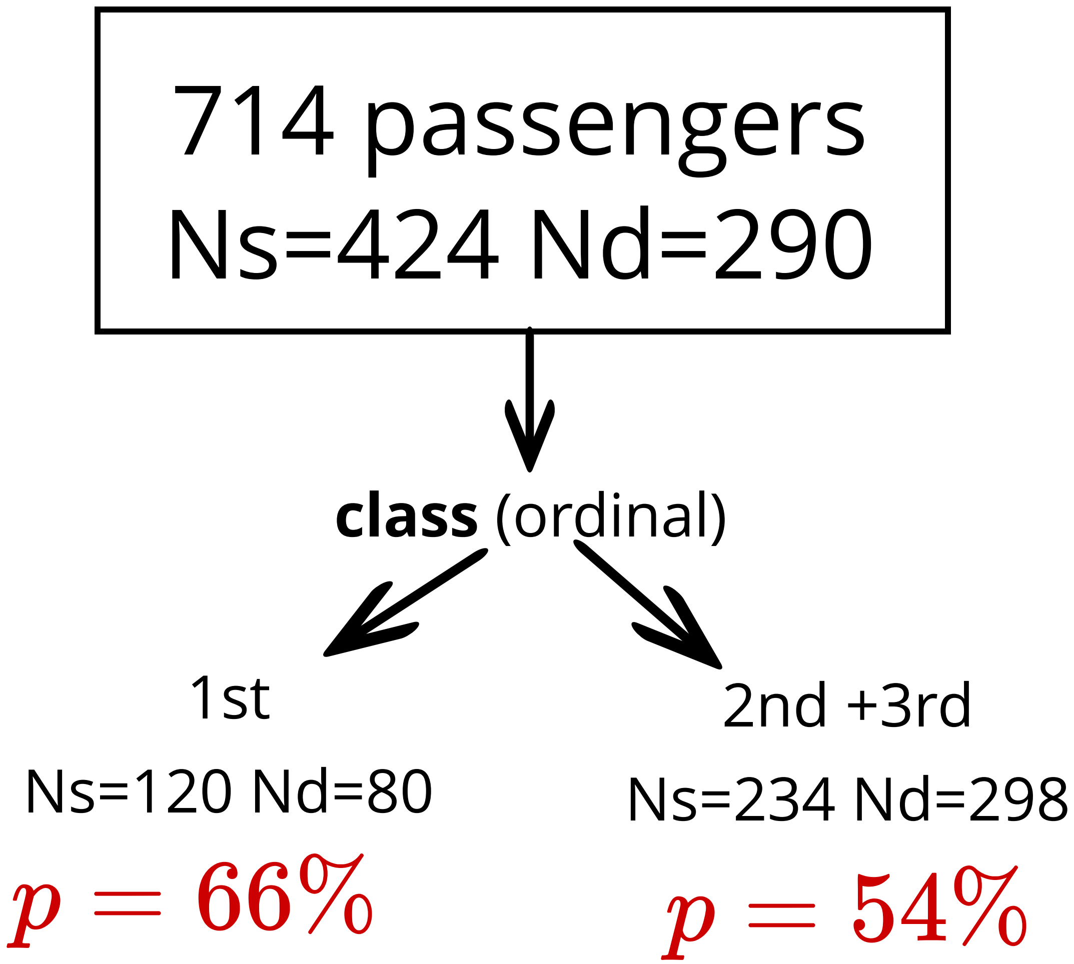
https://www.kaggle.com/c/titanic

features:

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

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

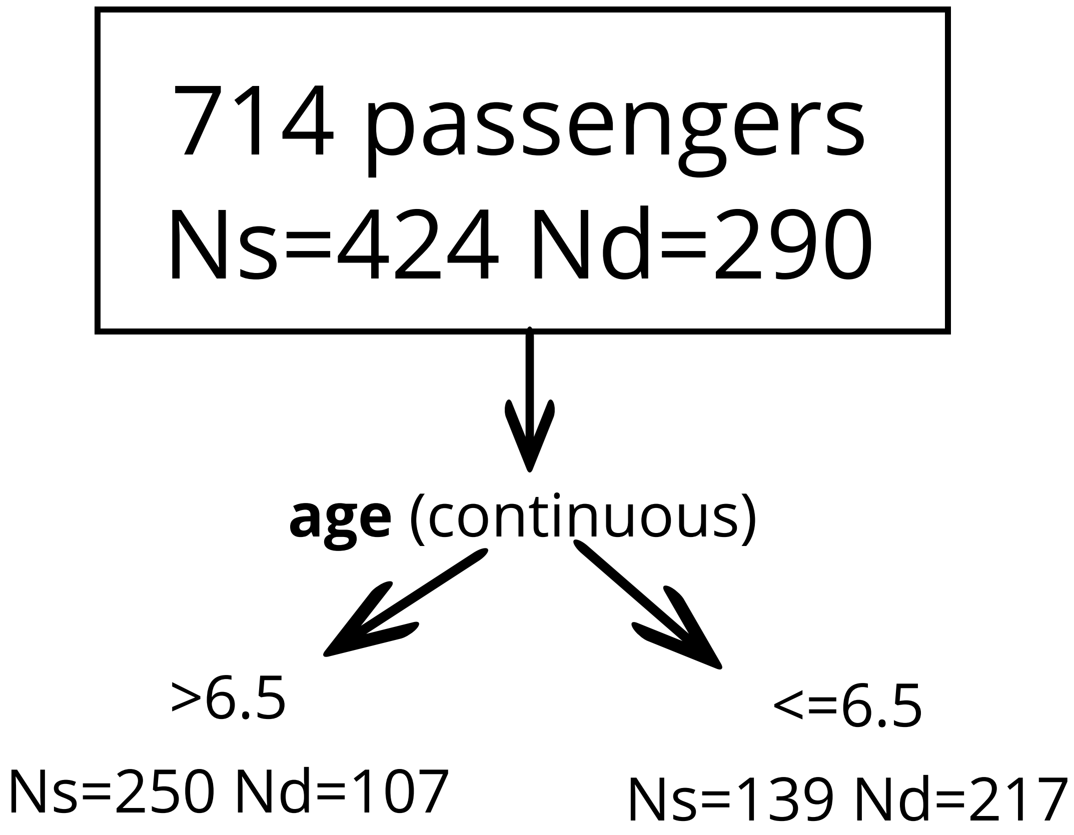
https://www.kaggle.com/c/titanic

features:

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

target variable:

-> survival (y/n)



Application:

a robot to predict surviving the Titanic

(Kaggle)

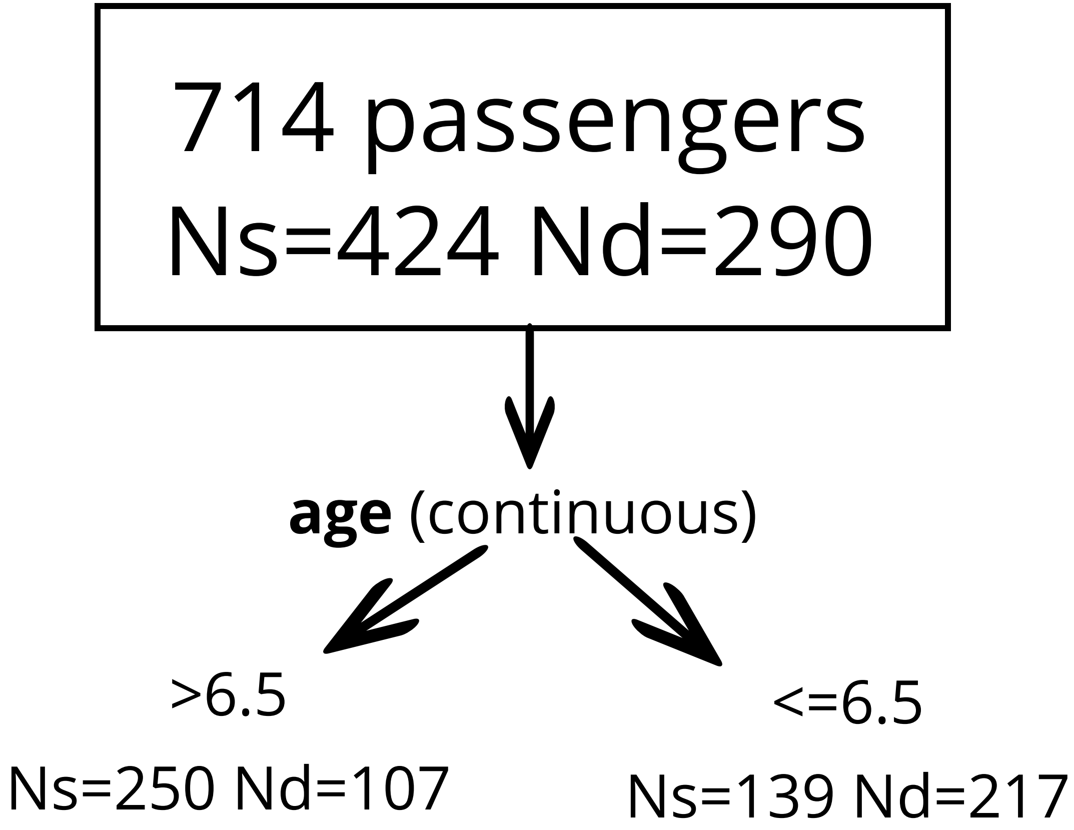
https://www.kaggle.com/c/titanic

features:

- gender 79%|75%
- ticket class 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)

714 passengers
Ns=424 Nd=290

gender (binary)

M
Ns=93 Nd=360

F
Ns=197 Nd=64

Application:

a robot to predict surviving the Titanic

(Kaggle)

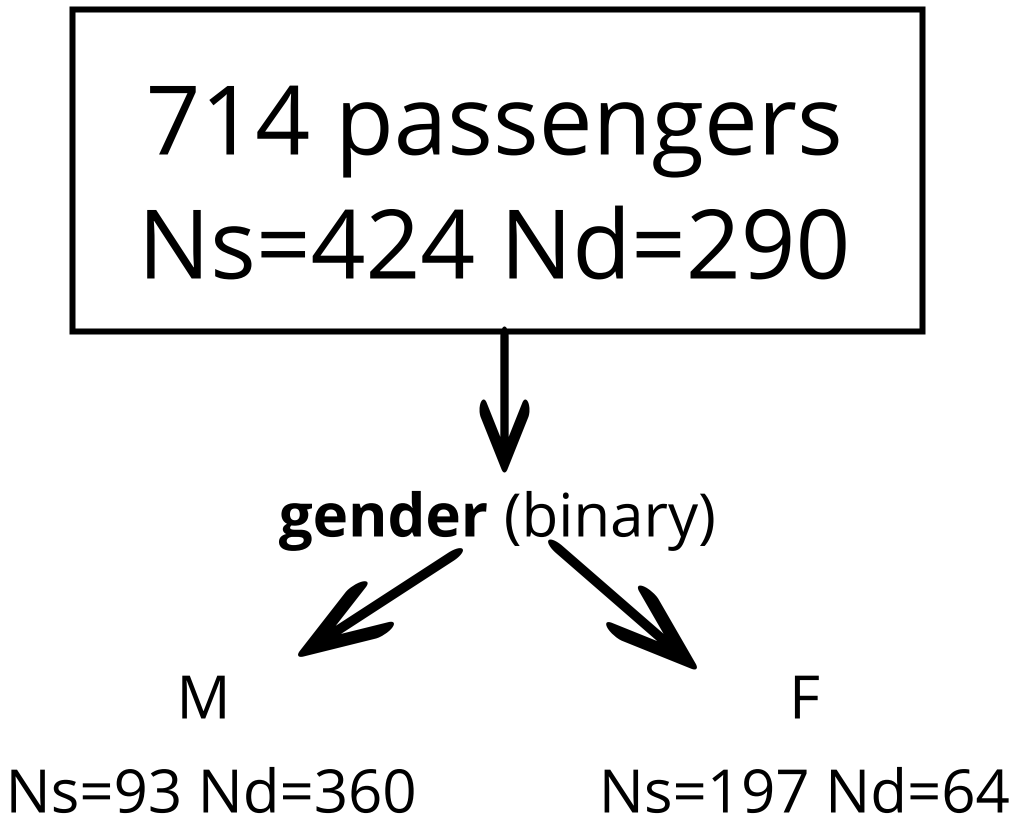
https://www.kaggle.com/c/titanic

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)

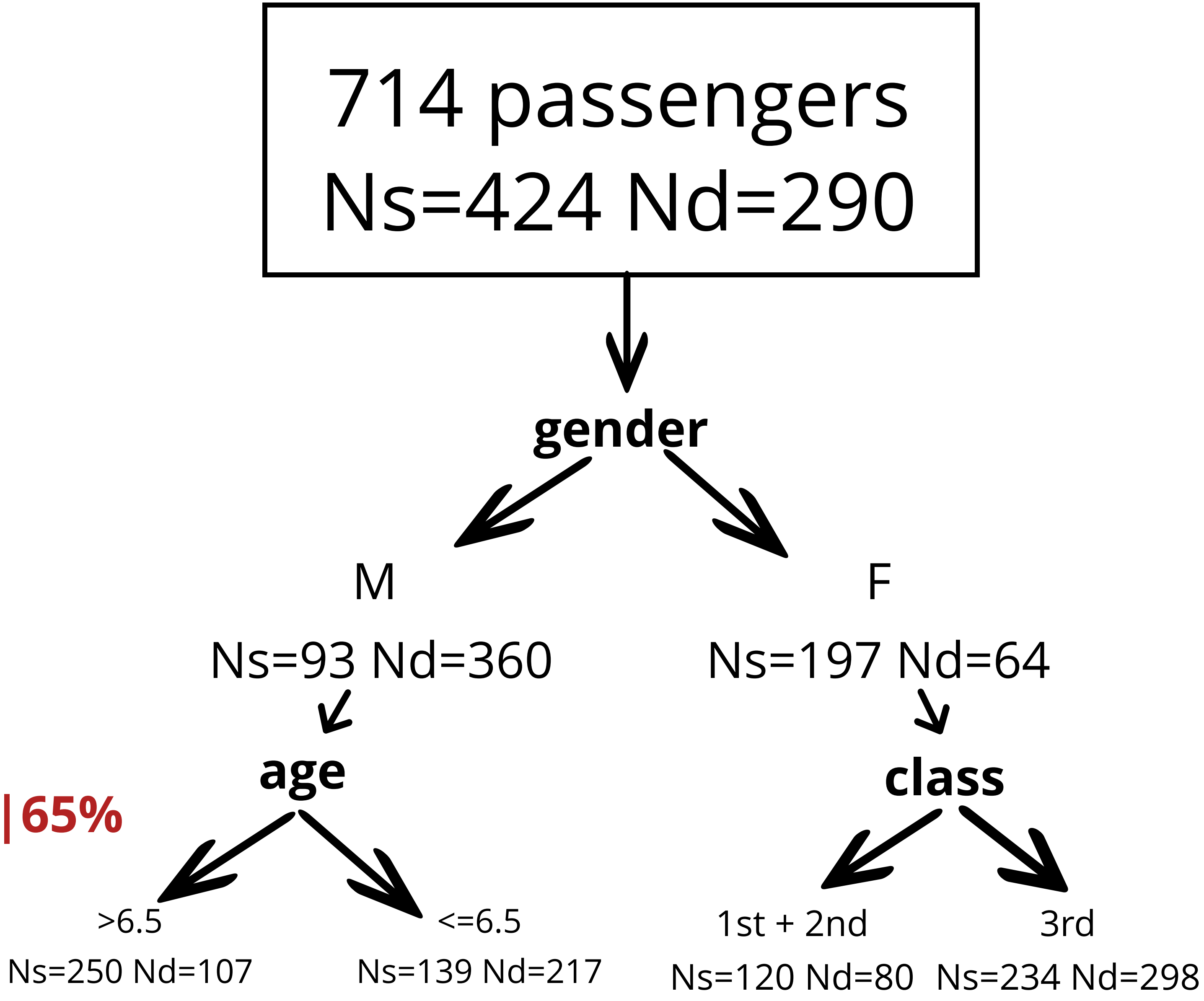
https://www.kaggle.com/c/titanic

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)

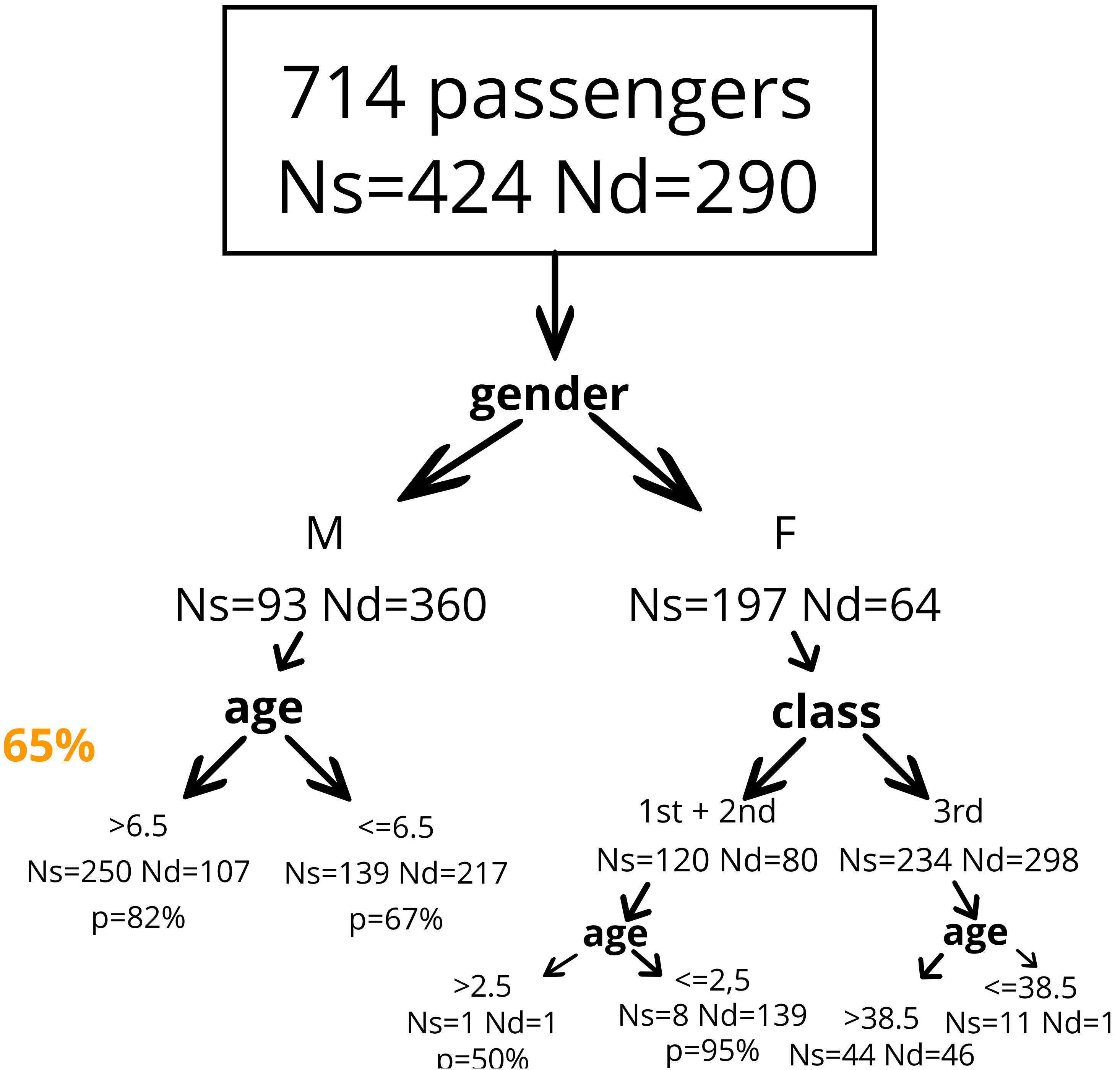
https://www.kaggle.com/c/titanic

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)

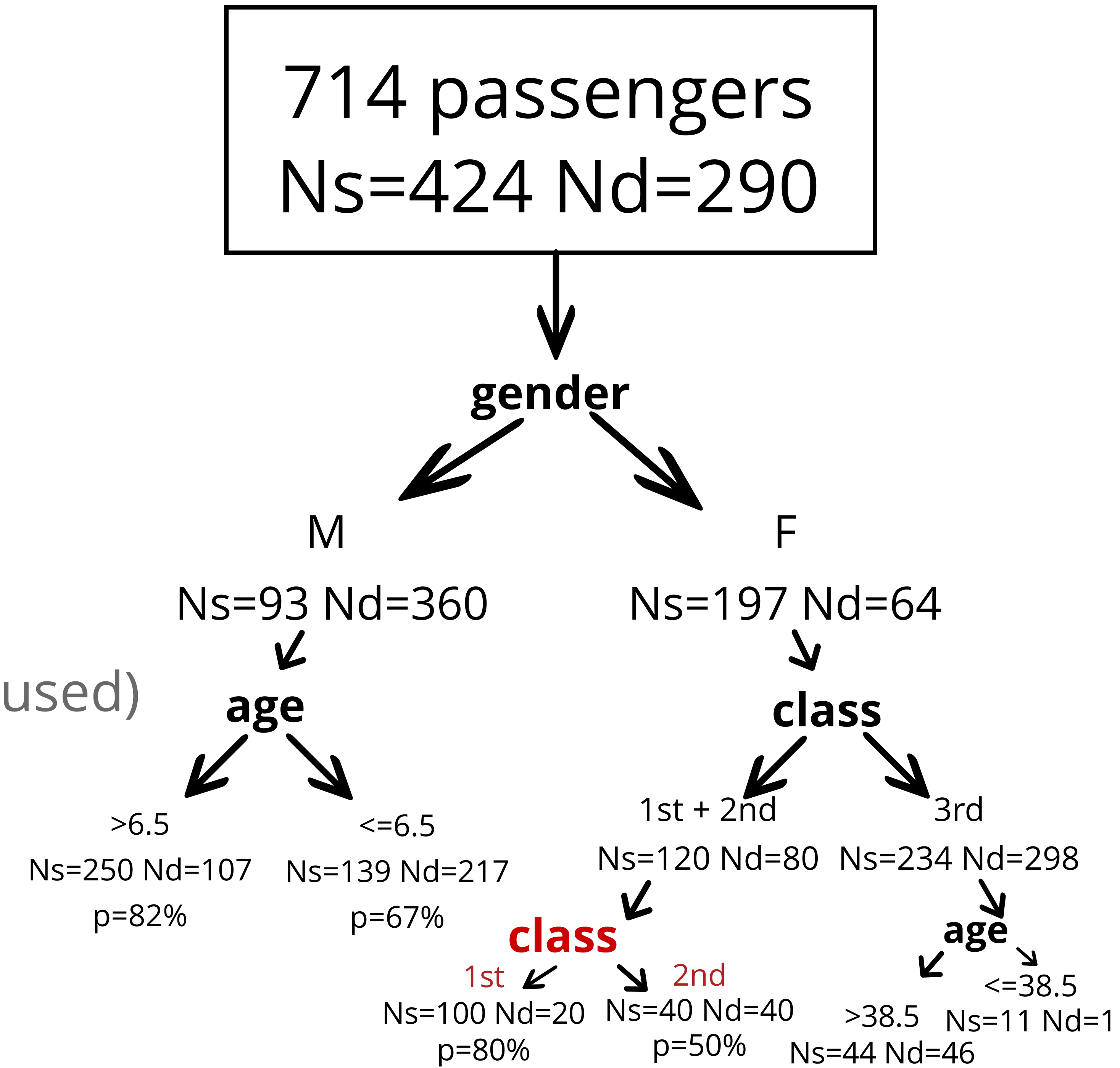
https://www.kaggle.com/c/titanic

features:

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

target variable:

-> survival (y/n)

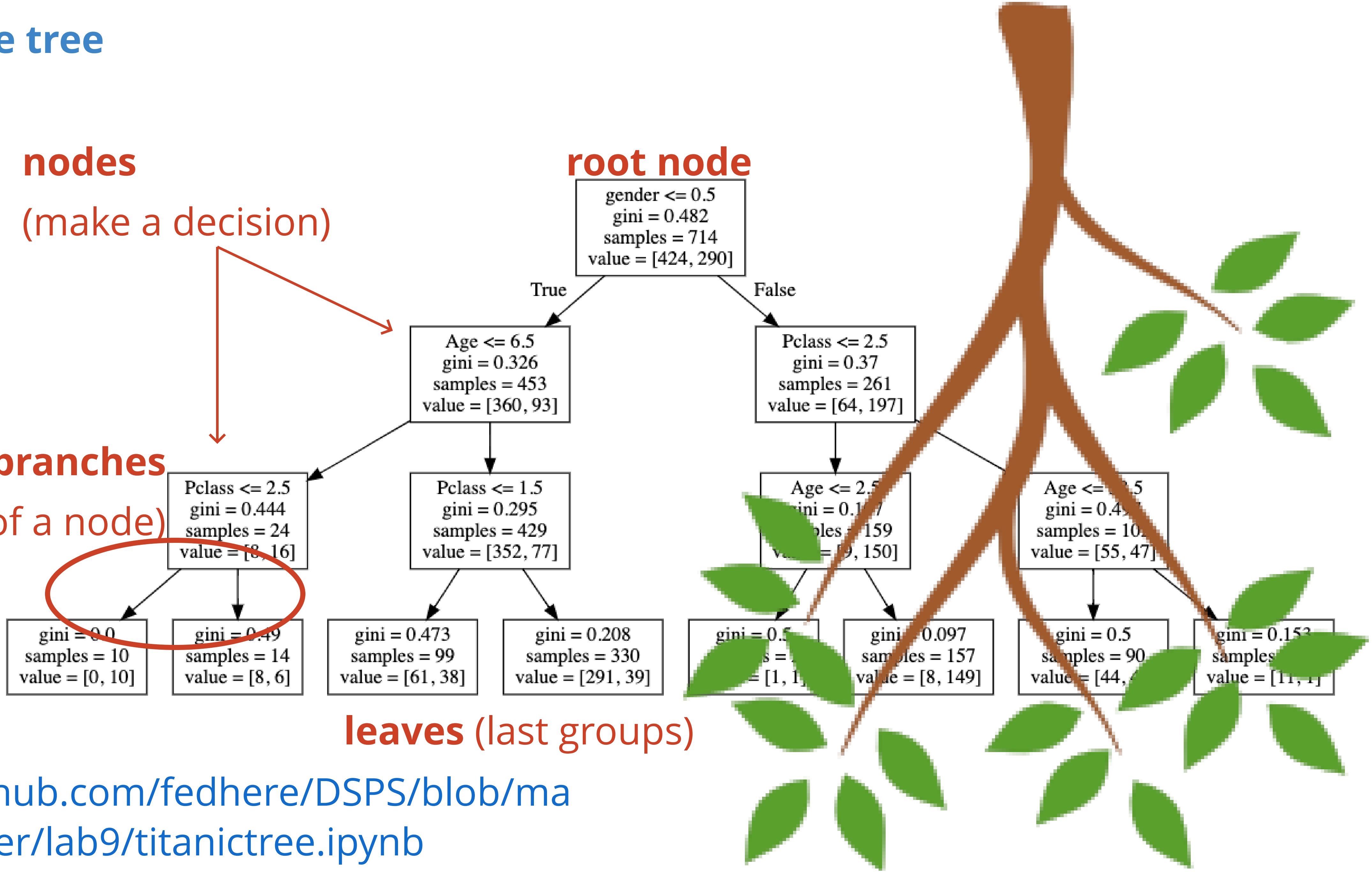


A single tree

nodes

(make a decision)

root node



branches

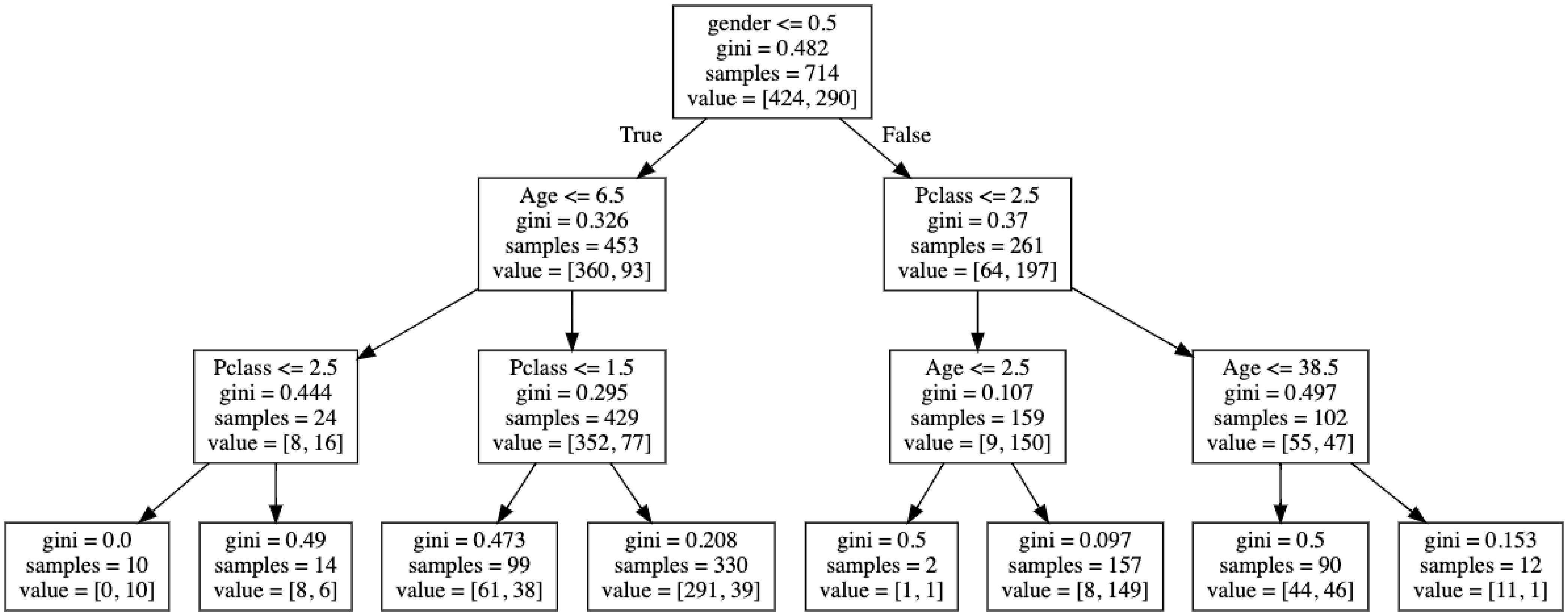
(split off of a node)

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

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

information gain (entropy)

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

p is the probability of drawing an object of a class in a random draw: in the Titanic example p is the probabilty of drawing (e.g.) a surviving passenger in the sample that enters the next node. in a frequentist sense this corresponds to the **fraction of members of the larger class over the total**

$$p_i = \frac{N_{\text{class}}}{N_{\text{total}}}$$

Application:

a robot to predict surviving the Titanic

(Kaggle)

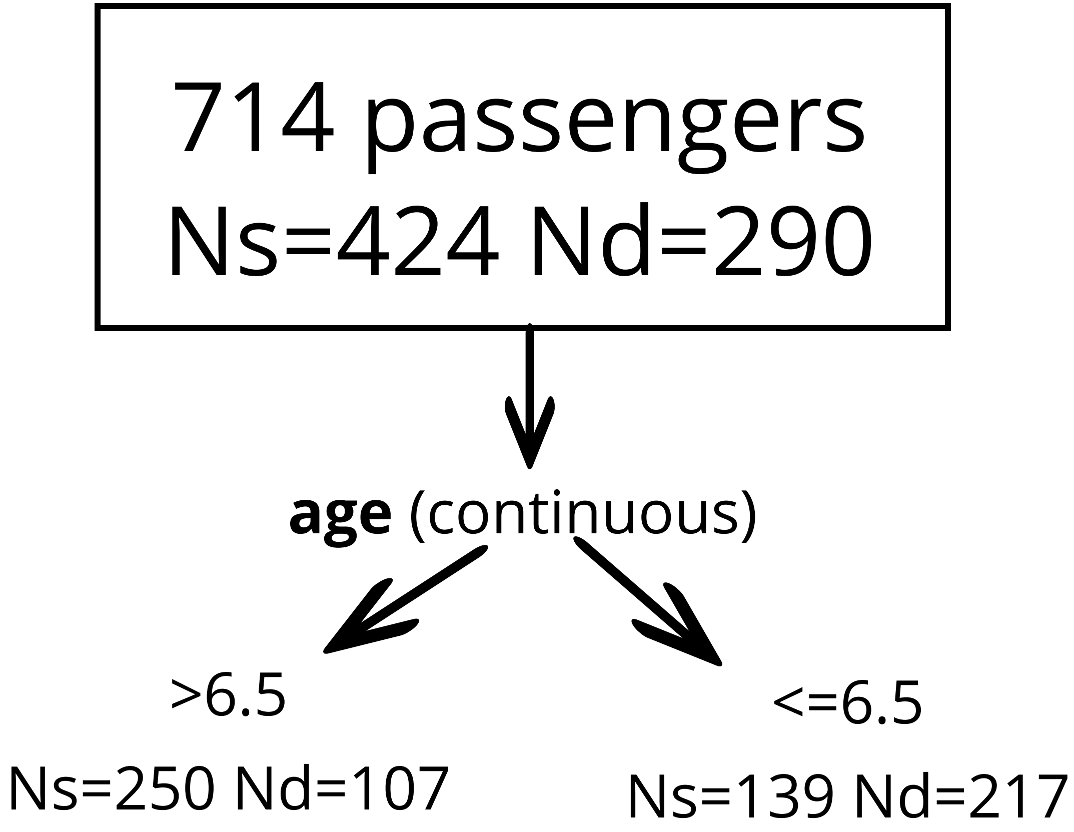
https://www.kaggle.com/c/titanic

features:

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

target variable:

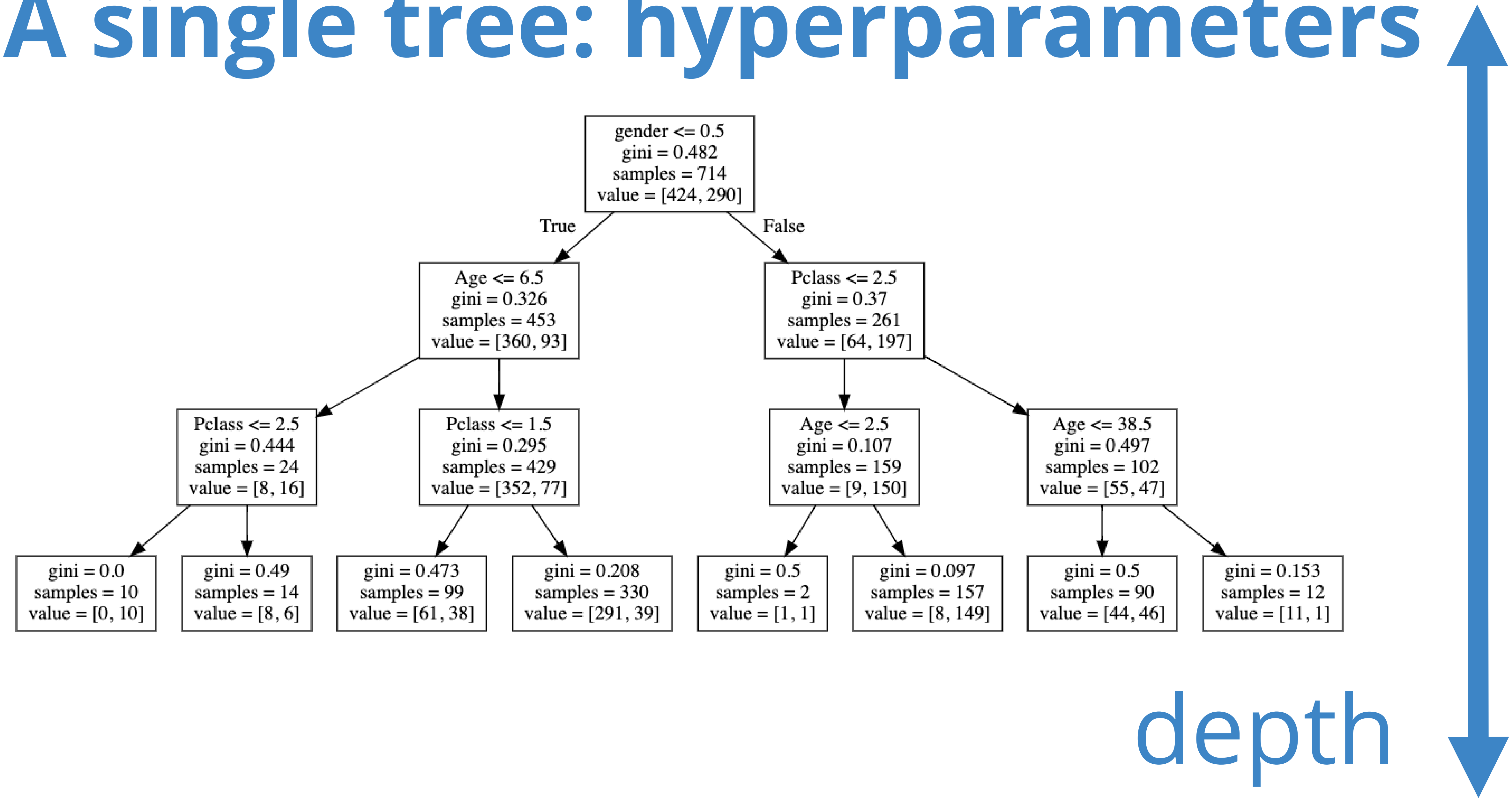
-> survival (y/n)



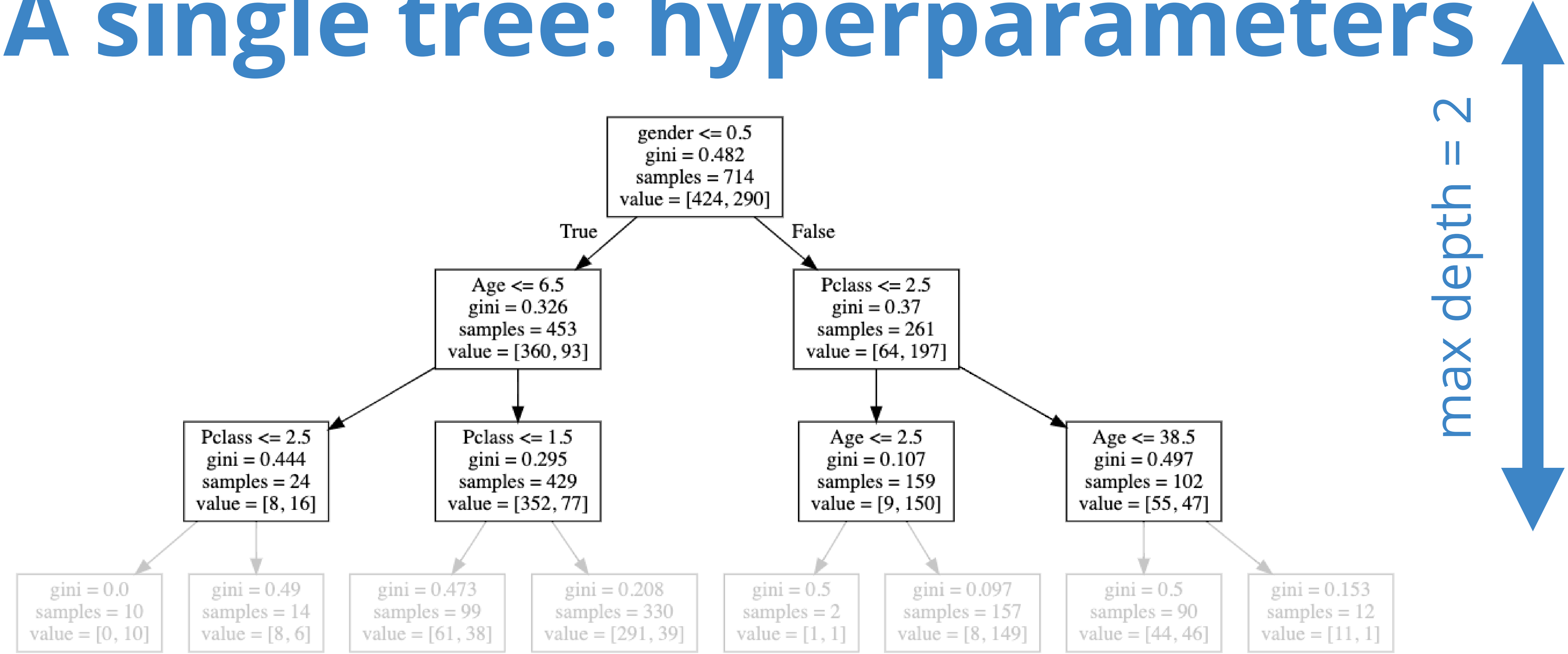
$$p_i = \frac{N_{\text{larger}}}{N_{\text{total}}} = \dots$$

https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb

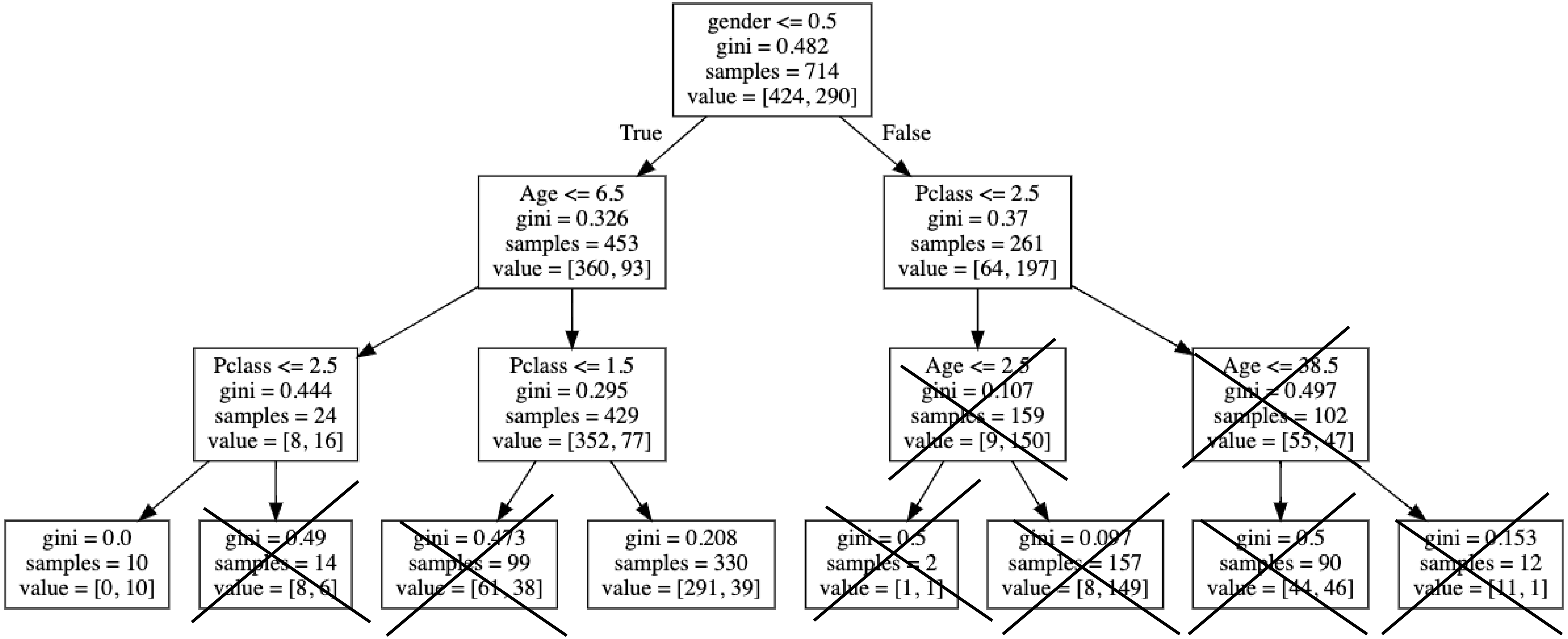
A single tree: hyperparameters



A single tree: hyperparameters



A single tree: hyperparameters



alternative: tree pruning

issues with trees

3

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

4

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

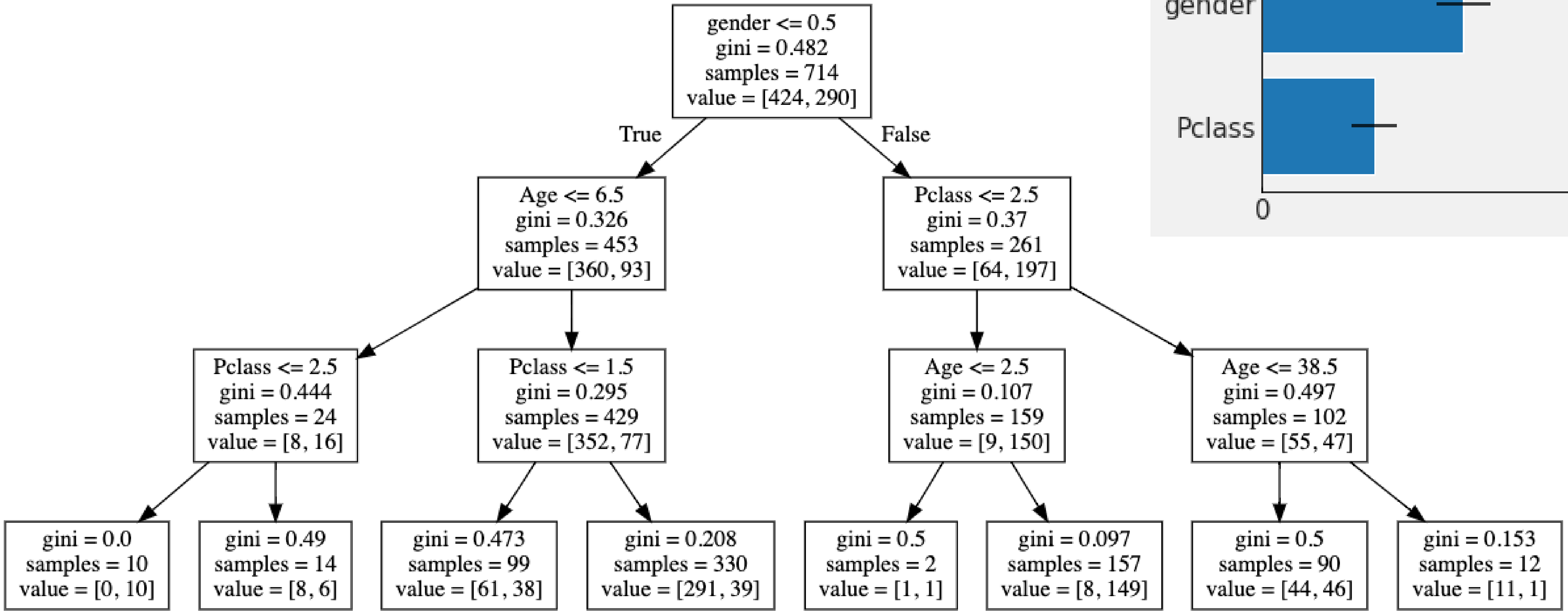
feature importance

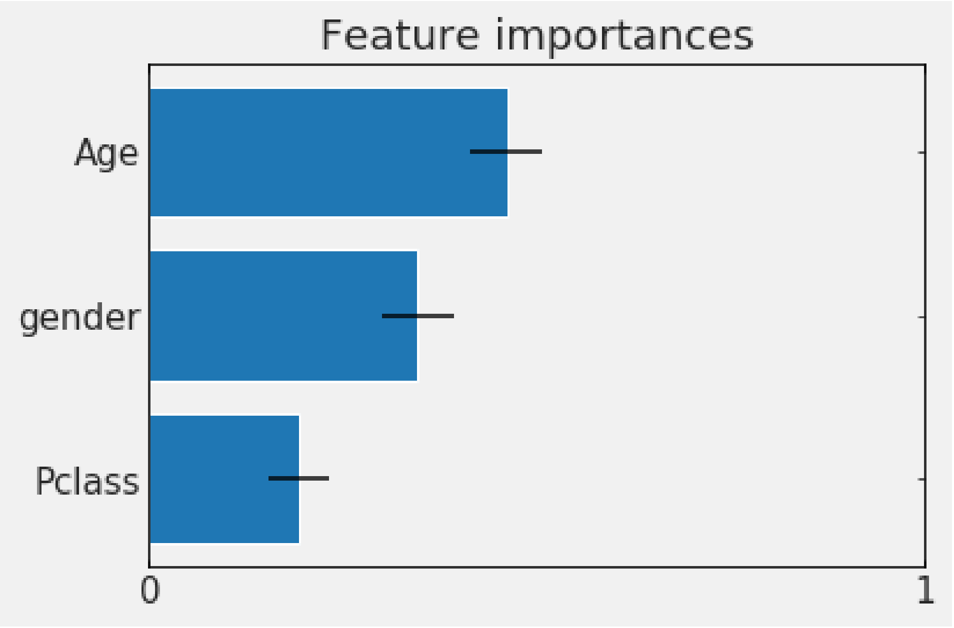
5

feature importance

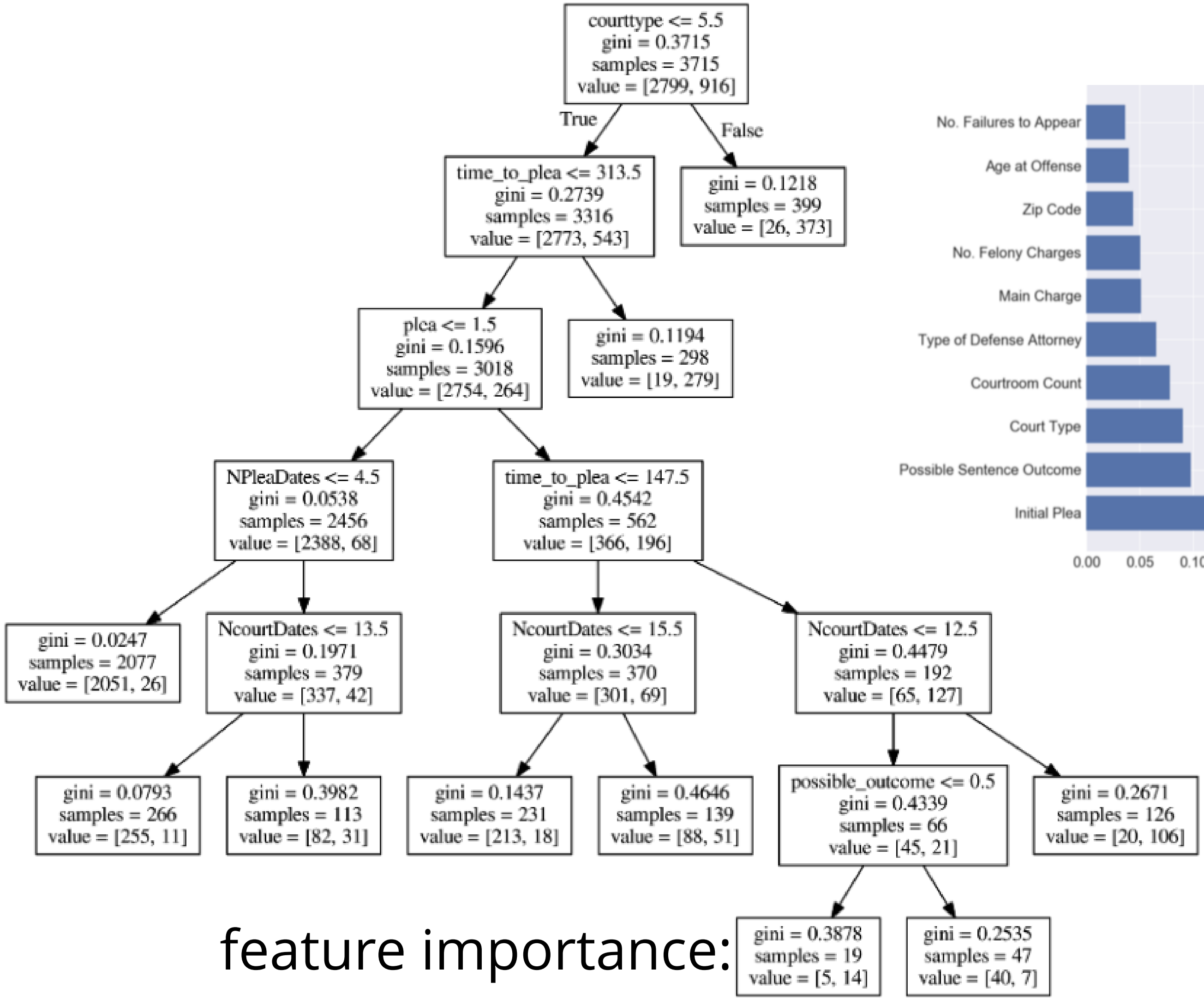
In principle CART methods are interpretable

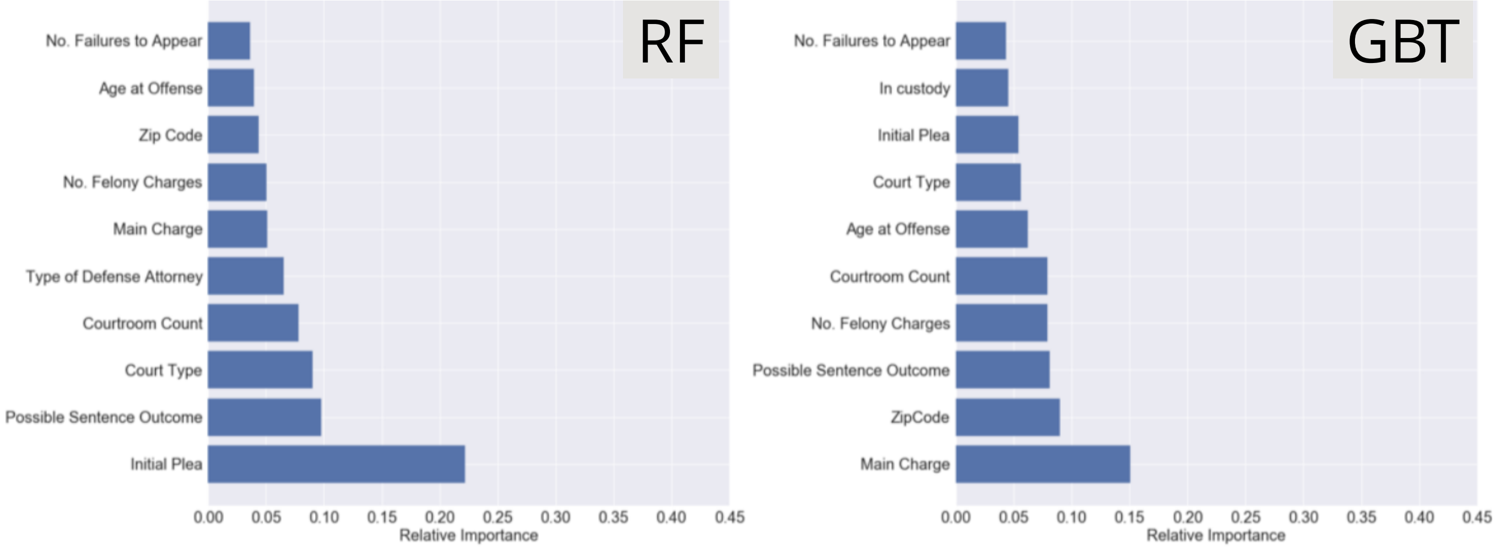
you can measure the influence that each feature has on the decision : feature importance





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

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

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

change each category to a binary

Preferred!

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

problematic if you are interested in feature importance

numerical encoding

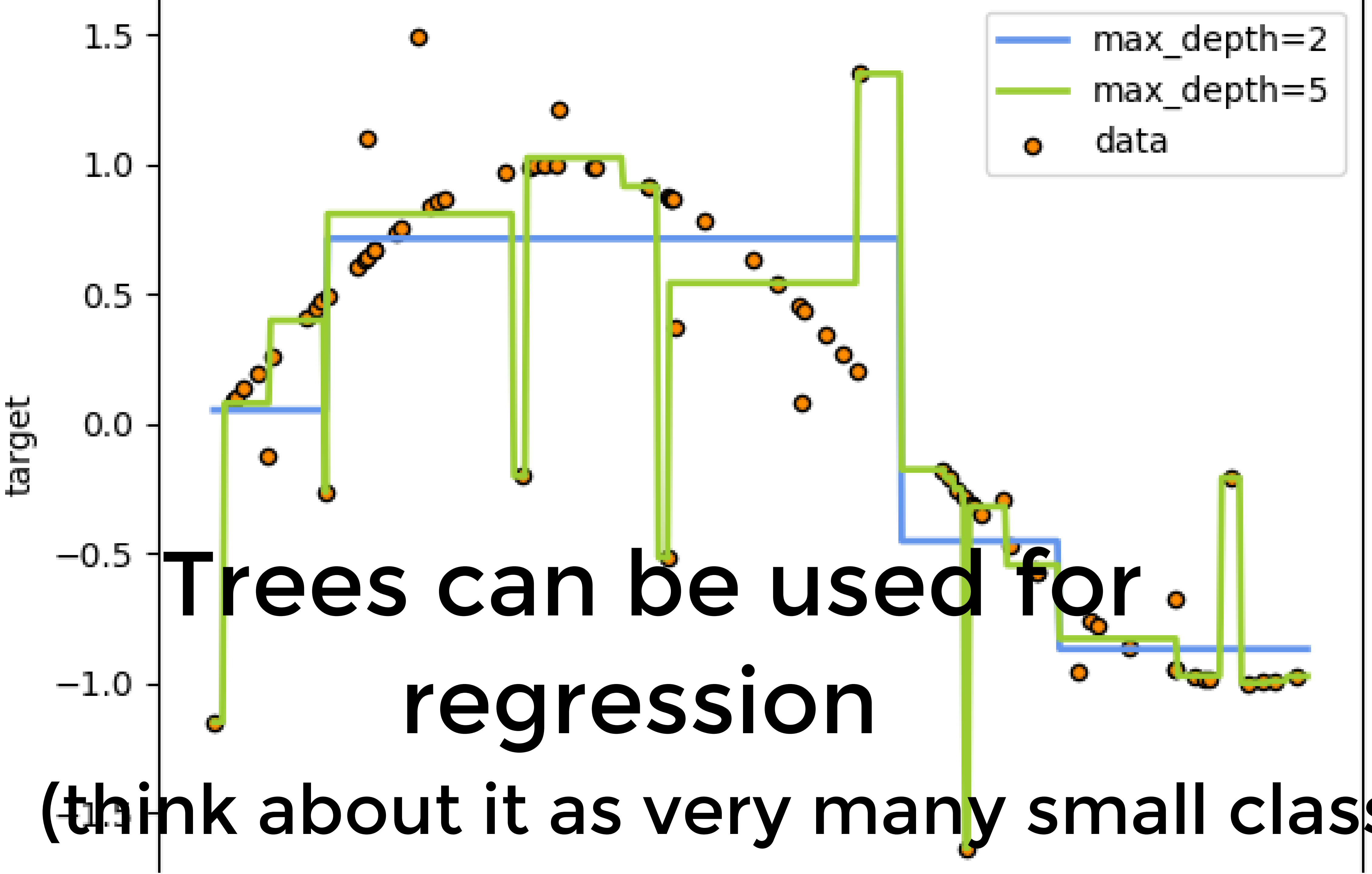
one-hot encoding

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

regression with trees

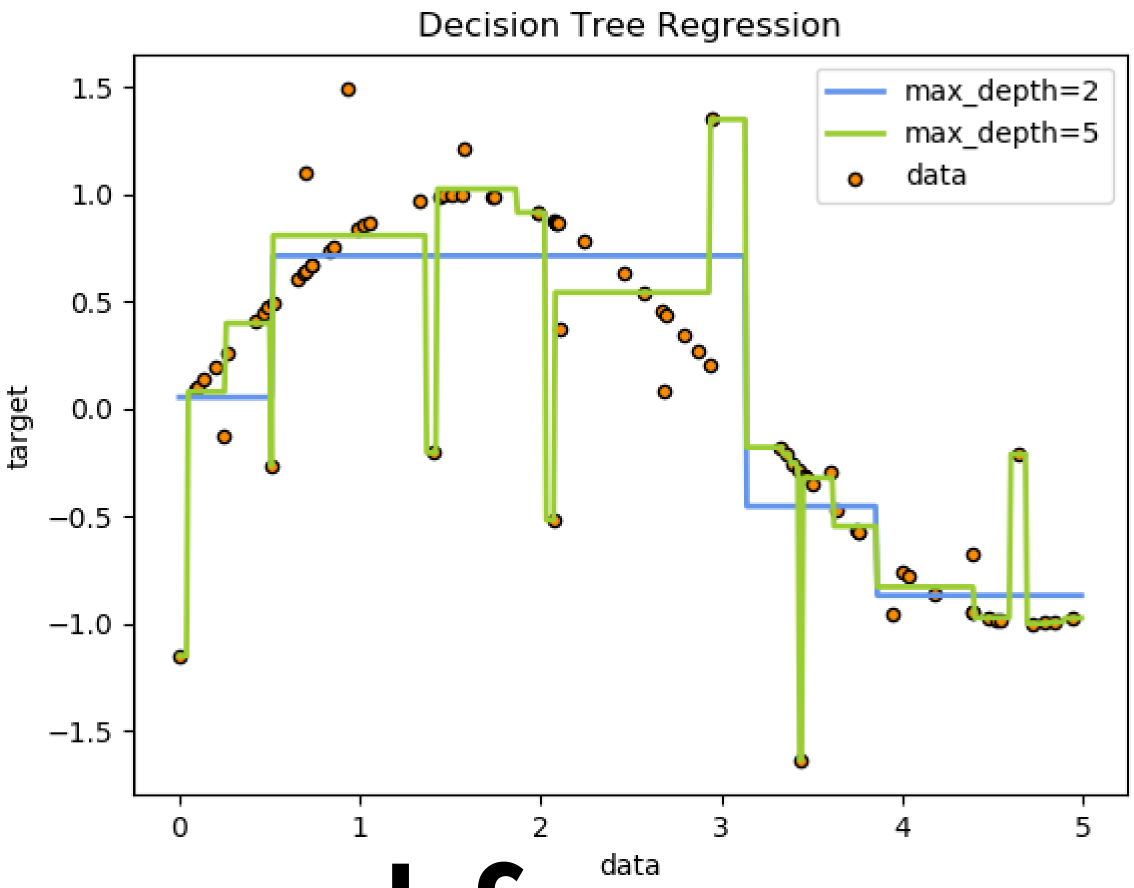
7

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

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

mean absolute error

$$L_1 = \Sigma \left|y_{true} - y_{predicted}\right|$$

ML model performance

8

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

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

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

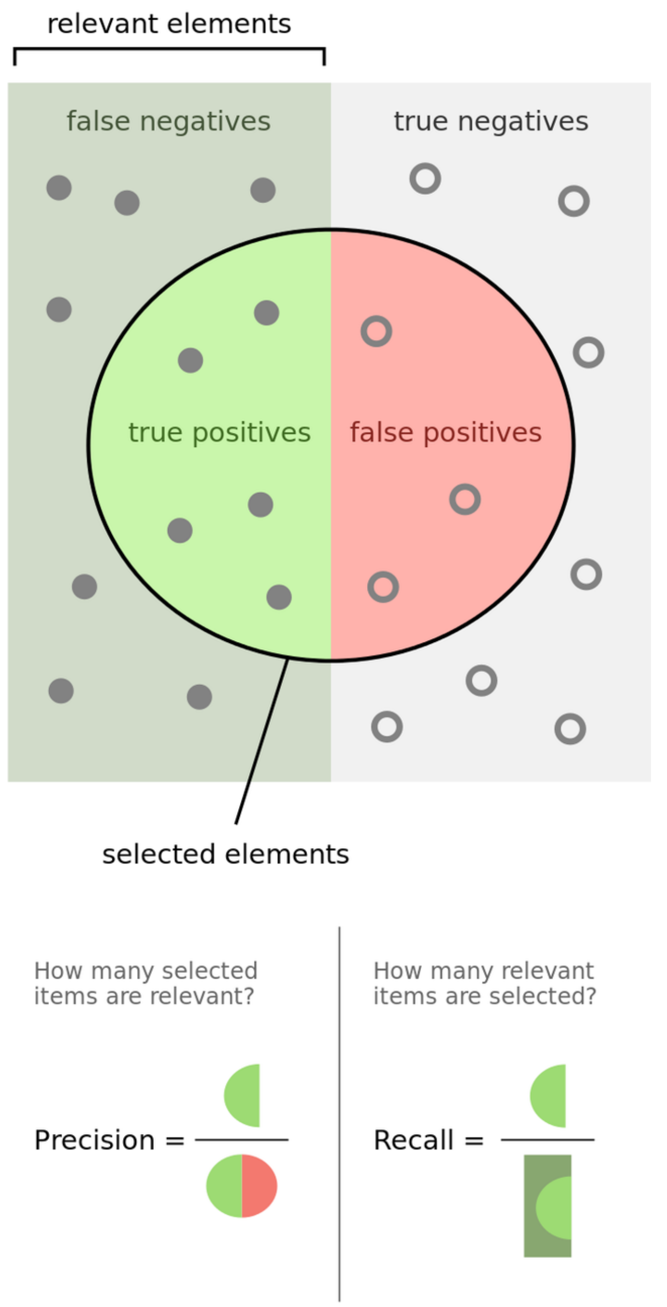
$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

TP=True Positive

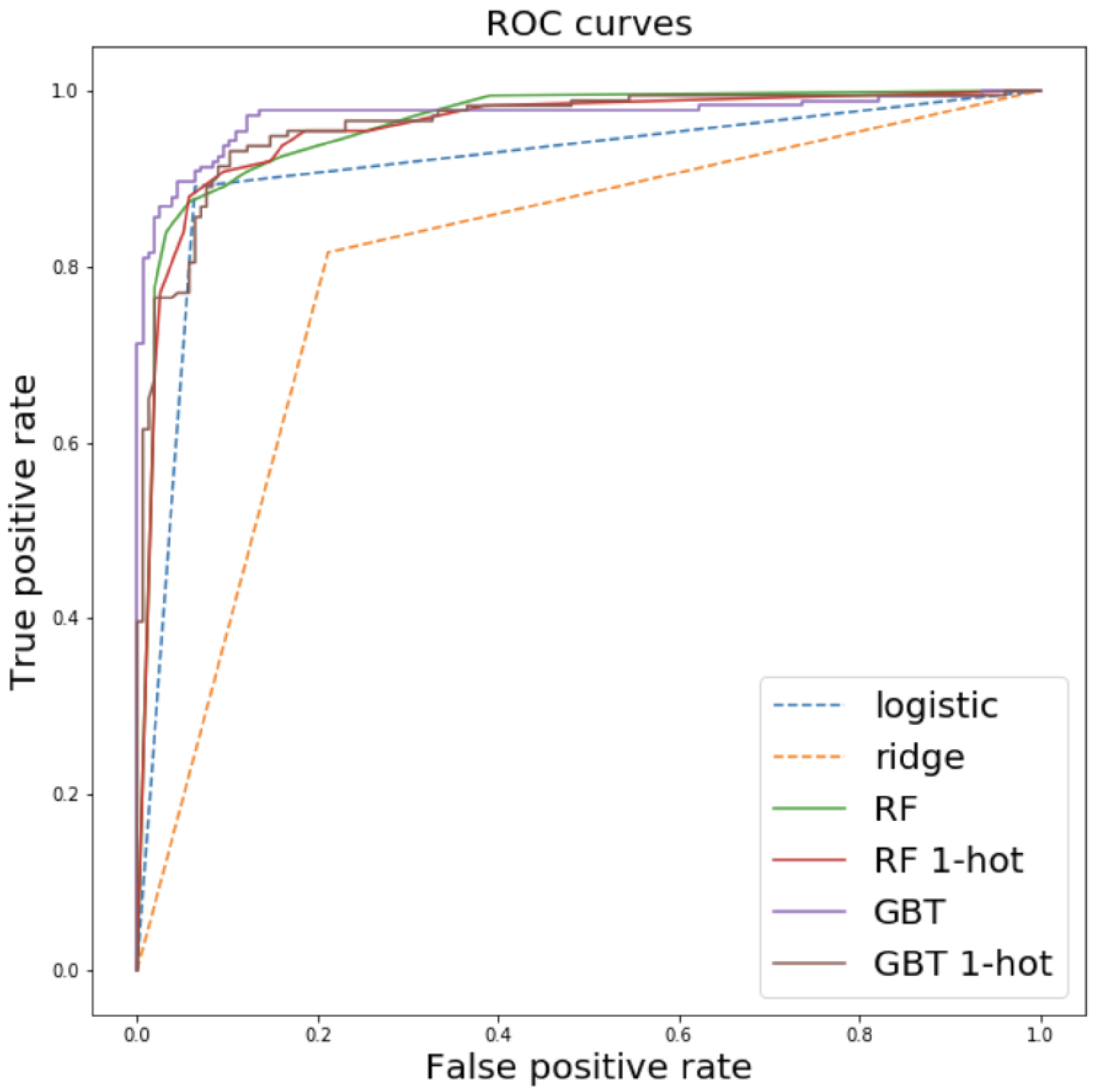
FP=False Positive

TN=True Negative

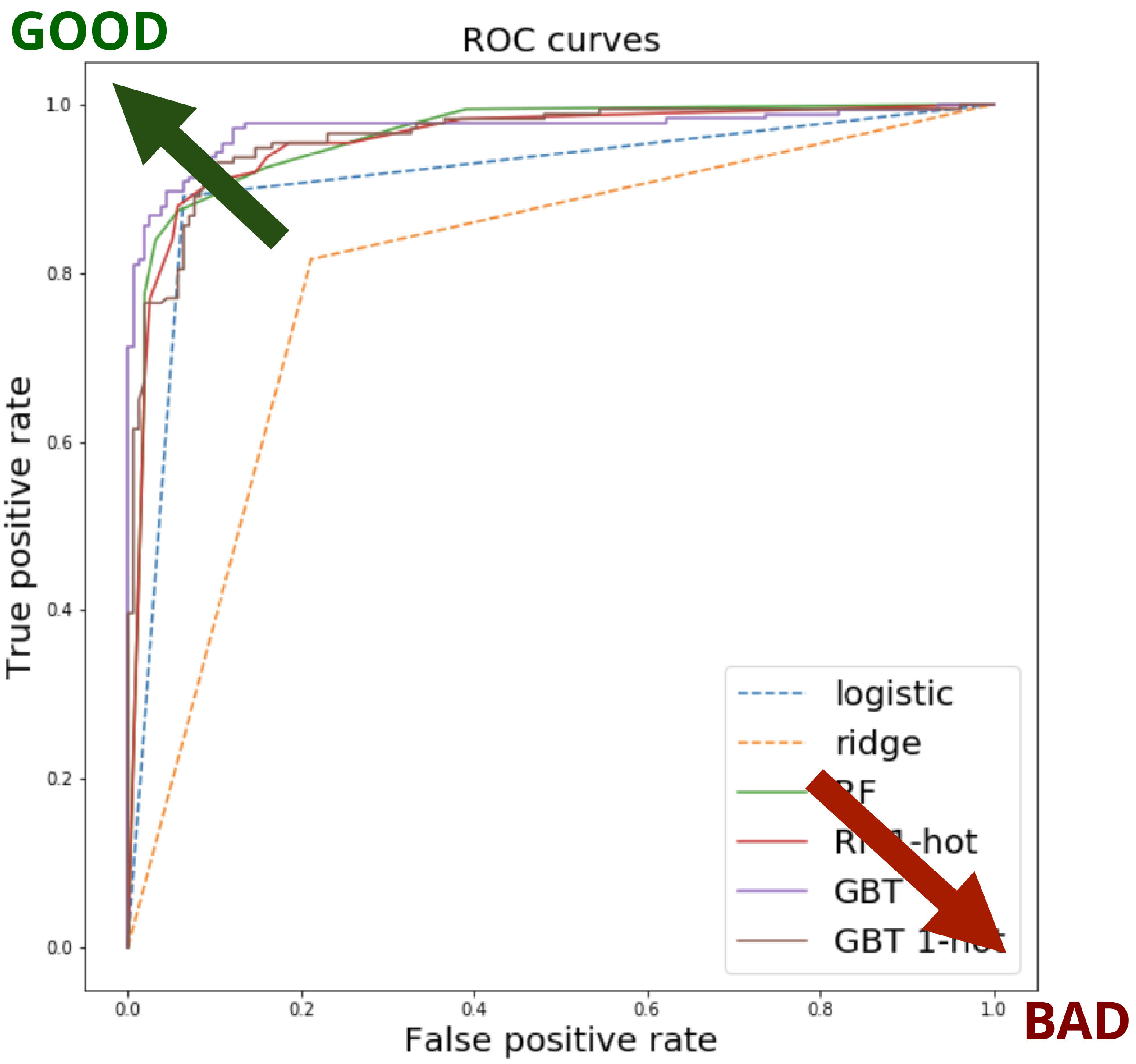
FN=False Positive



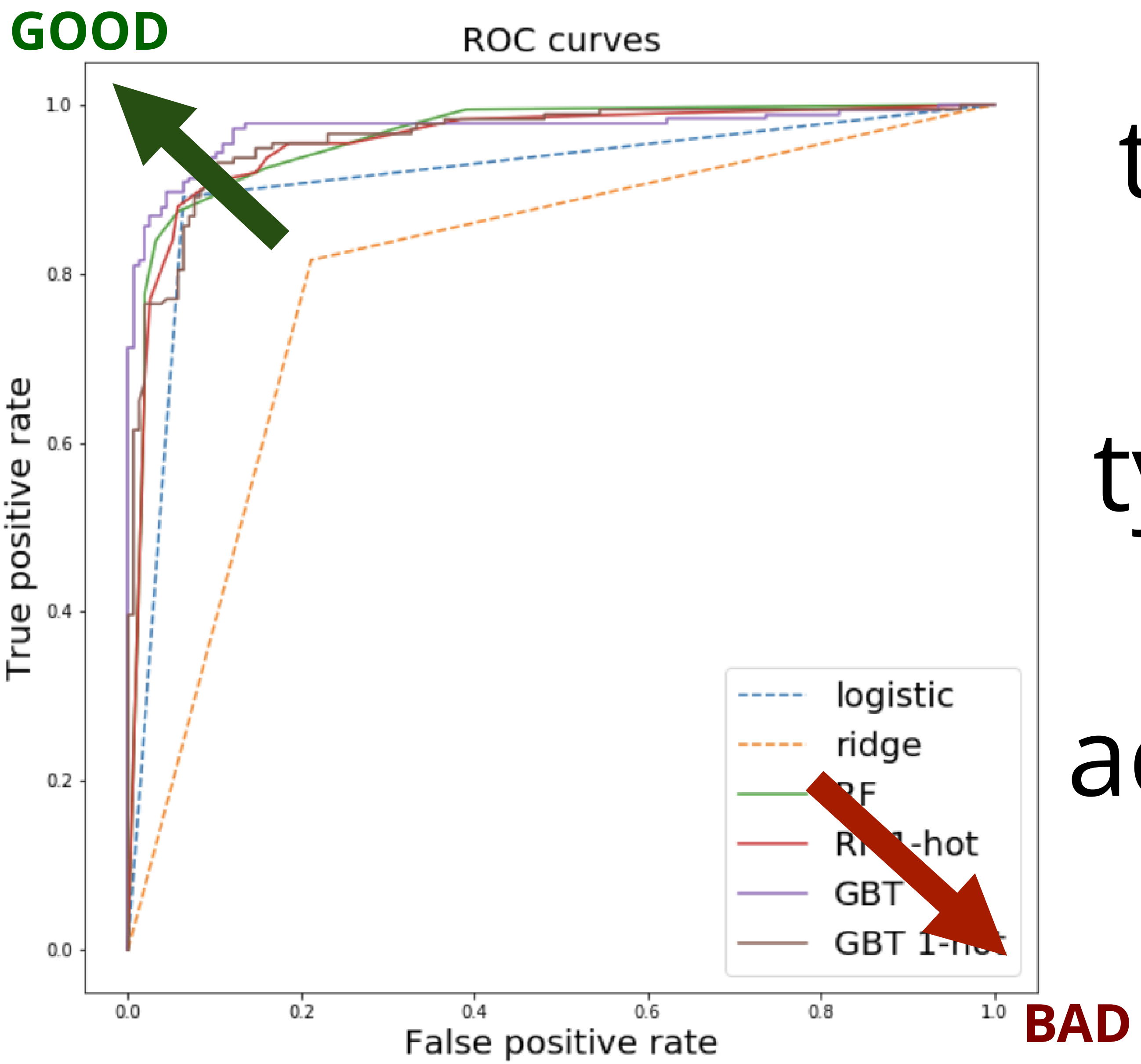
https://en.wikipedia.org/wiki/Precision_and_recall



Receiver operating characteristic



Receiver operating characteristic



tuning by changing *hyperparameters*

typically: probability threshold for acceptance of a class

Receiver operating characteristic

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

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

key concepts

encoding categorical variables:

variables have to be encoded as numbers for computers to understand them. You can encode categorical variables with integers or floating point but you implicitly impart an order. The standard is to ***one-hot-encode*** which means creating a binary (True/False) feature (column) for each category of a categorical variables but this *increases the feature space and generated covariance.*

model diagnostics for classifiers: Fraction of True Positives and False Positives are the metrics to evaluate classifiers. Combinations of those numbers include Accuracy (TP/ (TP+FP)), Precision (TP/(TP+FN)), Recall ((TP+TN)/(TP+TN+FP+FN)).

ROC curve: (TP vs FP) is a holistic metric of a model. It can be used to guide the choice of hyperparameters to find the "sweet spot" for your problem 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

reproduce this study on Urban metabolism

https://github.com/fedhere/PUS2020_FBianco/blob/master/HW8/HW8_RandomForest_instructions.ipynb

homework