

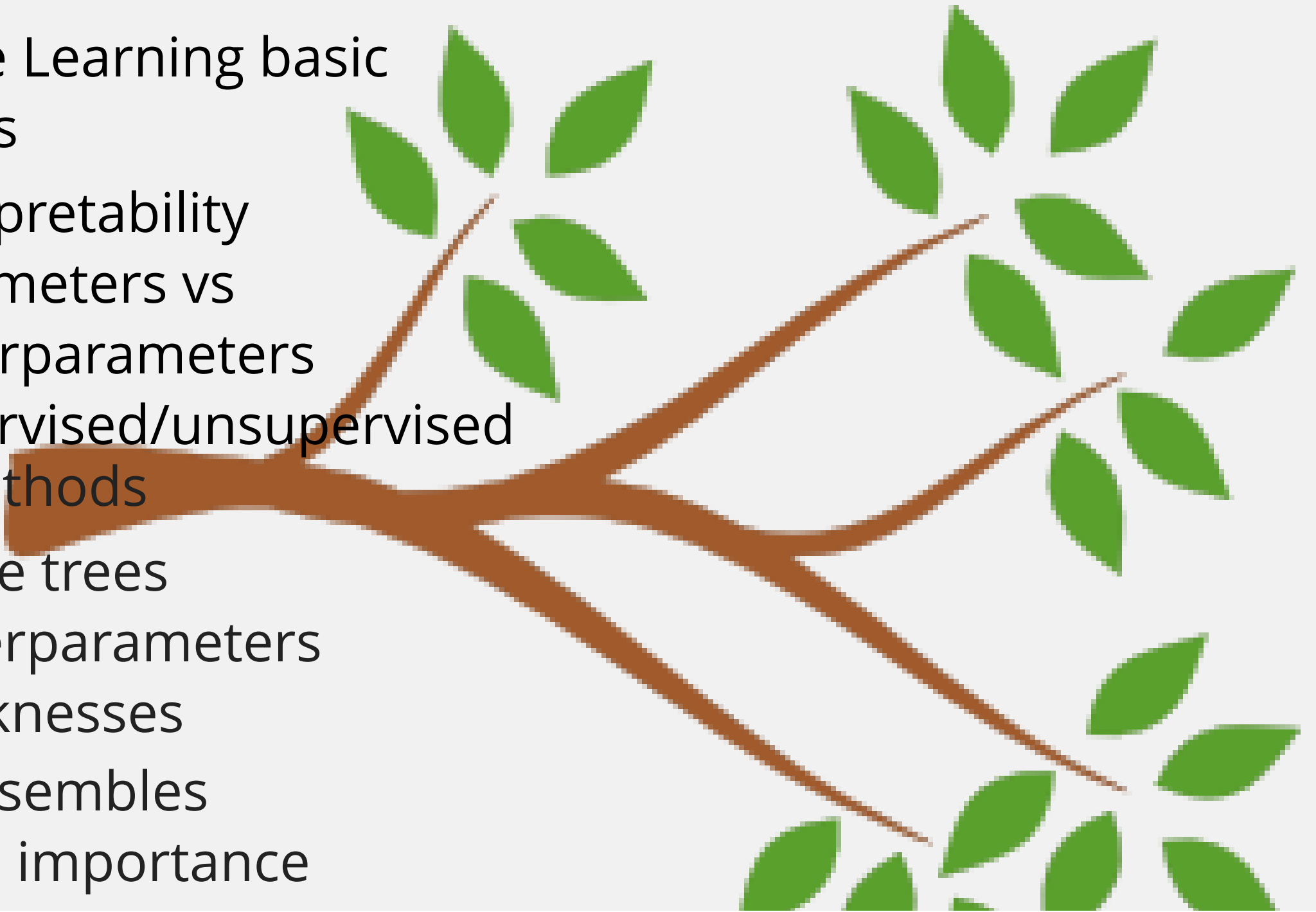
data science for (physical) scientists VIII

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



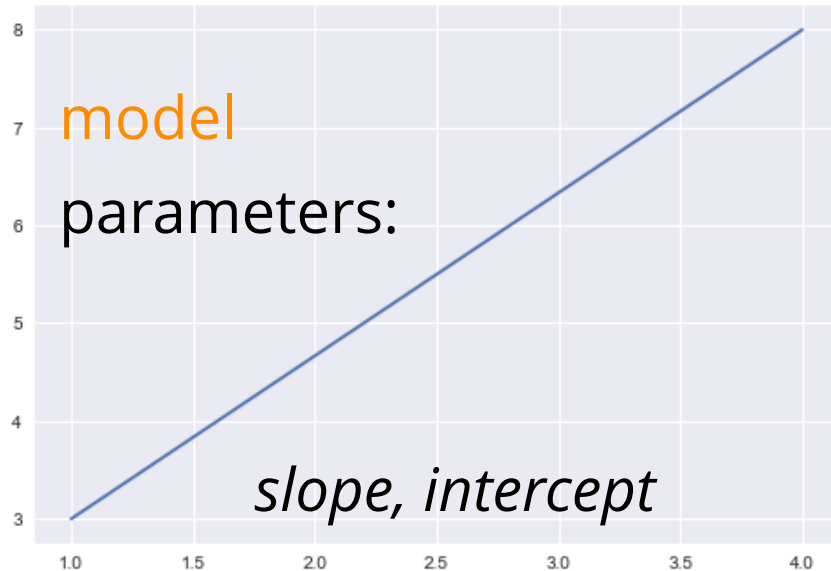


what is machine learning

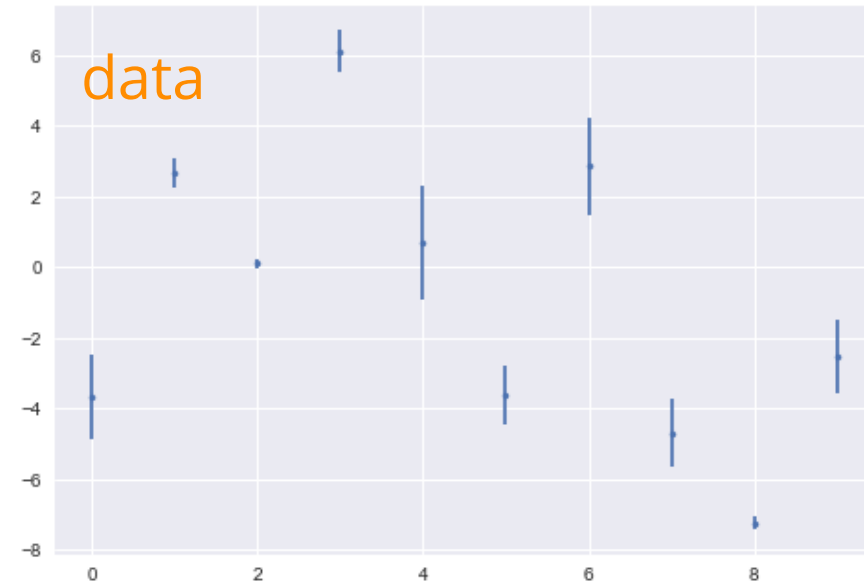
what is machine learning?

[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 parts

used to:

understand structure of feature space
classify based on examples,
regression (classification with infinitely
small classes)

general ML parts

should be interpretable: why?

ethical implications

predictive policing,

selection of conference participants.

general ML parts

should be interpretable: why?

ethical implications

predictive policing,

selection of conference participants.

causal connection

why the model made a choice?

which feature mattered?

general ML points

should be interpretable:
ethical implications

Why did Microsoft fund an Israeli firm that surveils West Bank Palestinians?

Microsoft committed to protecting democratic freedoms. Then it funded an Israeli facial recognition firm that secretly watched West Bank Palestinians.



— If Microsoft wants to safeguard “democratic freedoms,” why did it fund an Israeli facial recognition firm involved in secret military surveillance of Palestinians? Brian Stauffer / for NBC News

MAST @MAST_News · Jun 4

The Inclusive Astronomy 2 Conference will be happening at STScI in Baltimore October 14-15. Registration opens later this month. If interested, visit the conference webpage for more information: outerspace.stsci.edu/display/IA2

STScI Outerspace Site Home Search

Inclusive Astronomy 2

Inclusive Astronomy 2 Conference

October 14-15, 2019

Space Telescope Science Institute
Baltimore, MD

It has been four years since the 2015 Nashville Inclusive Astronomy meeting, an event that brought astronomers together with sociologists, policy makers, and leaders in the field to discuss issues affecting underrepresented groups in astronomy. The Nashville Recommendations, which emphasize equity and intersectionality, build upon a rich history of work to broaden participation and improve climates.

We now have the opportunity to bring together the astronomy community to discuss the current state of the profession and make recommendations for the 2020s and beyond. Specifically, we will discuss community expectations on inclusivity and representation, evaluate our progress towards meeting equity goals, and address the needs of marginalized groups in the workforce. We will advance these broad goals by focusing on barriers in professional development (e.g., training, jobs, promotion, tenure) and barriers to accessing resources (e.g.,

general ML parts

ML models 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

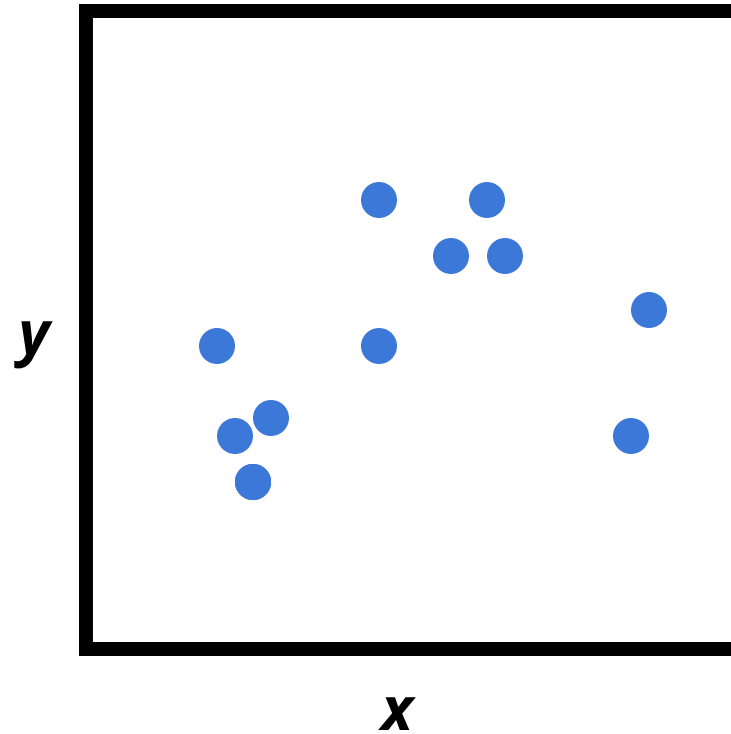
classification vs clustering

1

clustering vs classifying

unsupervised

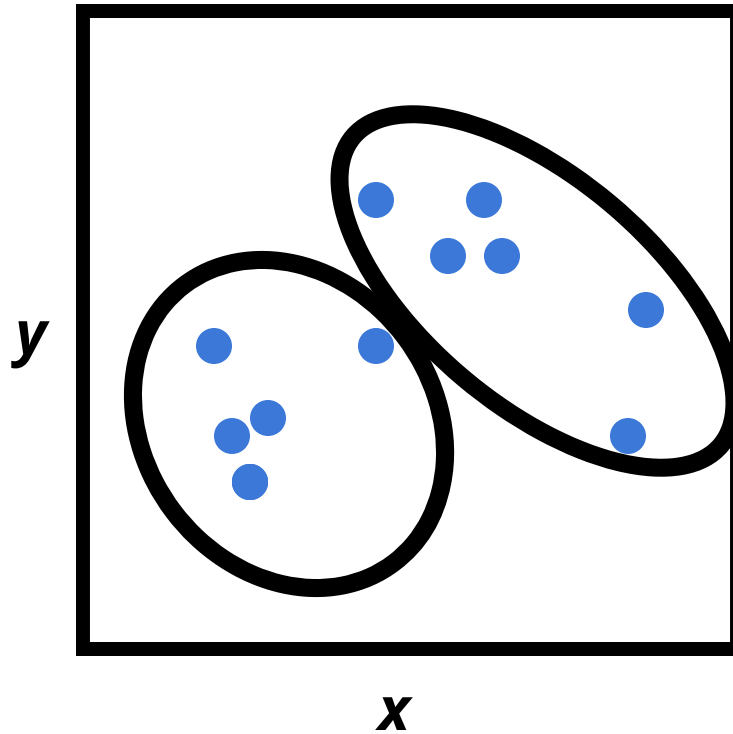
observed **features:**
 (\vec{x}, \vec{y})



clustering vs classifying

unsupervised

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



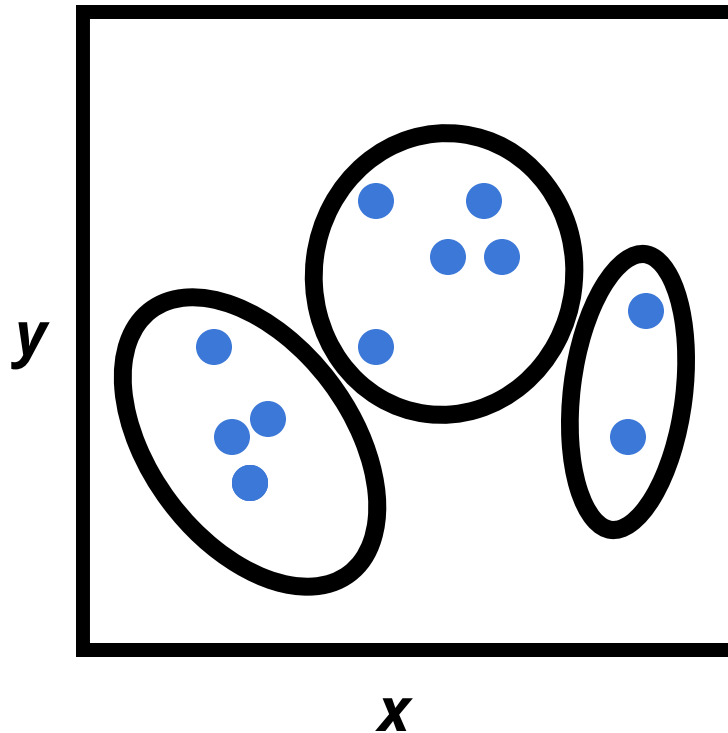
clustering vs classifying

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

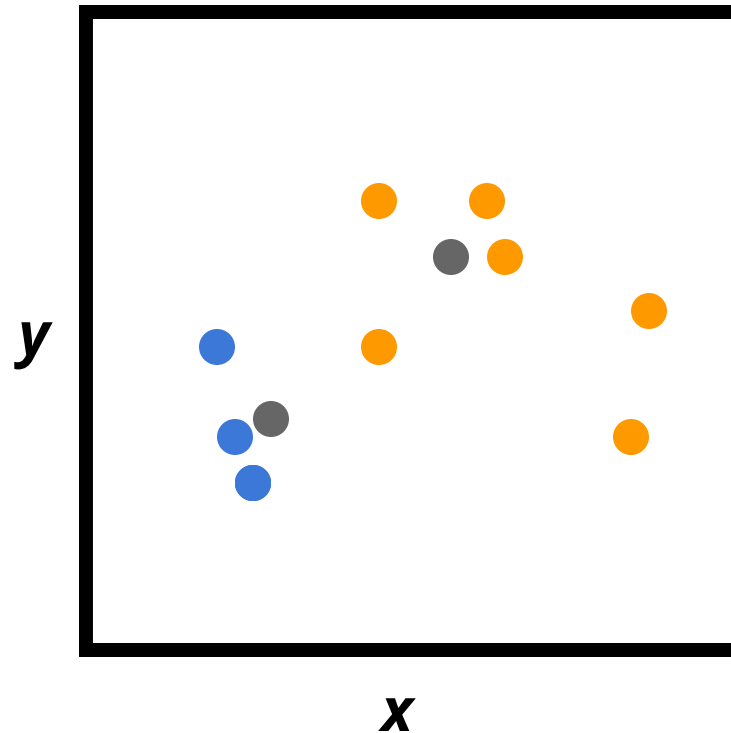
clustering vs classifying

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})



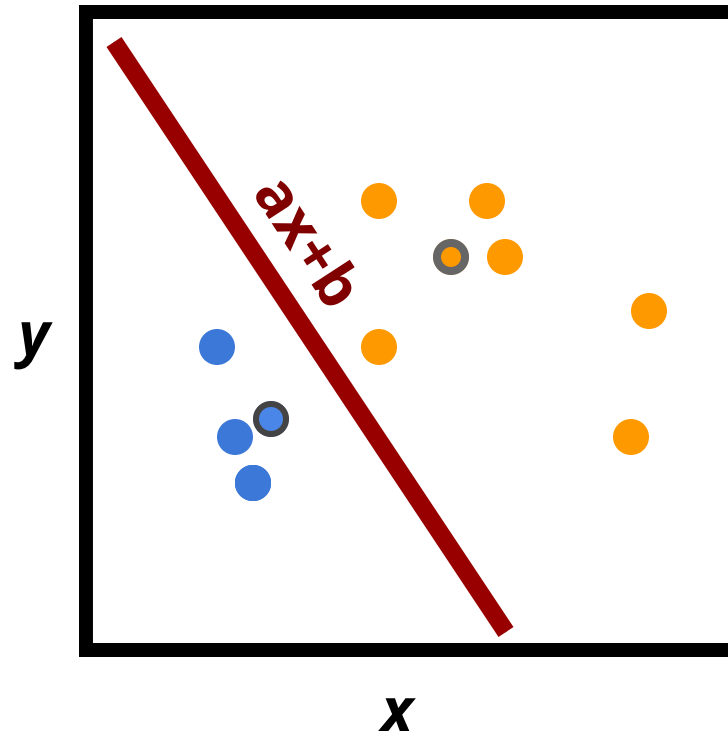
target **features:**
 (\overrightarrow{color})

models typically return a partition of the space

clustering vs classifying

unsupervised *supervised*

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



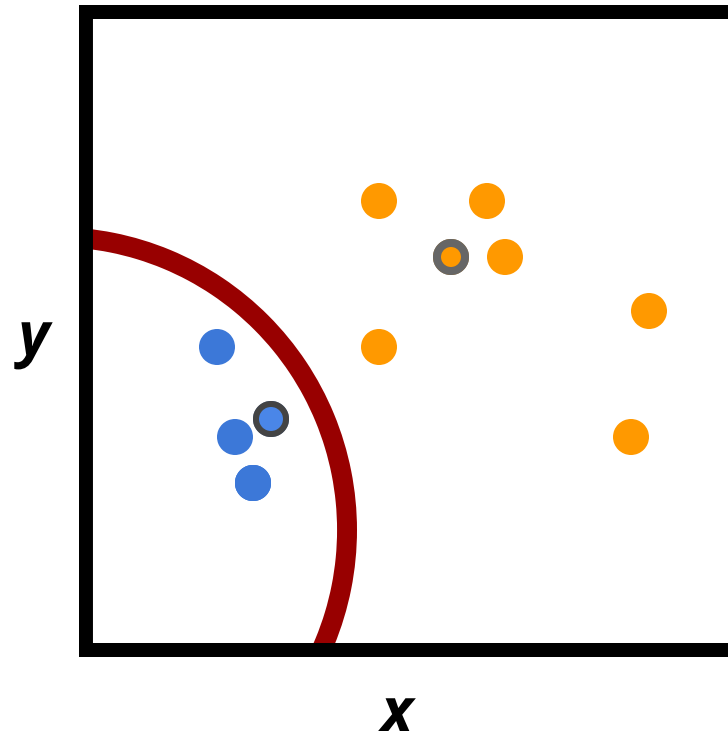
target **features:**
 $\overrightarrow{(color)}$

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

clustering vs classifying

unsupervised *supervised*

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



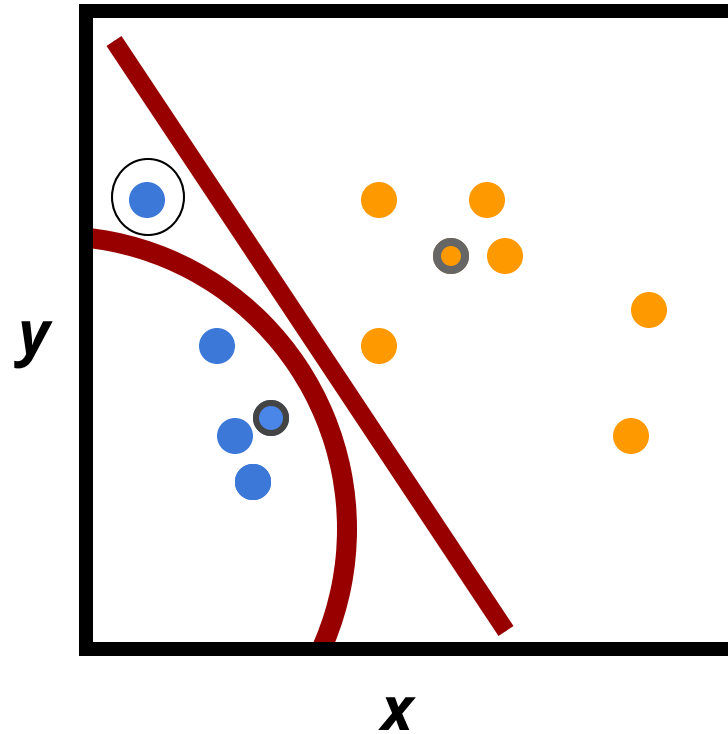
target **features:**
 $\overrightarrow{(color)}$

```
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})



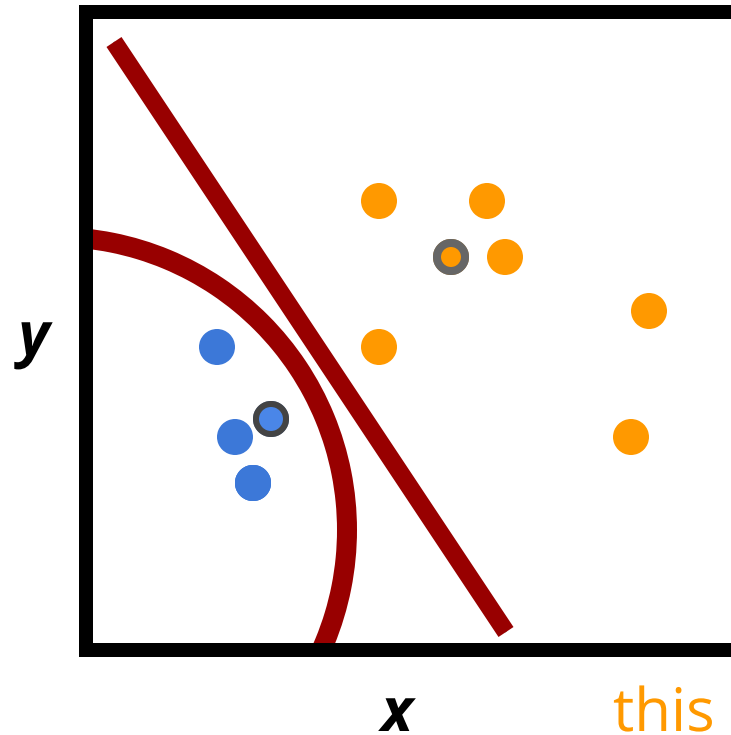
target **features:**
 (\overrightarrow{color})

```
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

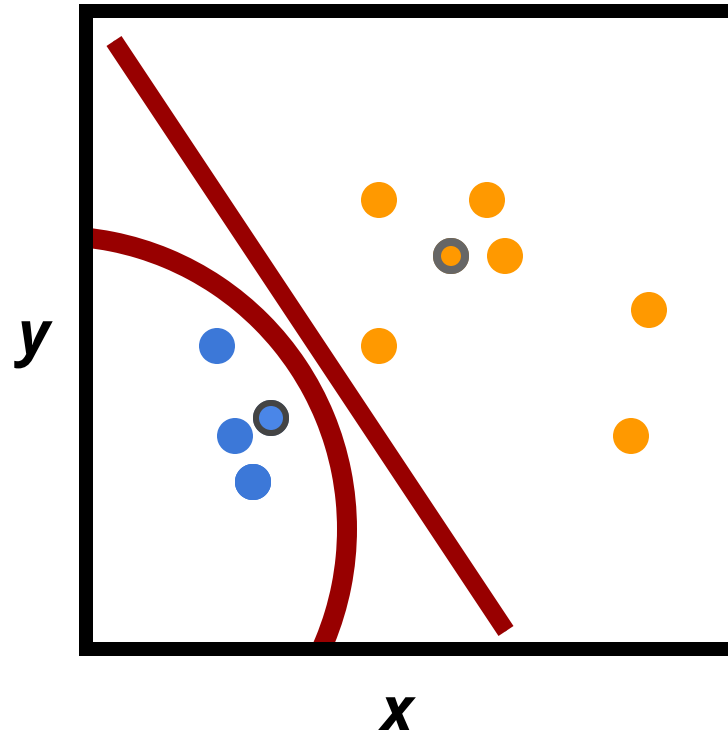
supervised ML: classification

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:**
 $\overrightarrow{(color)}$

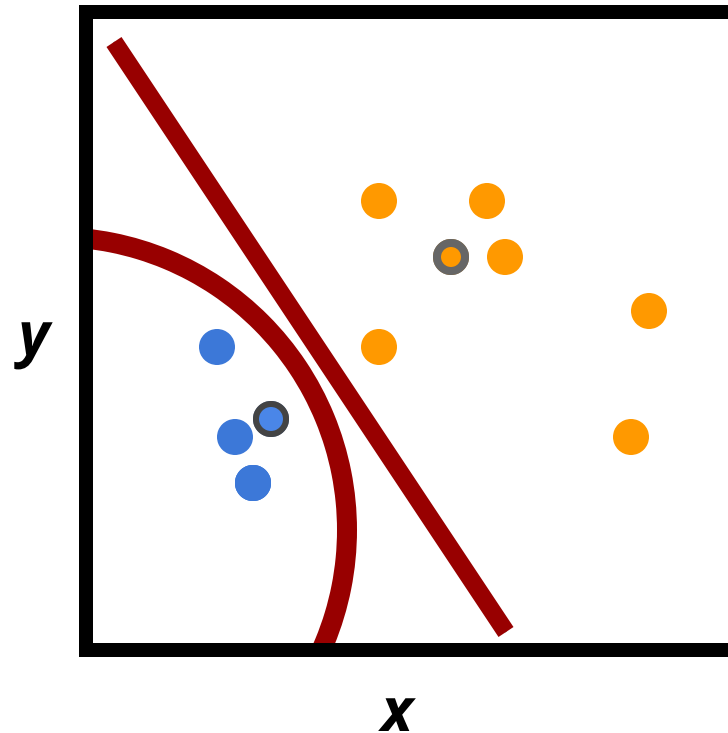
supervised ML: classification

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})

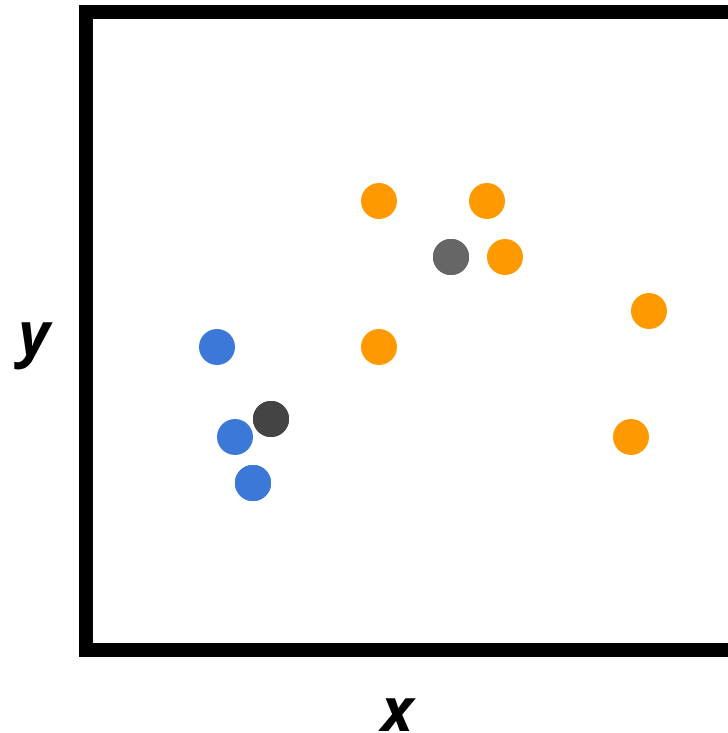
supervised ML: classification

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})

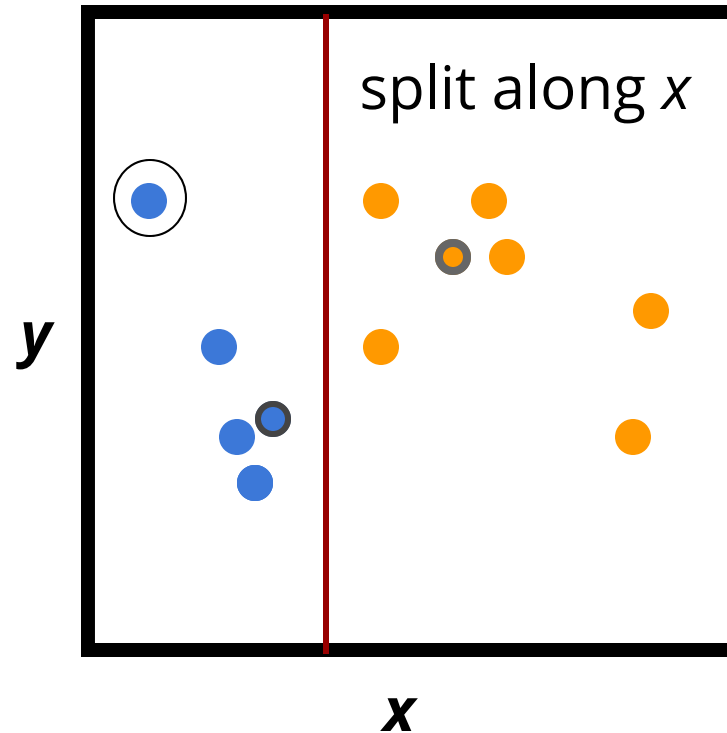
supervised ML: classification

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
```

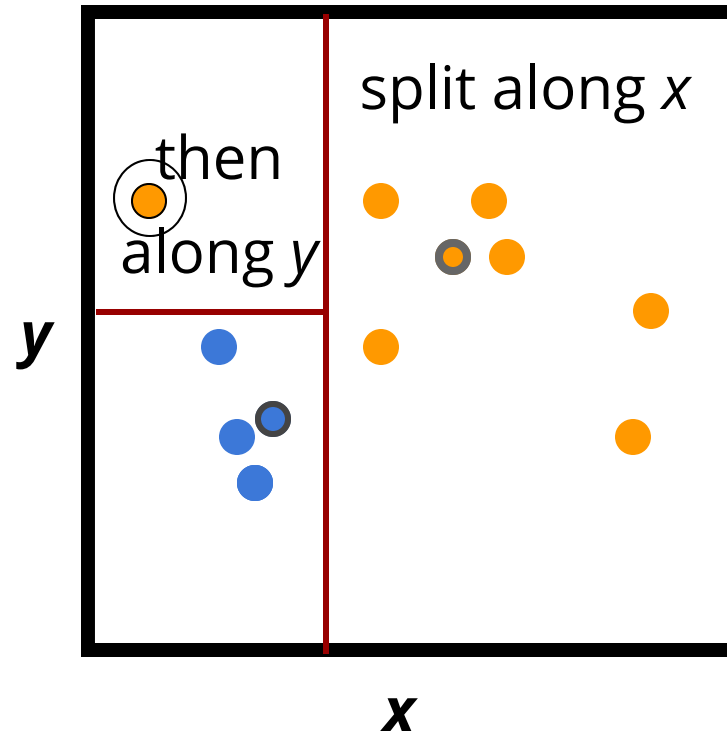
supervised ML: classification

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
```

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 ensemble methods)
- Tendency to overfit
- (not really easily interpretable after all...)

single tree

1

Application:
a robot to predict surviving the Titanic

(Kaggle)

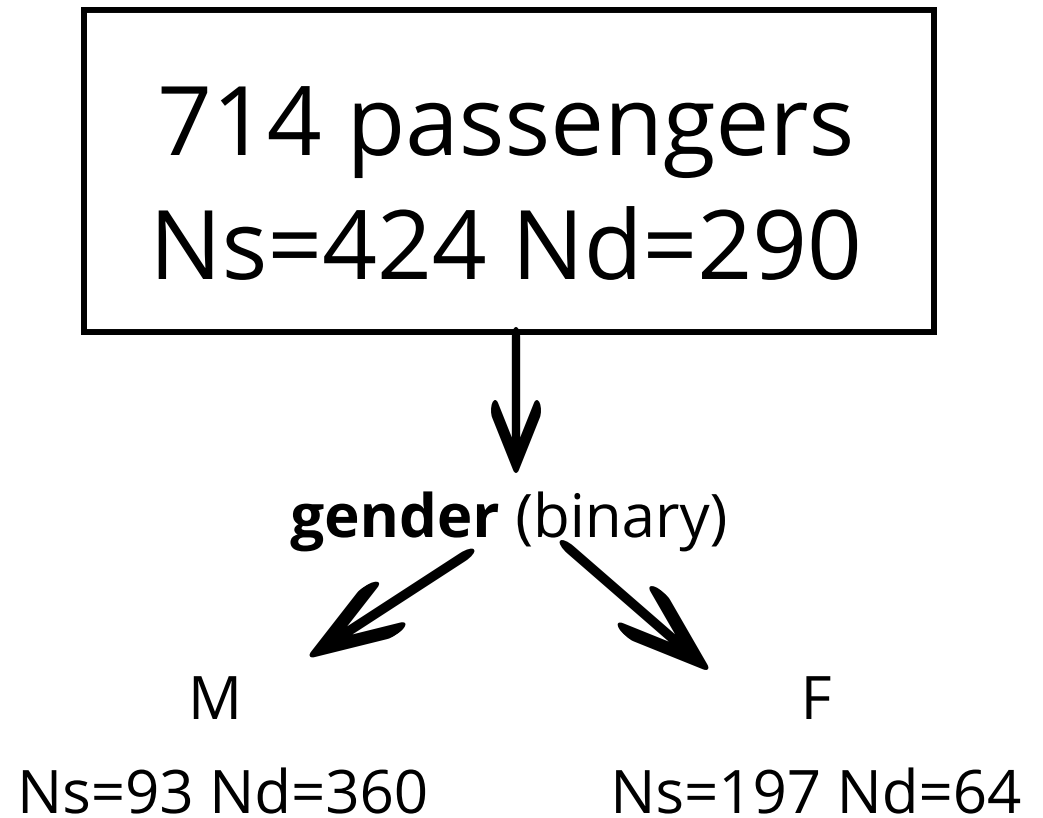
<https://www.kaggle.com/c/titanic>

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)



Application:
a robot to predict surviving the
Titanic

(Kaggle)

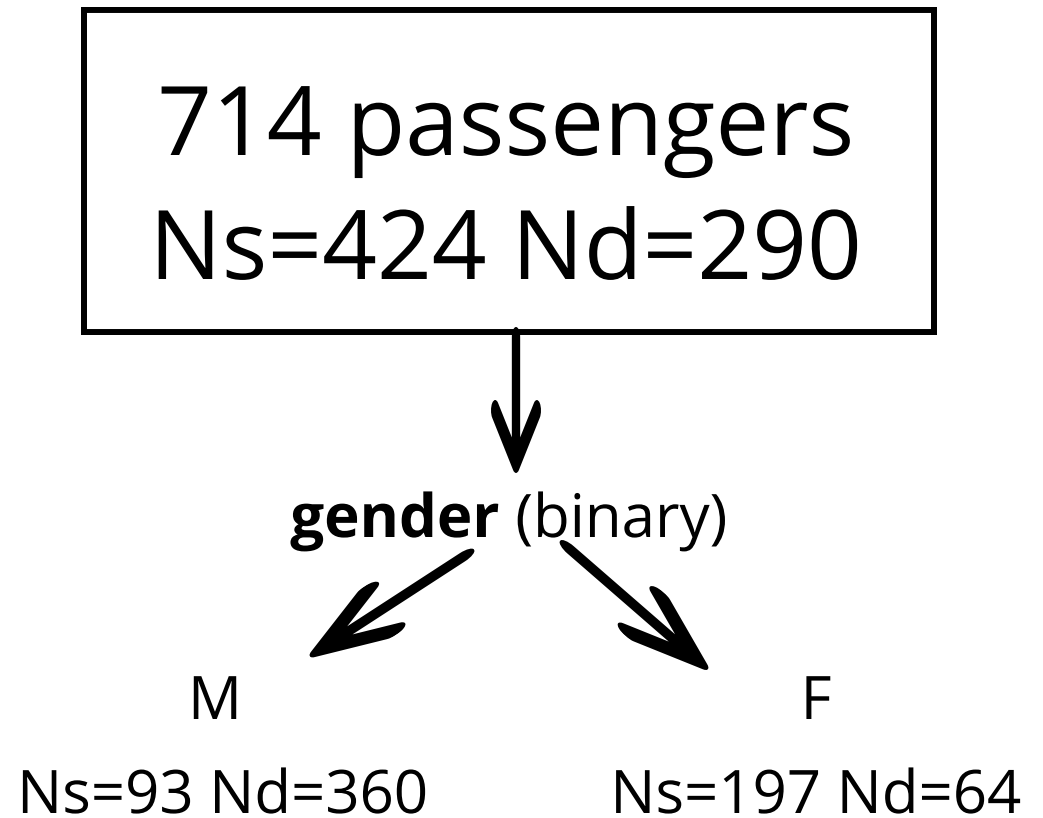
<https://www.kaggle.com/c/titanic>

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)



optimize over purity:

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

Application:
a robot to predict surviving the
Titanic

(Kaggle)

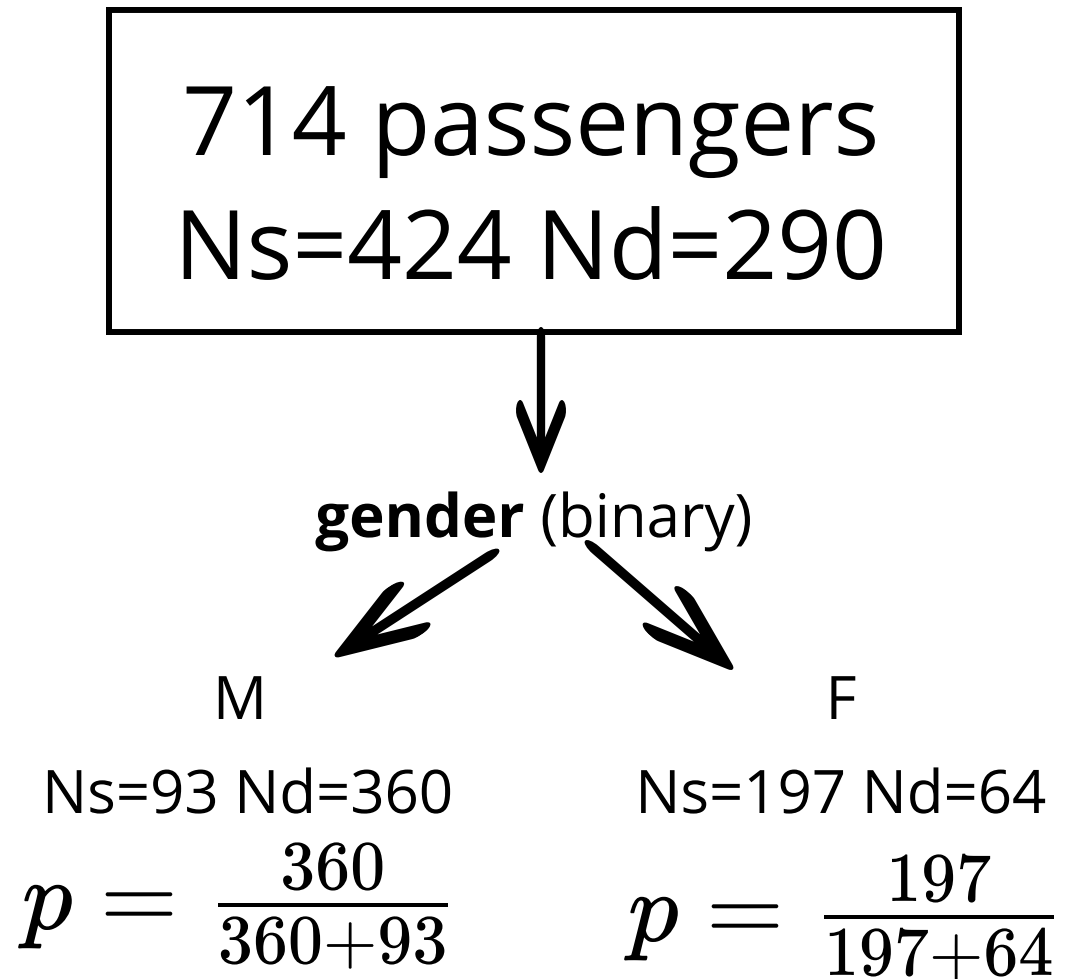
<https://www.kaggle.com/c/titanic>

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)



optimize over purity:

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

Application:
a robot to predict surviving the
Titanic

(Kaggle)

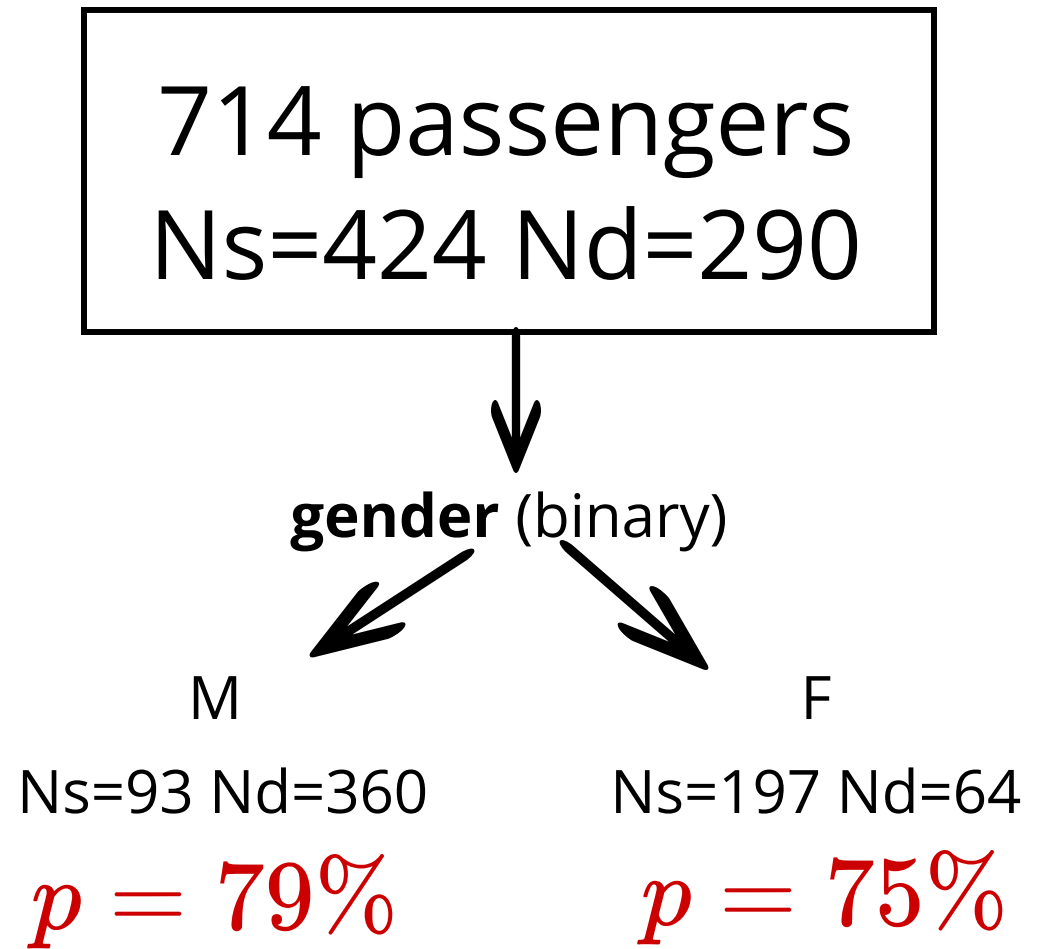
<https://www.kaggle.com/c/titanic>

features:

- gender
- ticket class
- age

target variable:

-> survival (y/n)



optimize over purity:

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

Application:
a robot to predict surviving the
Titanic

(Kaggle)

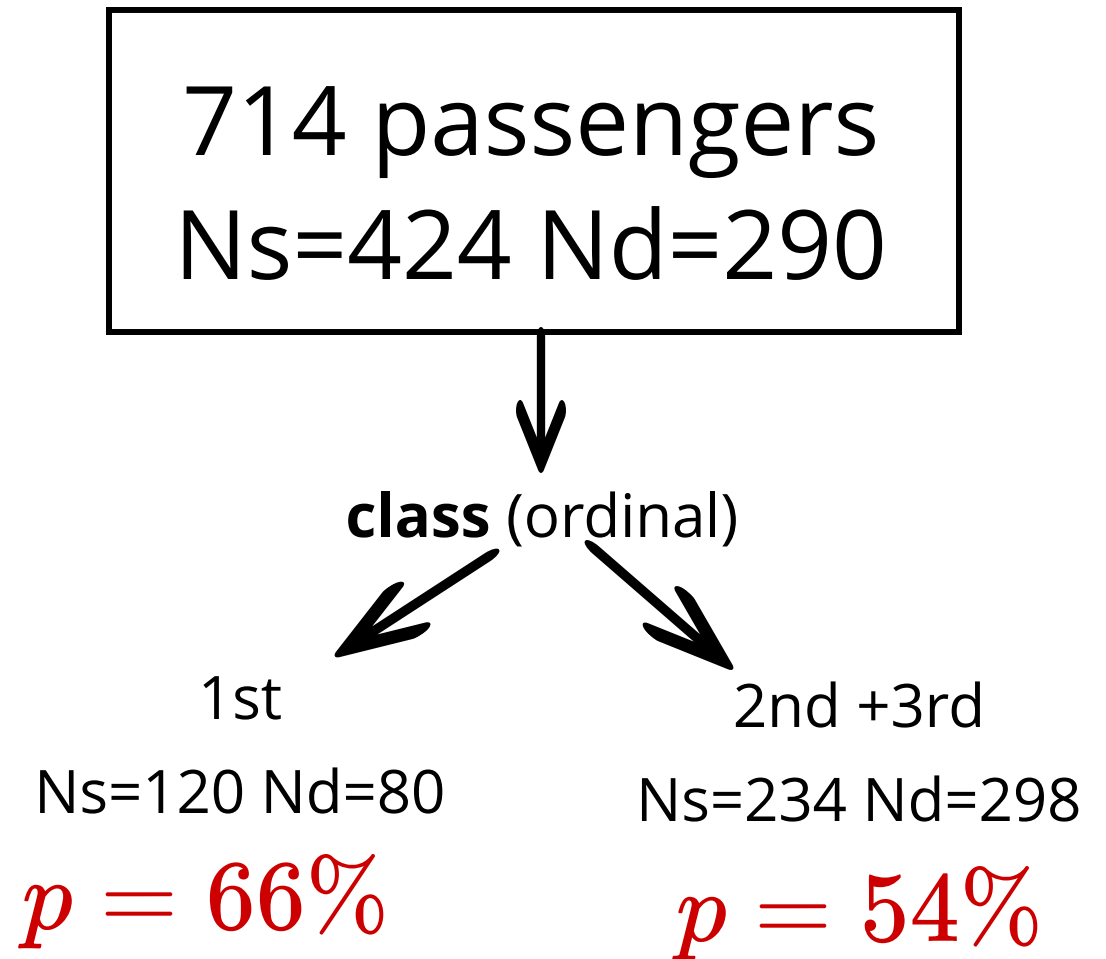
<https://www.kaggle.com/c/titanic>

features:

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

target variable:

-> survival (y/n)



Application:
a robot to predict surviving the Titanic

(Kaggle)

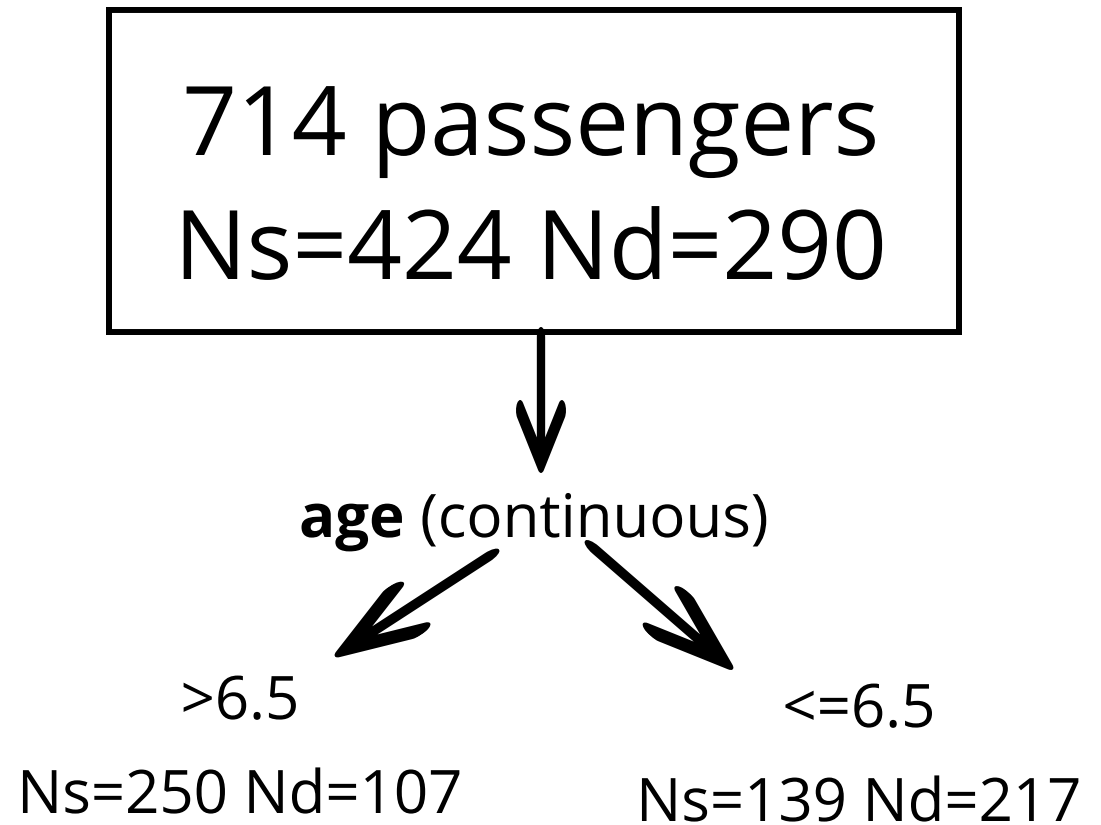
<https://www.kaggle.com/c/titanic>

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)

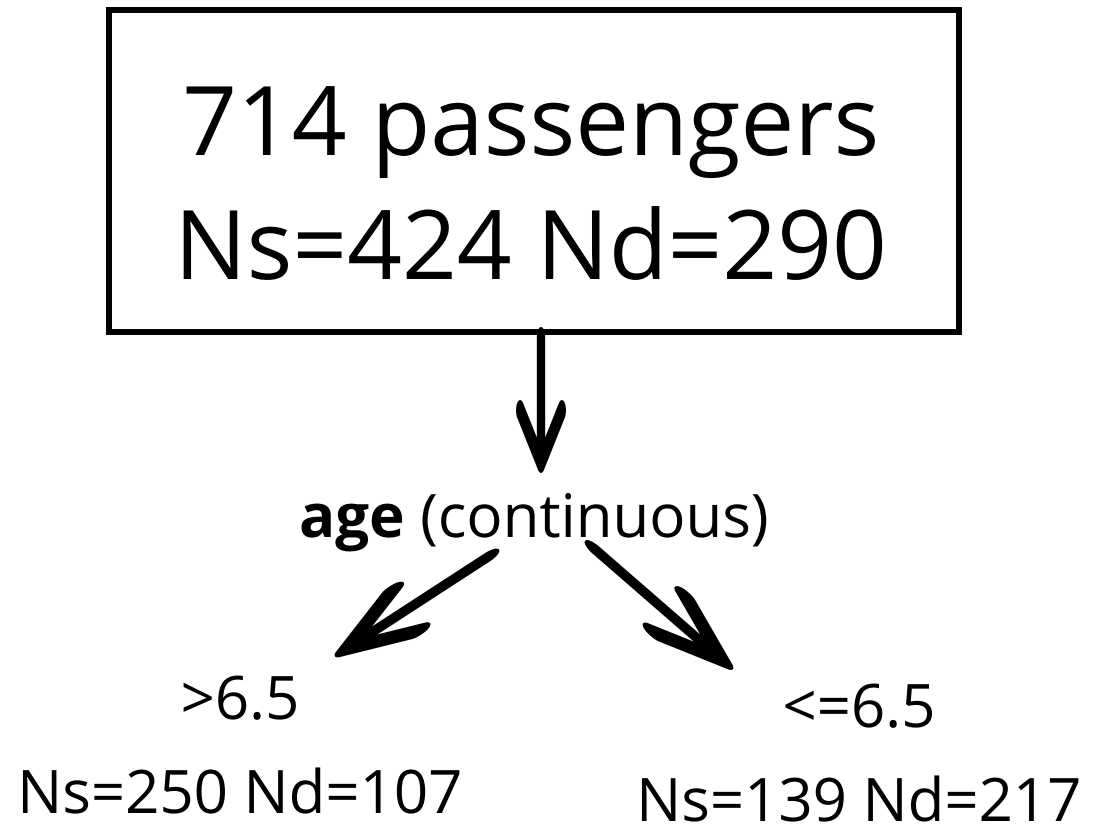
<https://www.kaggle.com/c/titanic>

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)

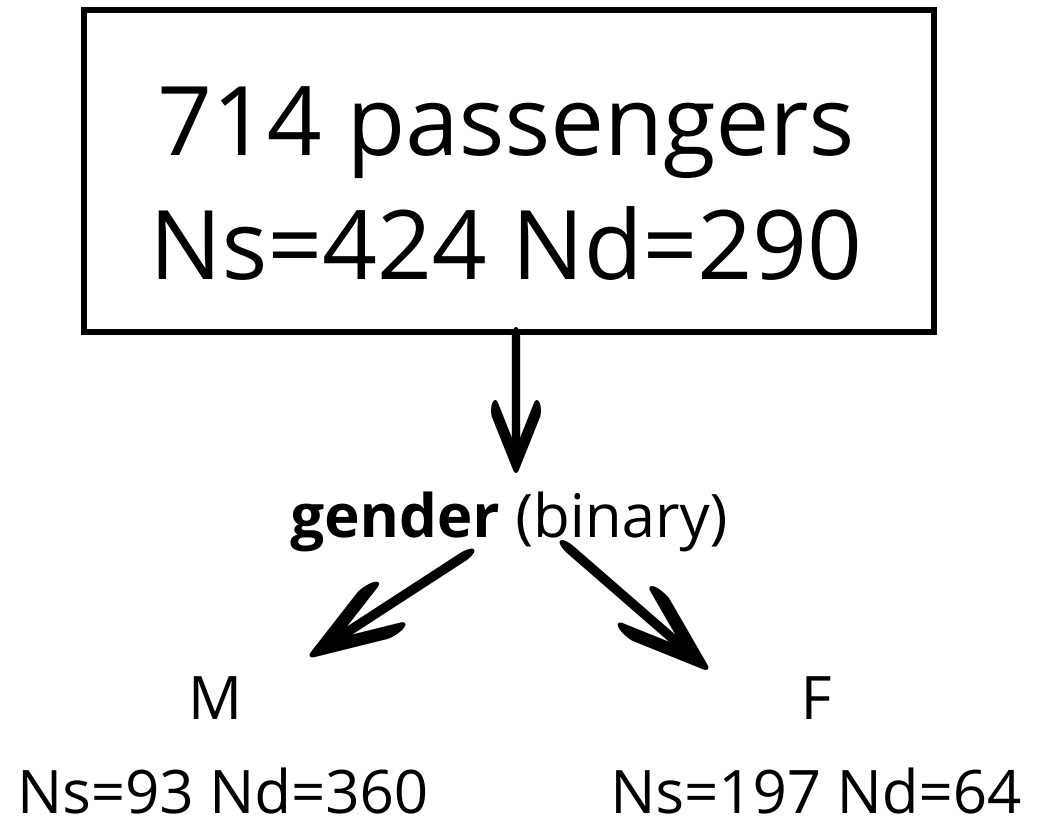
<https://www.kaggle.com/c/titanic>

features:

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

target variable:

-> survival (y/n)



Application:
a robot to predict surviving the
Titanic

(Kaggle)

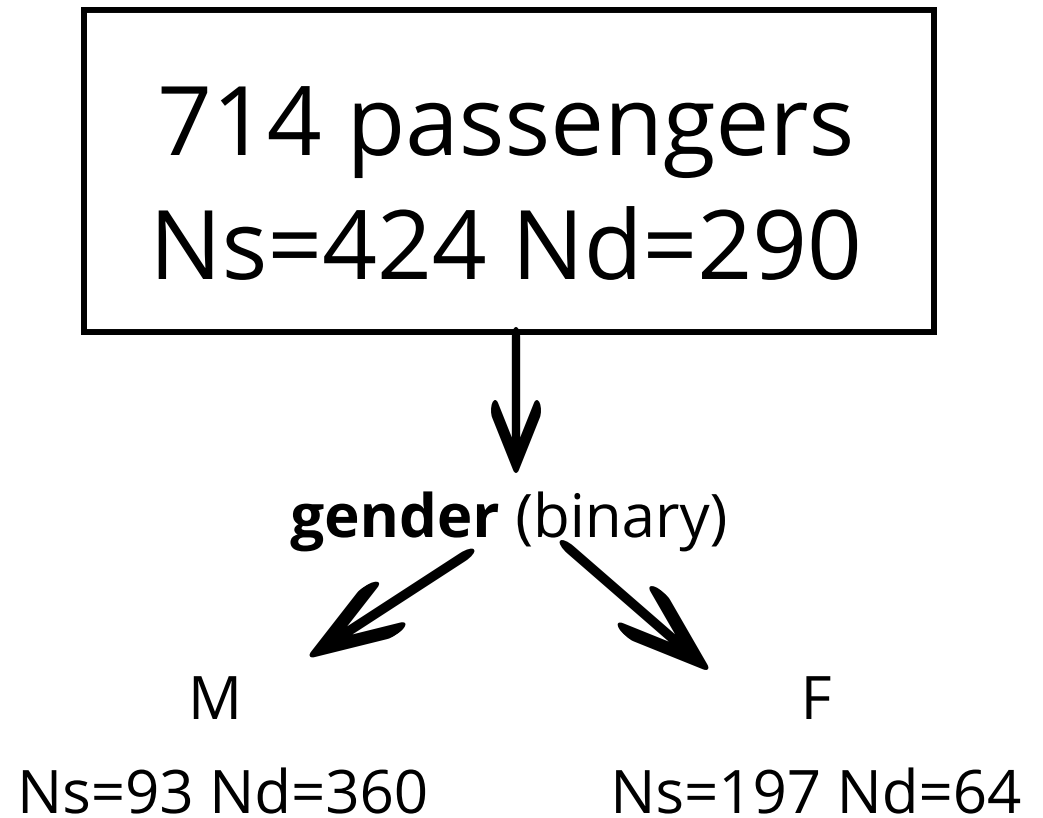
<https://www.kaggle.com/c/titanic>

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)

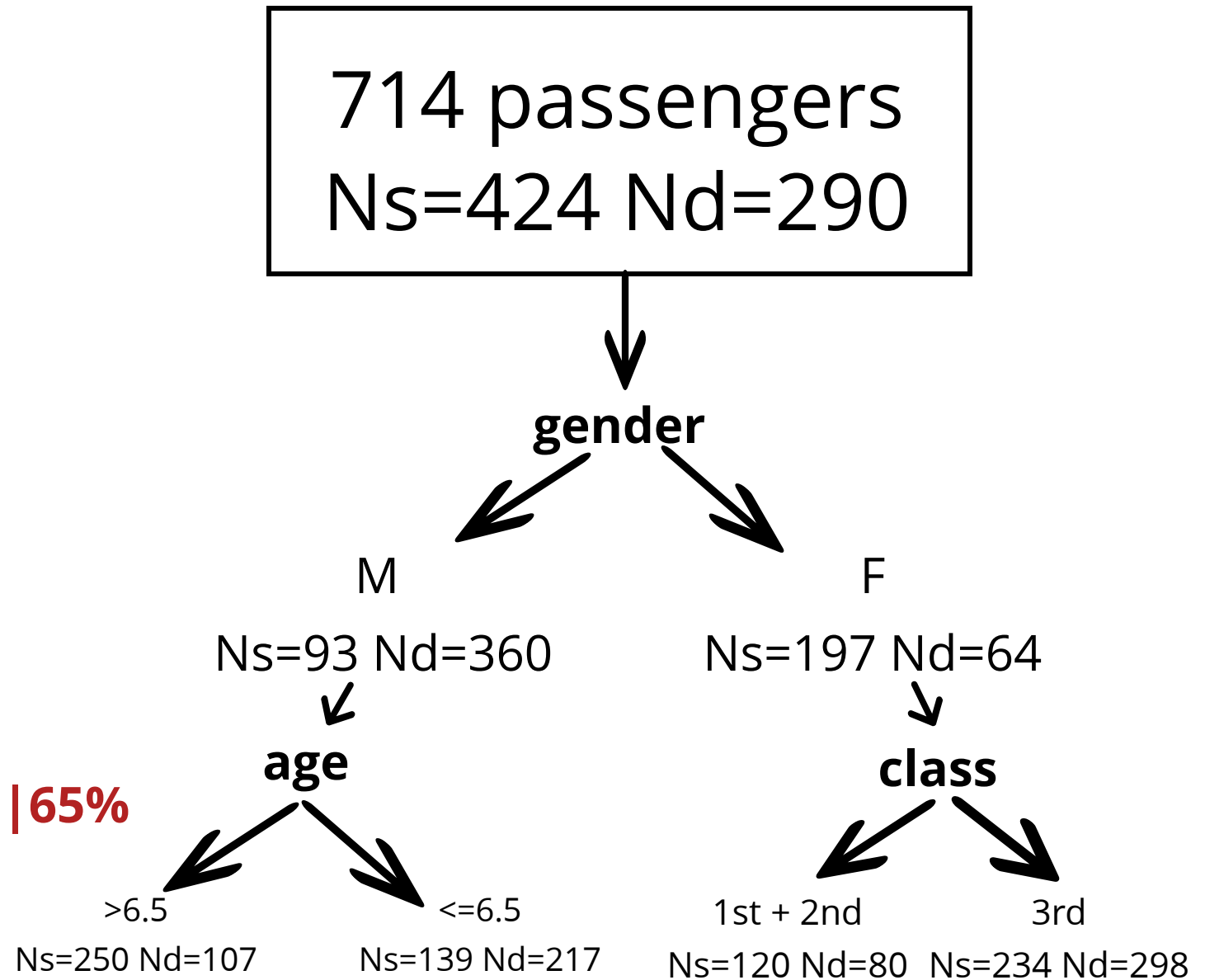
<https://www.kaggle.com/c/titanic>

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)

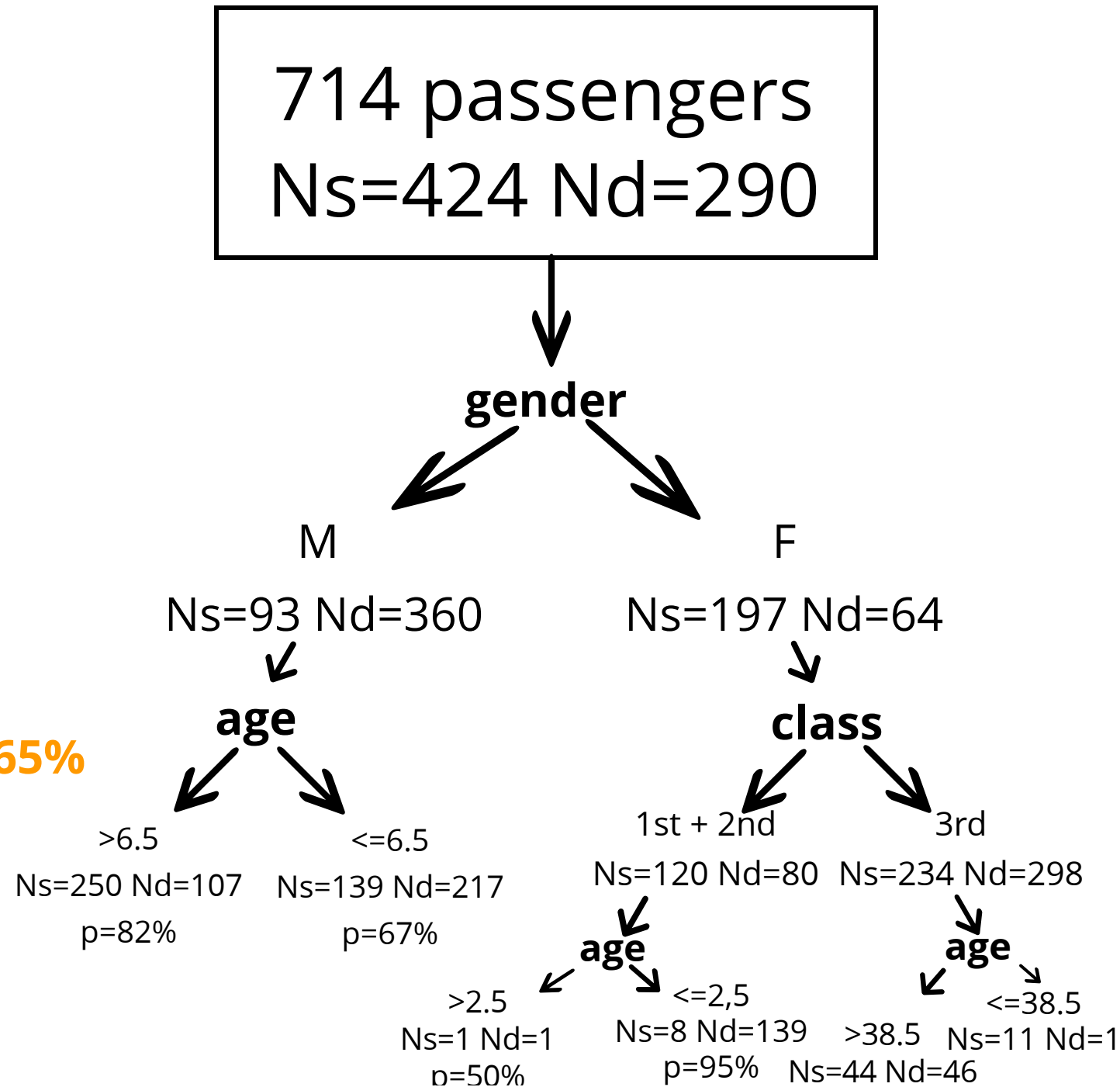
<https://www.kaggle.com/c/titanic>

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)

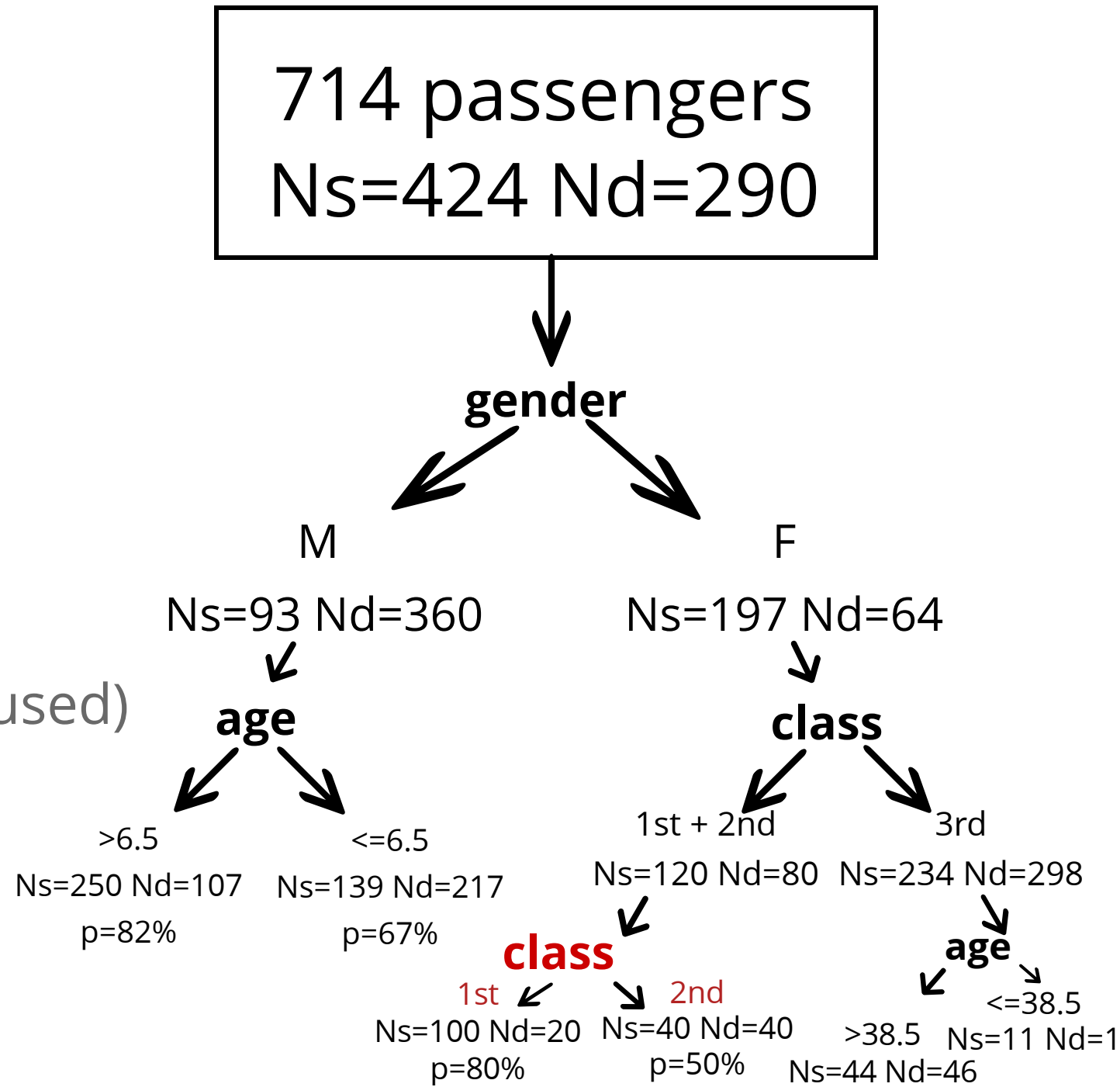
<https://www.kaggle.com/c/titanic>

features:

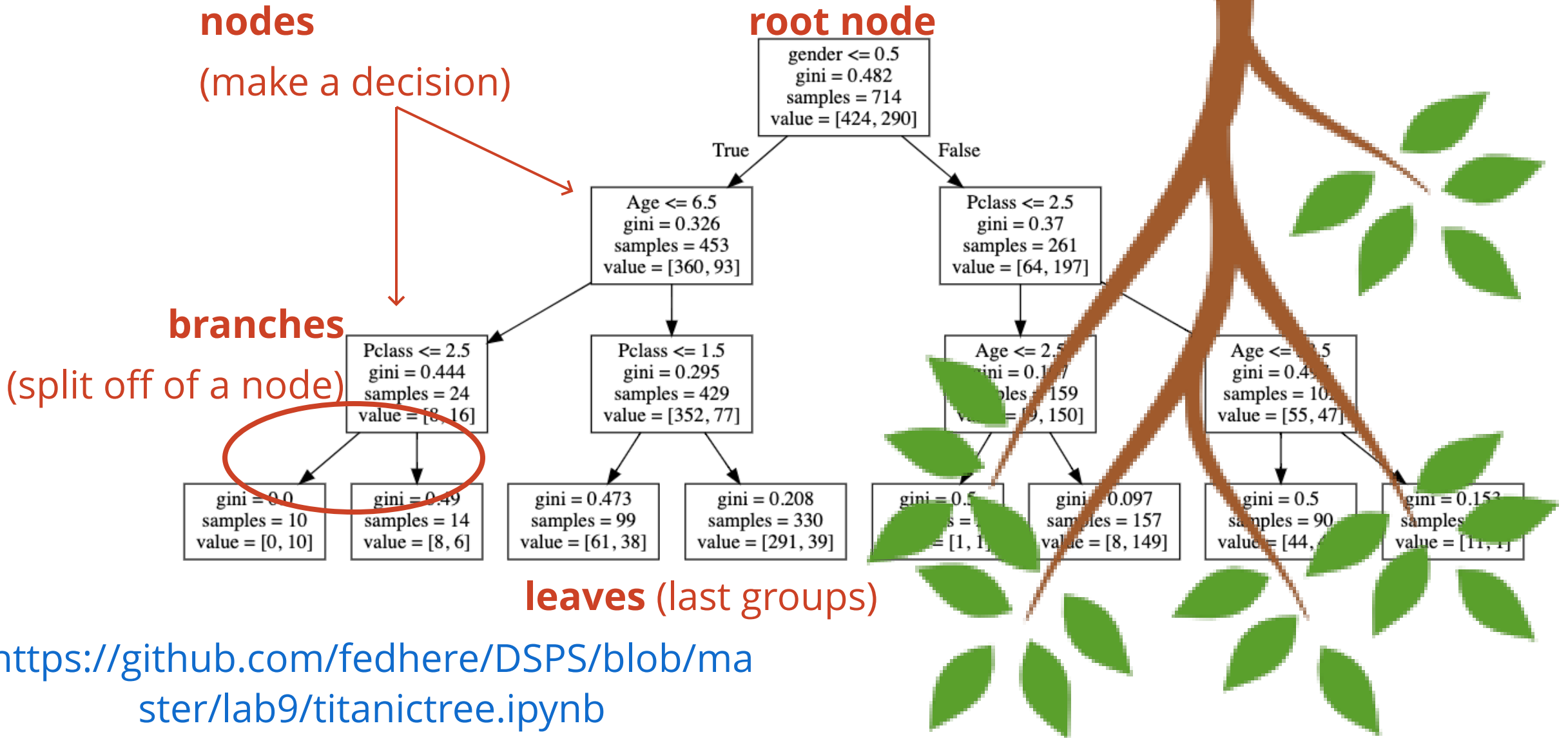
- gender (binary already used)
- ticket class (*ordinal*)
- age (continuous)

target variable:

-> survival (y/n)

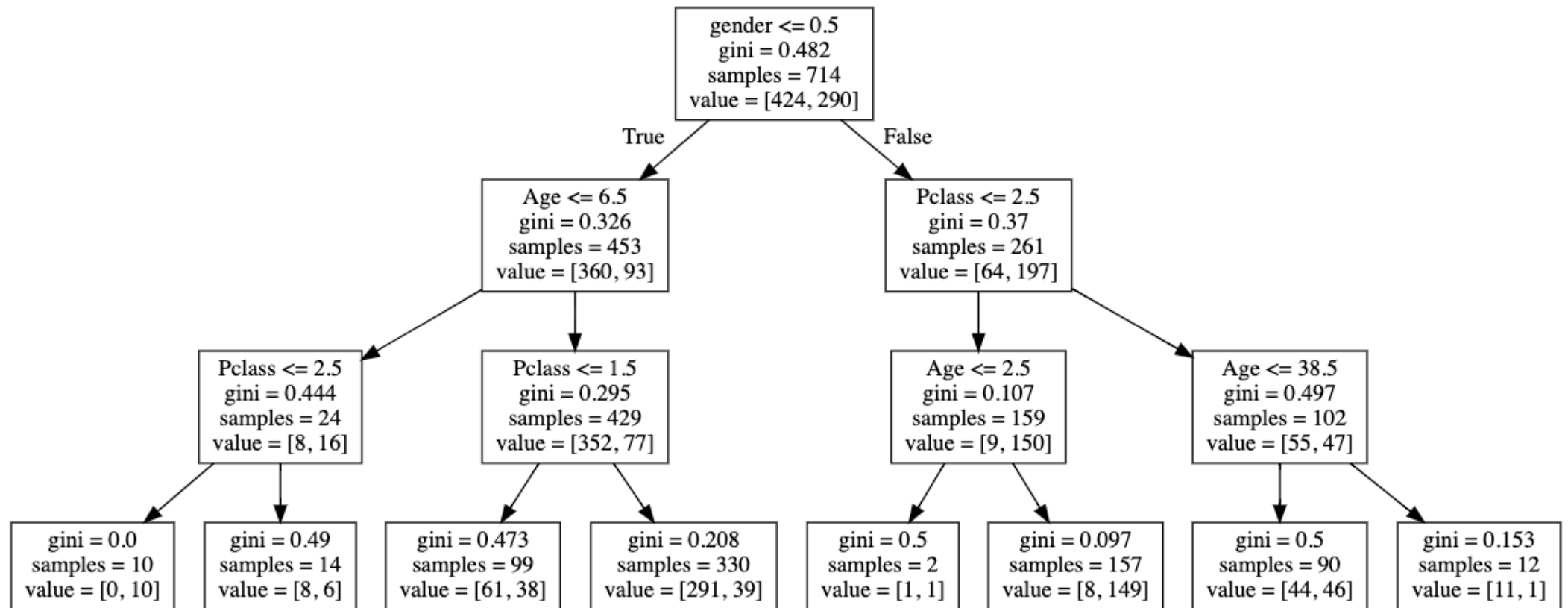


A single tree



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.

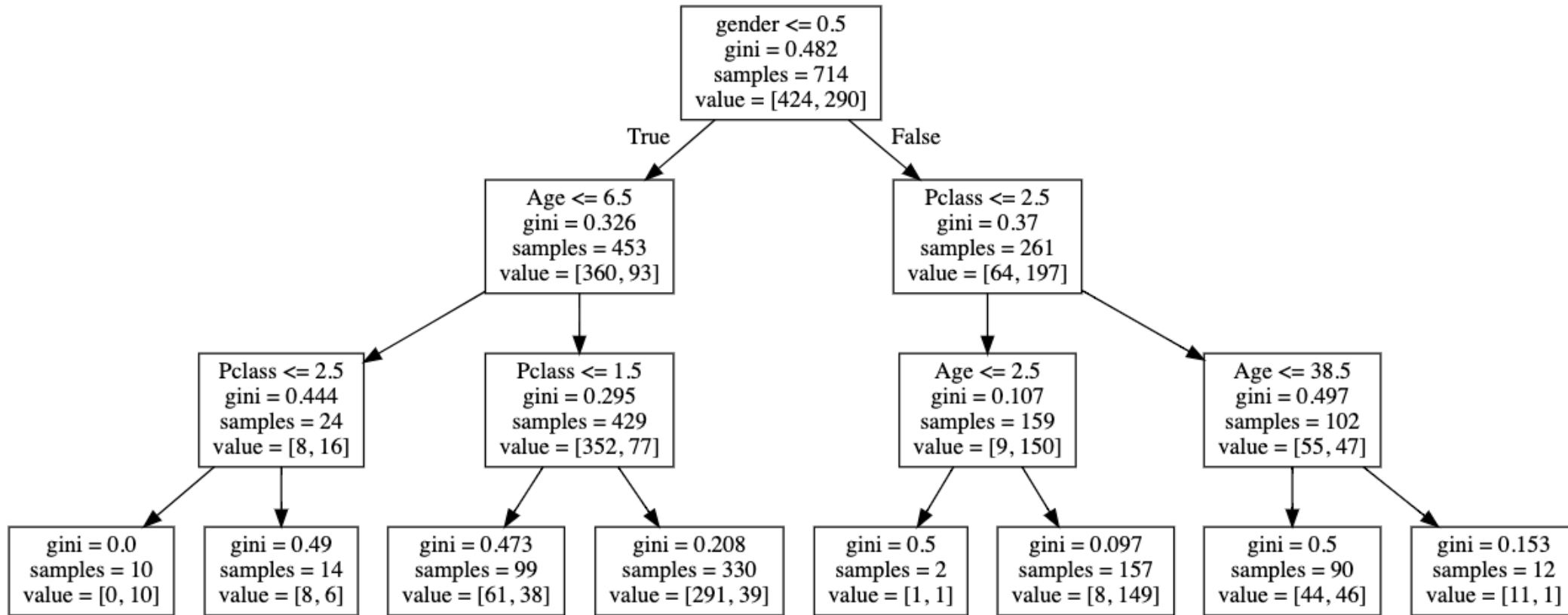
gini impurity

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

information gain (entropy)

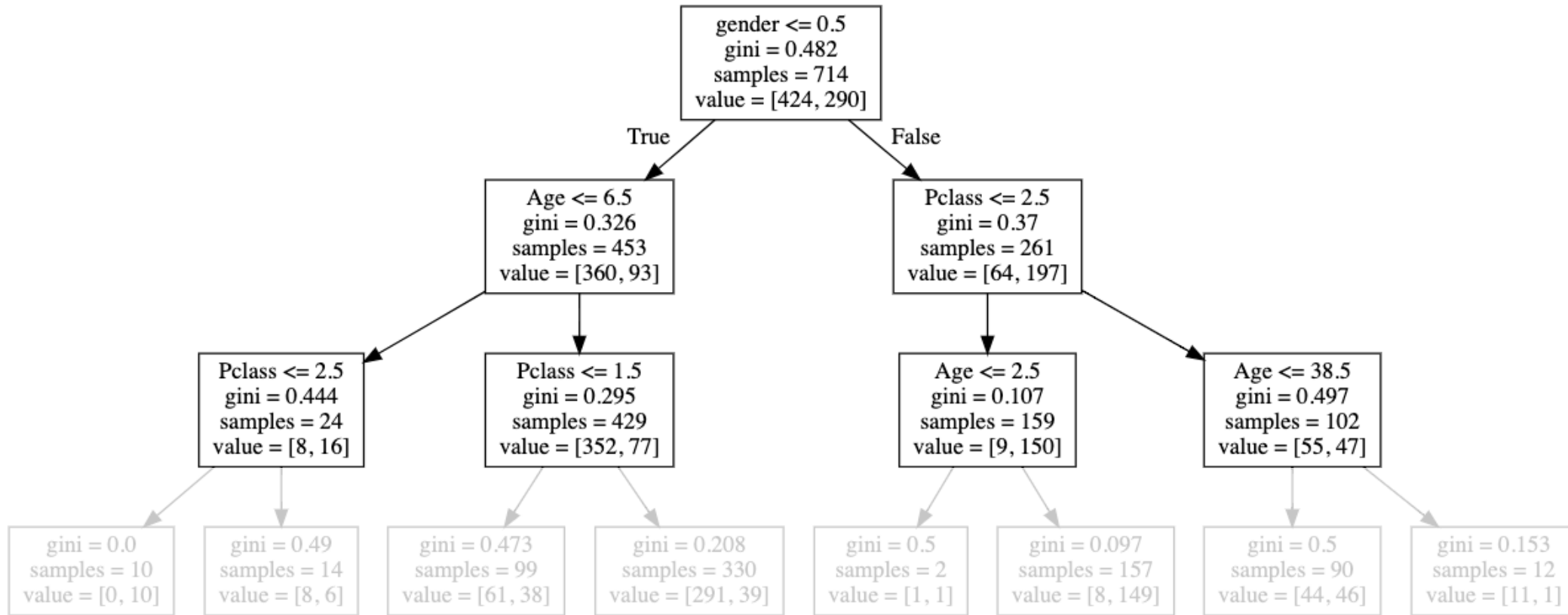
$$H(T) = - \sum_{i=1}^J p_i \log_2 p_i$$

A single tree: hyperparameters



depth

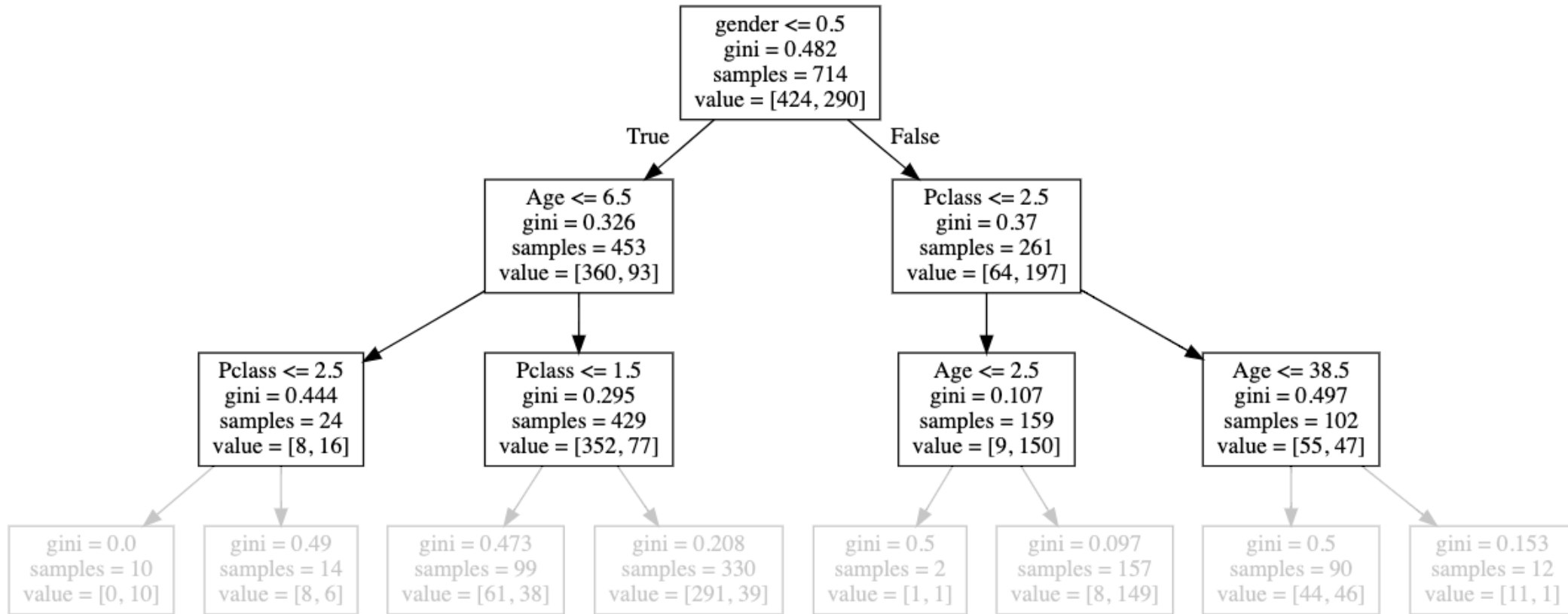
A single tree: hyperparameters



max depth = 2



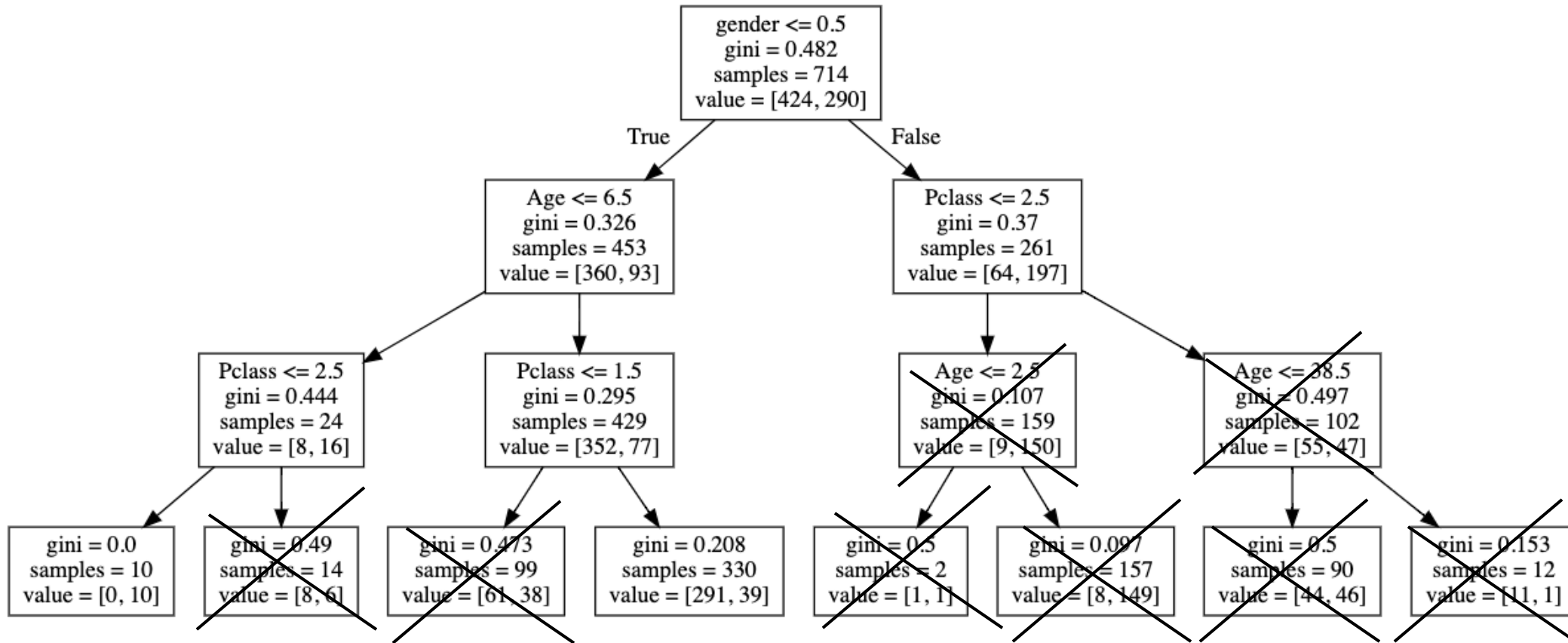
A single tree: hyperparameters



max depth = 2

**PREVENTS
OVERFITTING**

A single tree: hyperparameters

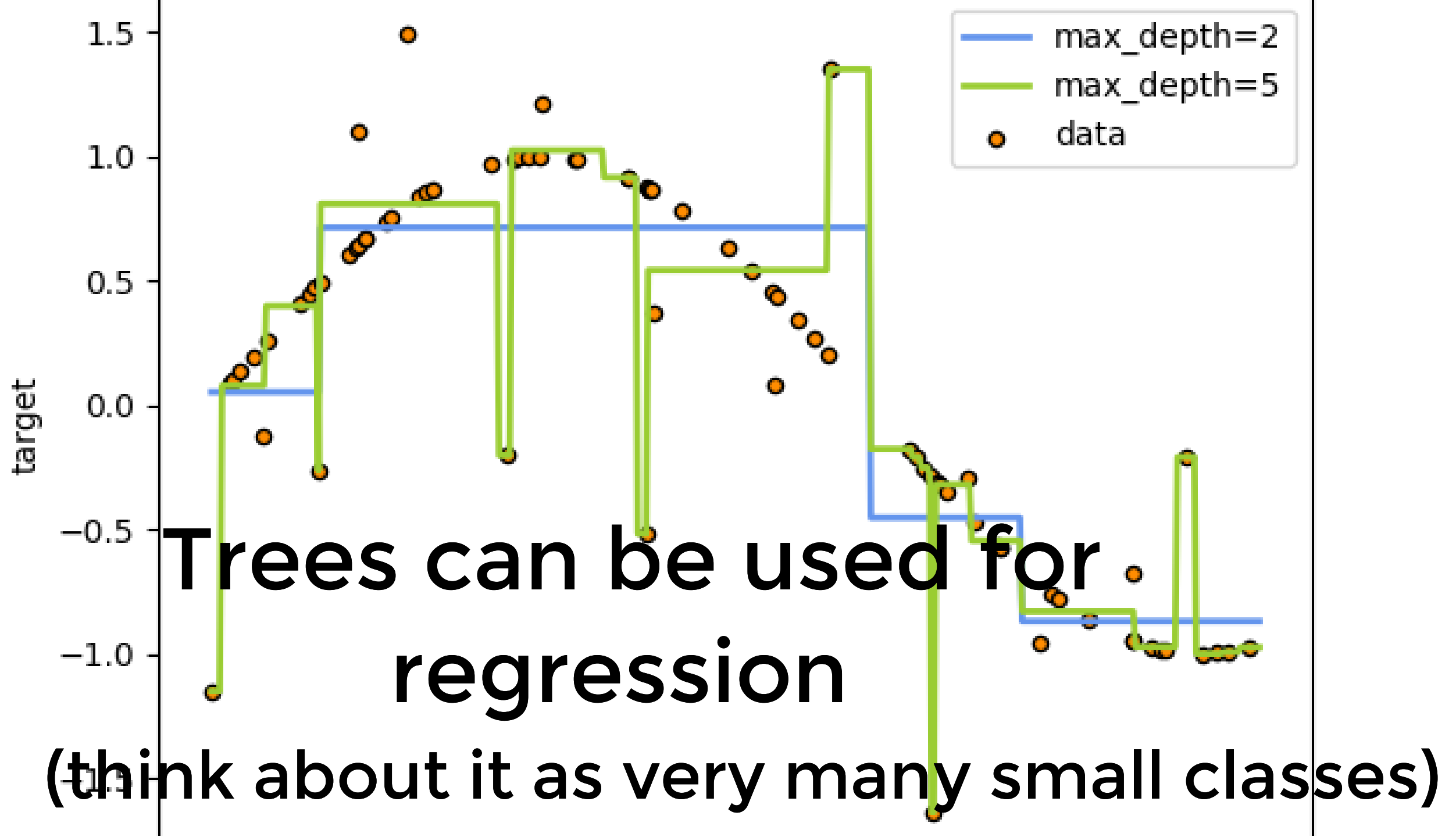


alternative: tree pruning

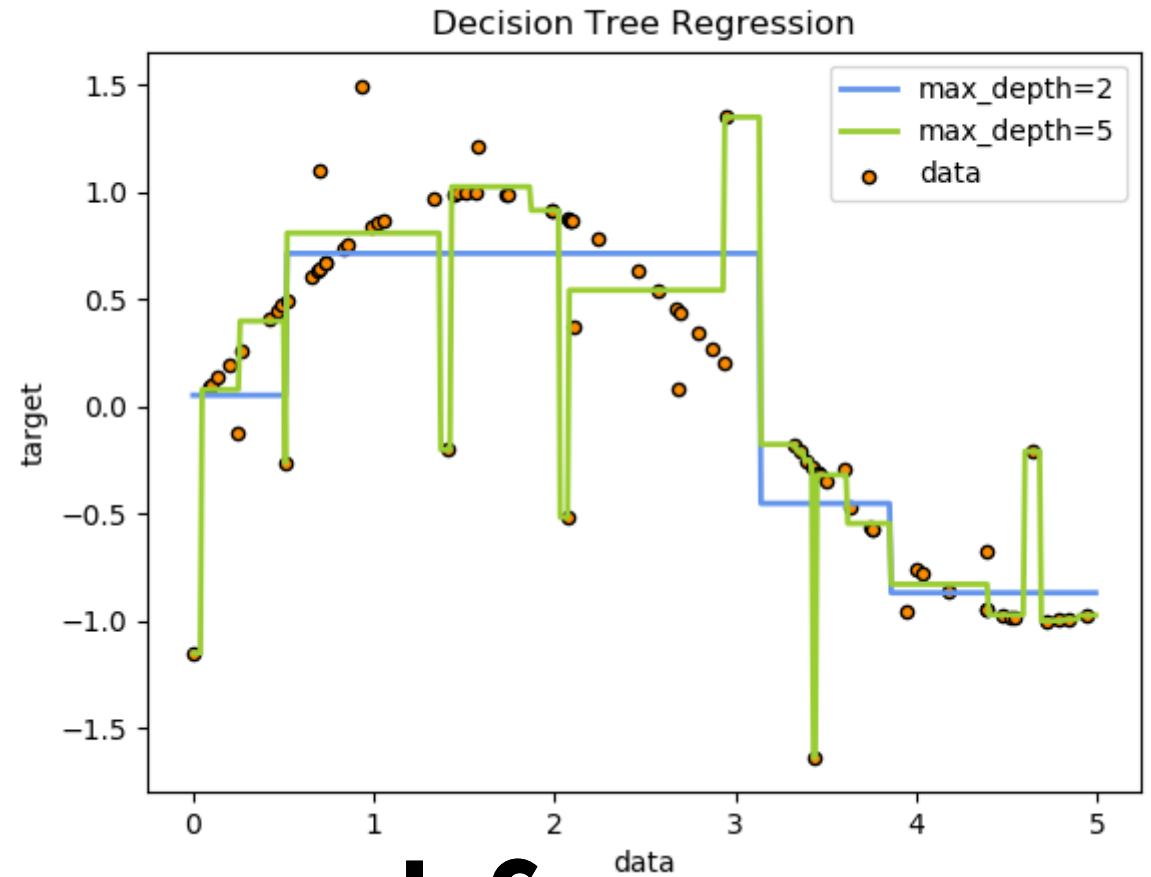
regression with trees

3

CART: Classification and Regression Trees



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) ¶
```

[\[source\]](#)

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

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

mean absolute error

$$L_1 = \sum |y_{true} - y_{predicted}|$$

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
node is an untractable problem!**

e.g. 2 continuous 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 "ensemble" 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 ensemble of methods

tree ensemble methods

Random forest:

trees run in parallel
(independently of each other)

each tree uses a random subset
of observations/features
(bootstrap - 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 weights from the
previous tree

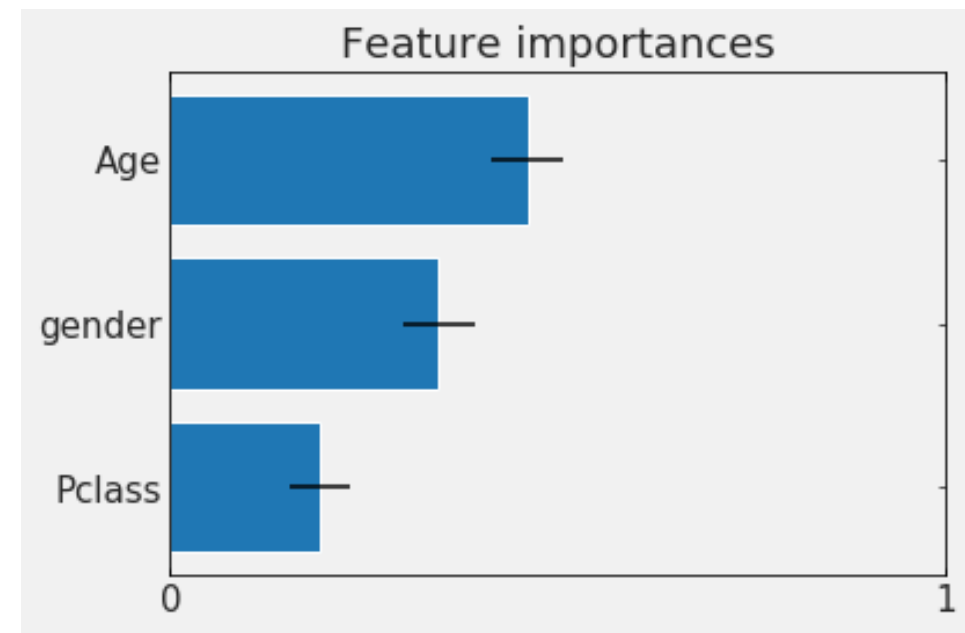
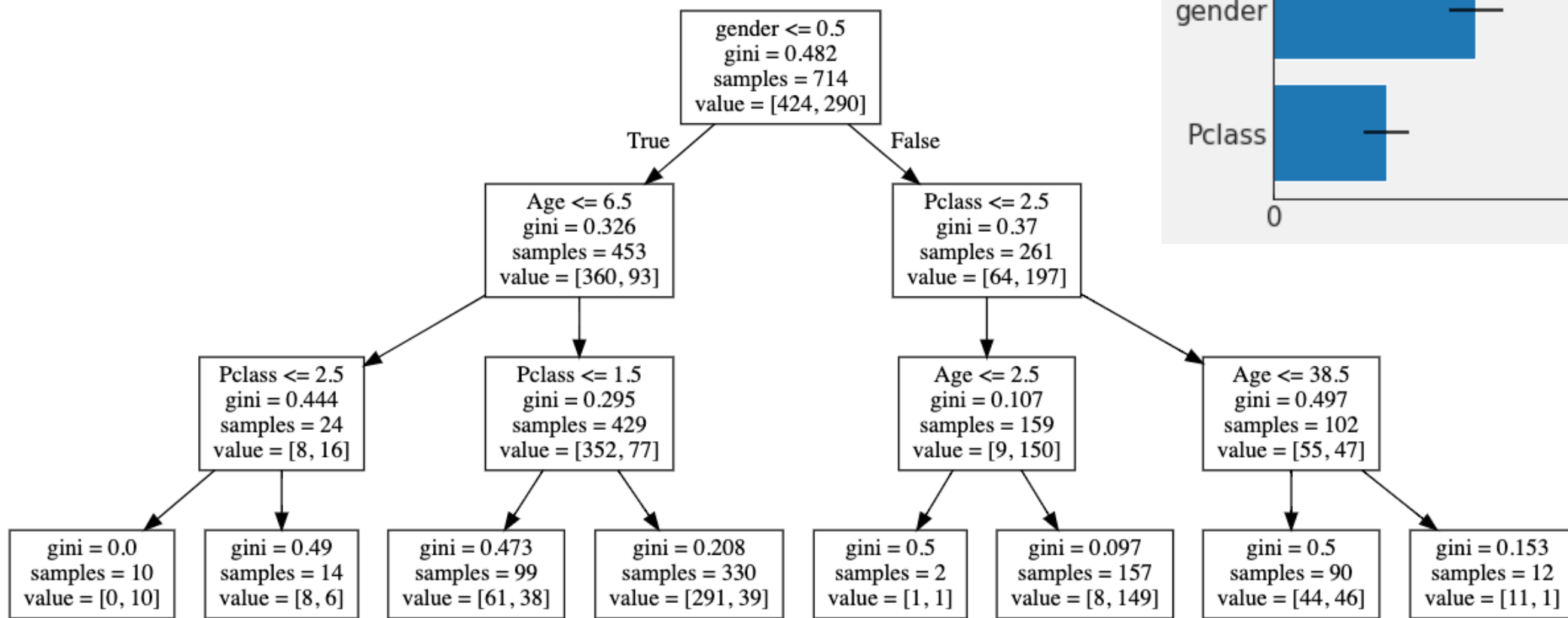
the last tree has the prediction

feature
importance

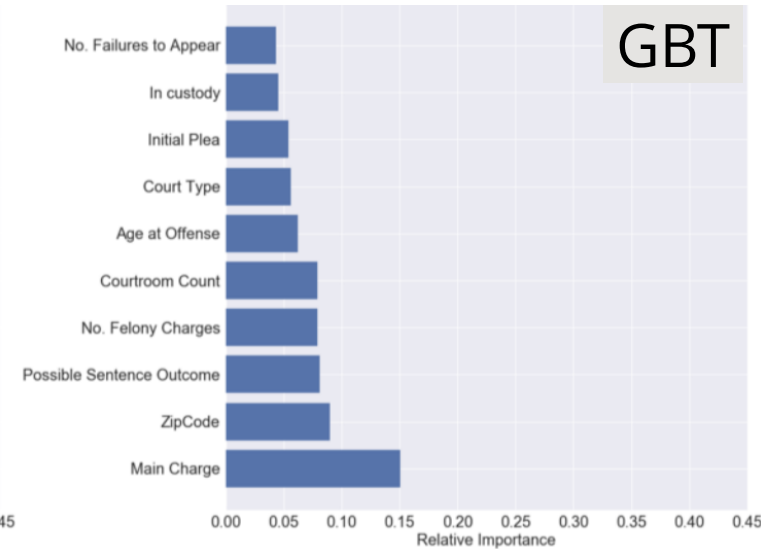
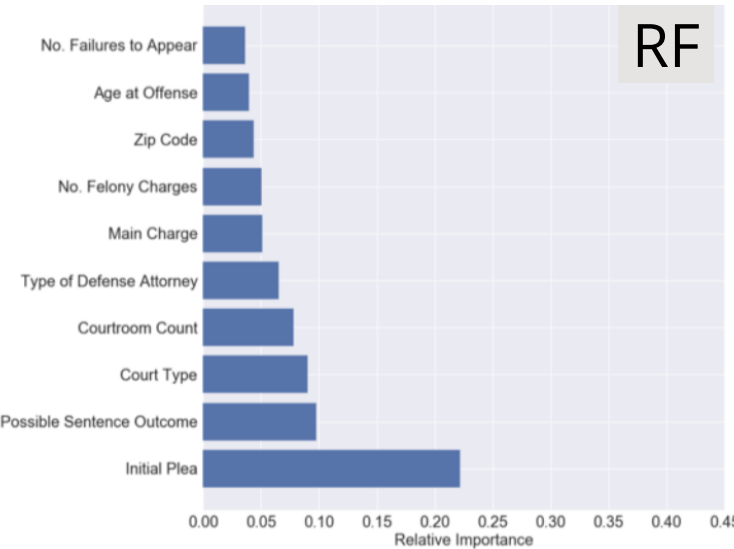
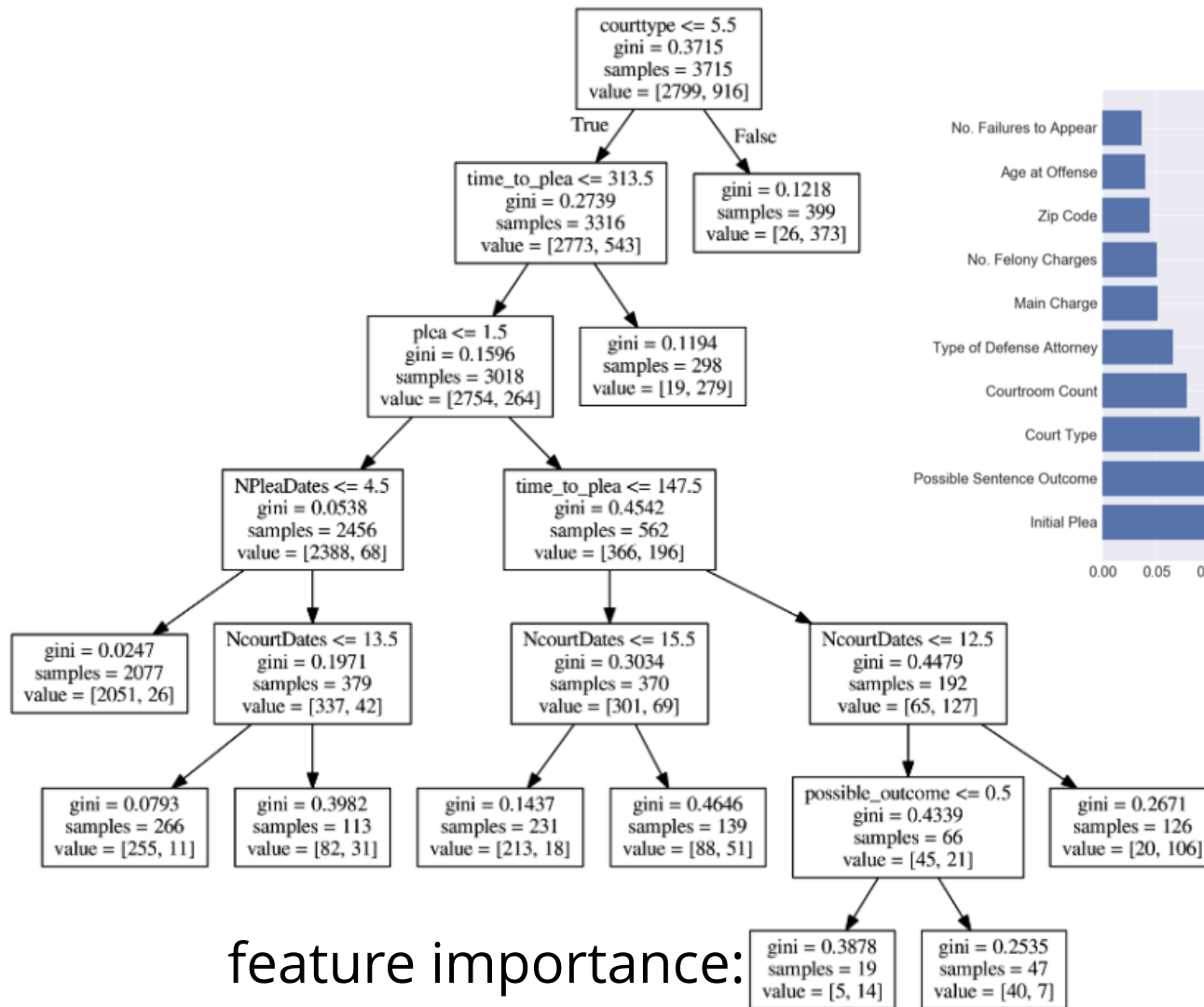
6

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>



feature importance:

how soon was a feature chosen,

how many times was it used...

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

A Data-Driven Evaluation of Delays in Criminal Prosecution

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

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**

7 encoding categorical variables

Categorical Variable
variable that can take a finite
number of values.

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

Categorical Variable

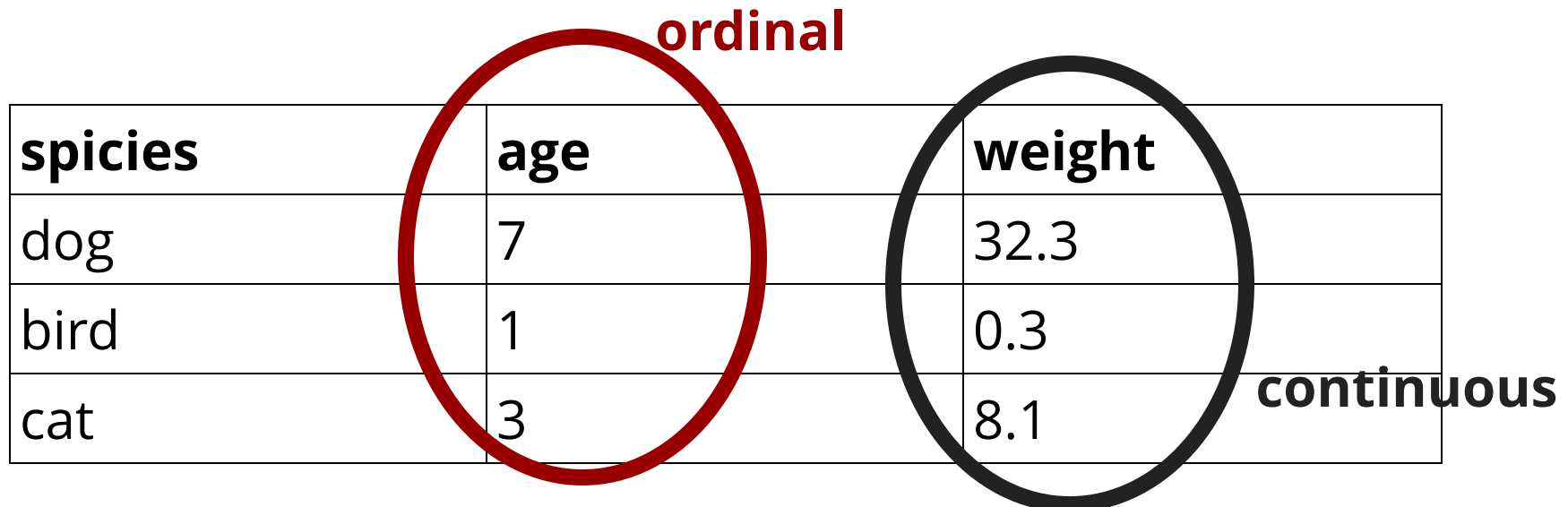
variable that can take a finite number of values.

species	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

species	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.

categorical	species	ordinal	weight	continuous
dog	7	32.3		
bird	1	0.3		
cat	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

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

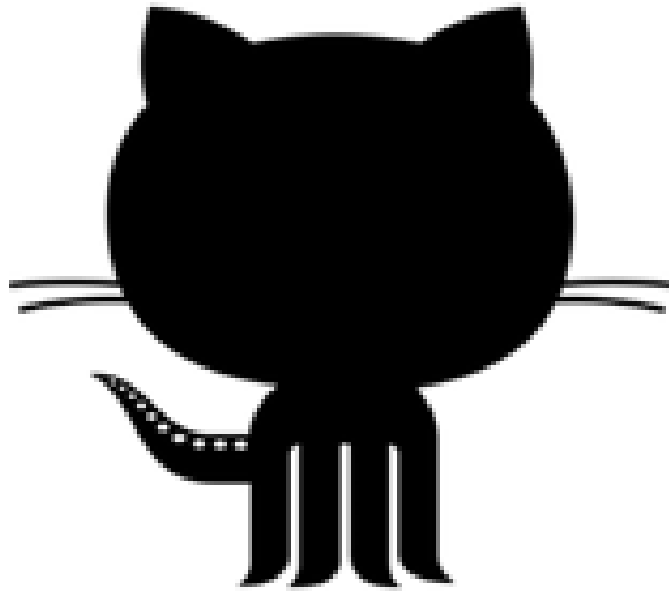
Preferred!

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>

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

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

	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

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

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

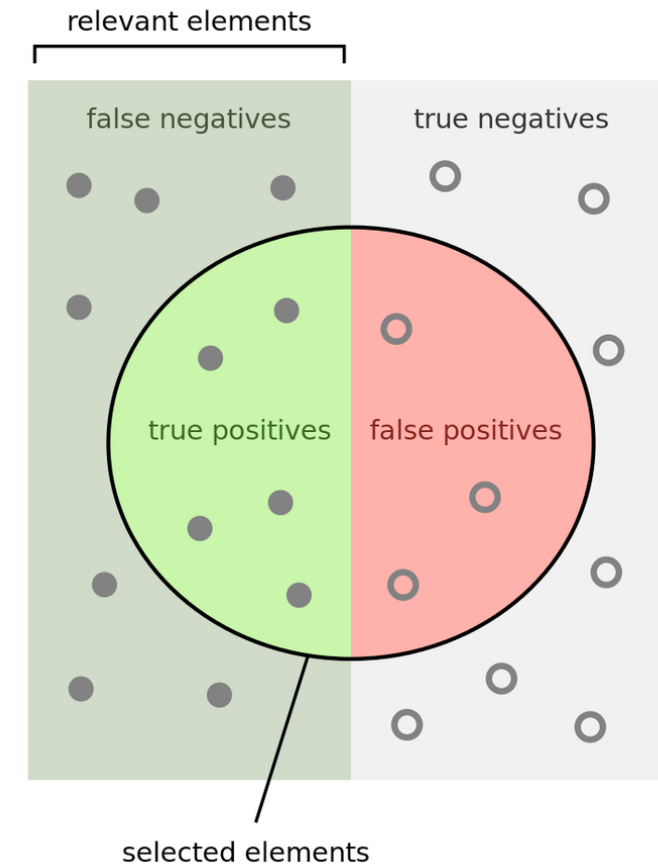
$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

TP=True Positive

FP=False Positive

TN=True Negative

FN=False Negative

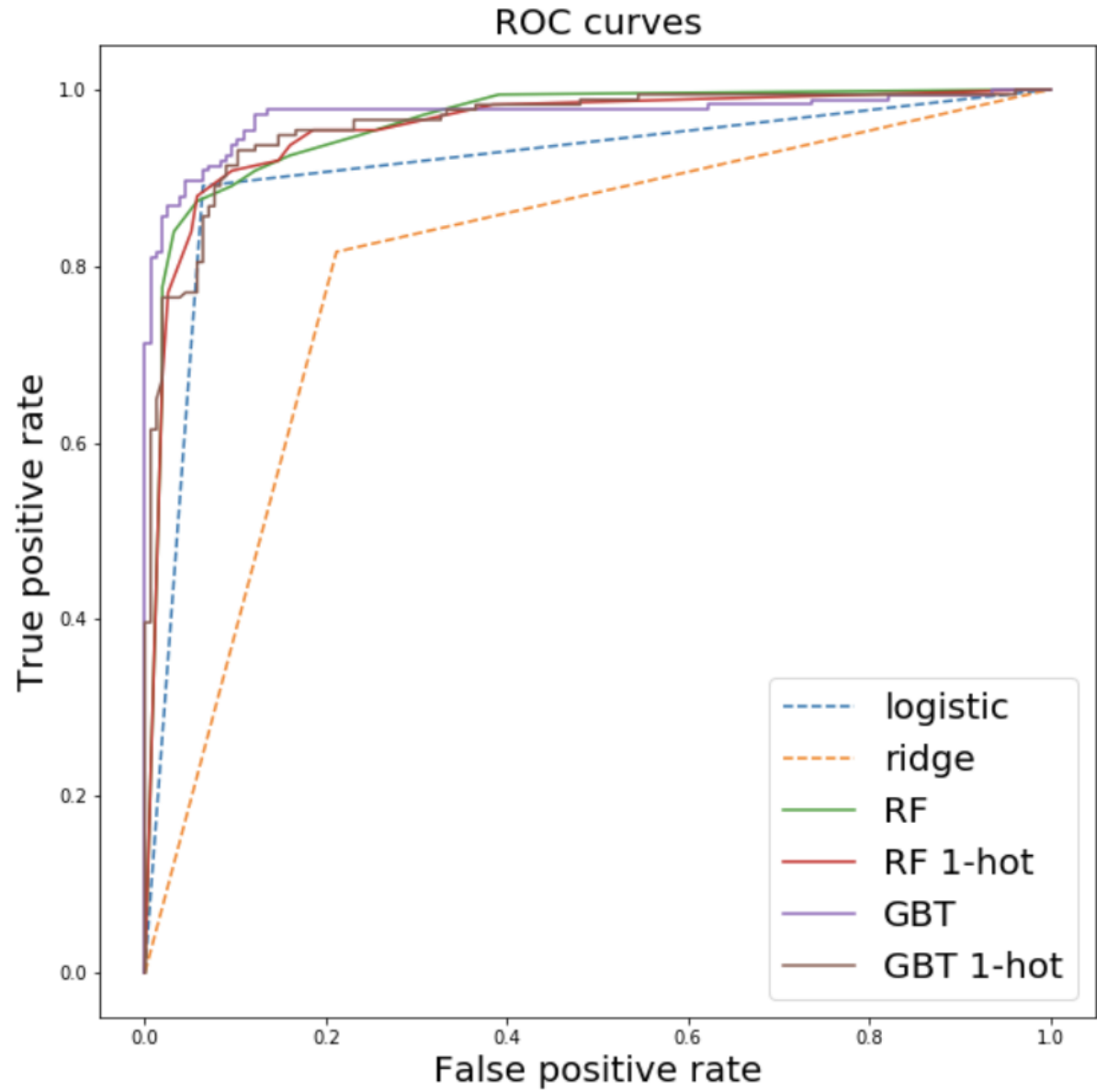


How many selected items are relevant?

$$\textbf{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

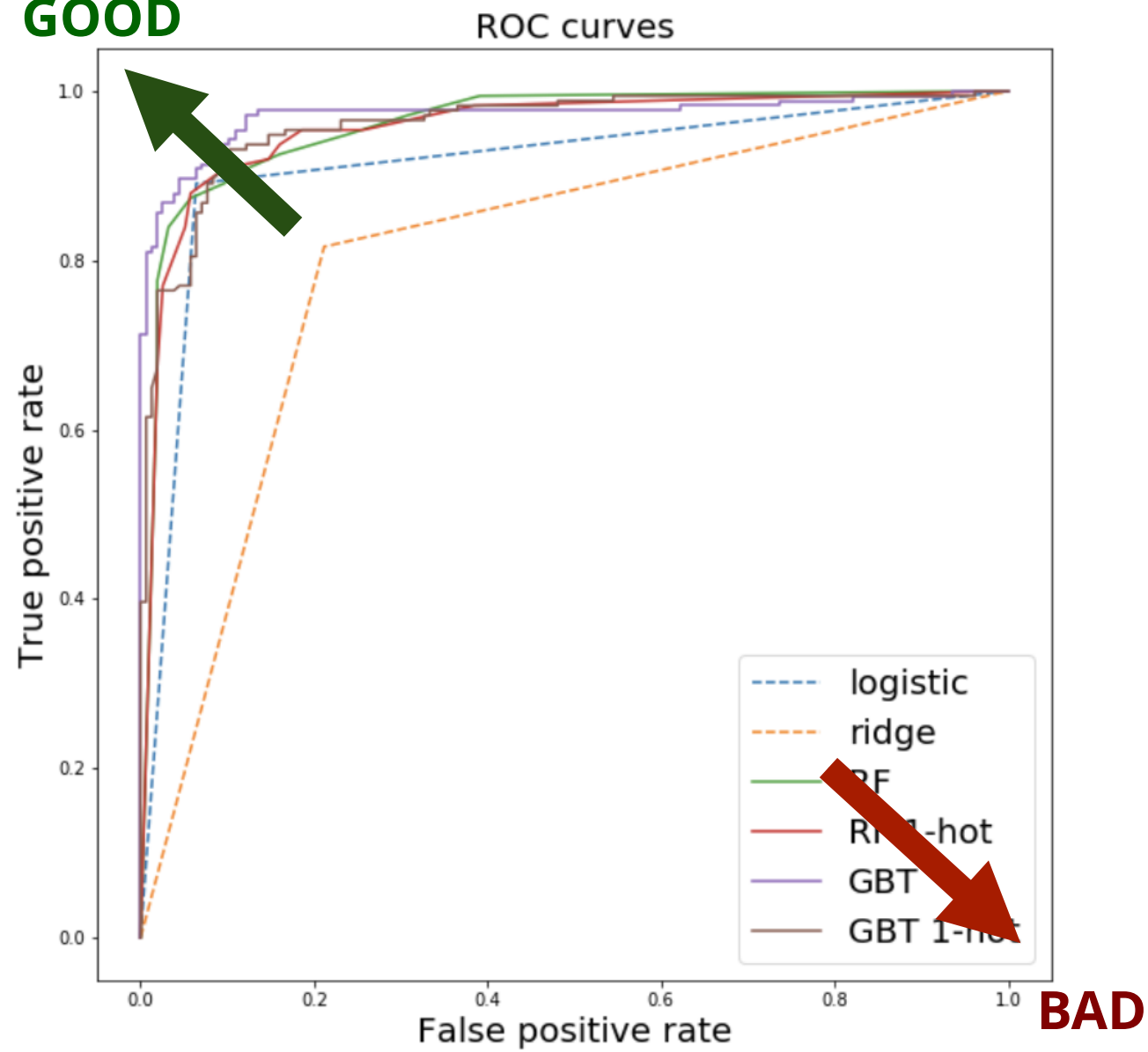
How many relevant items are selected?

$$\textbf{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$



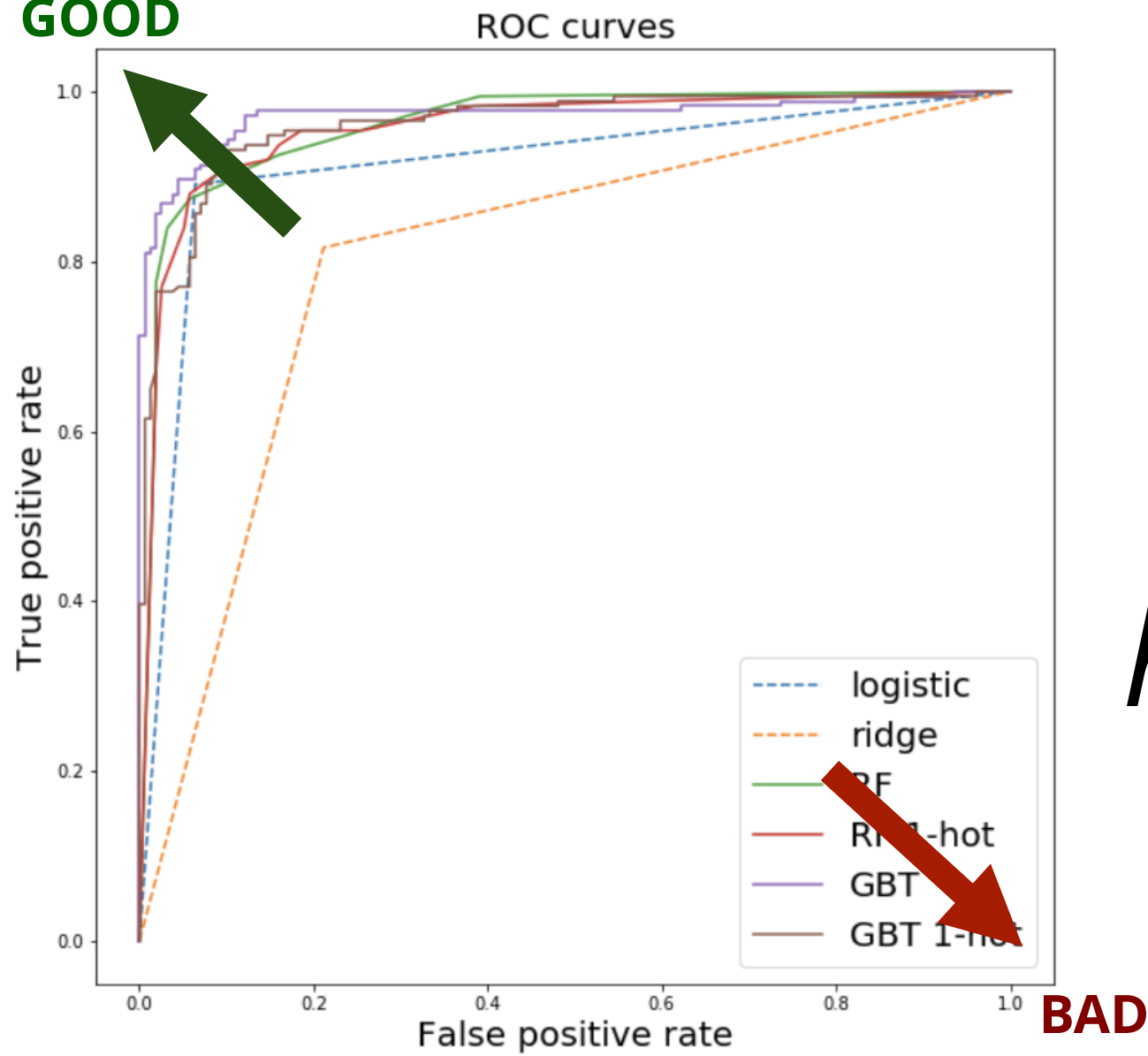
Receiver operating characteristic

GOOD



Receiver operating characteristic

GOOD



tuning by
changing
hyperparameters

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

Keynotes

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 ($TP / (TP + FN)$), and F1 score ($2 * TP / (2 * TP + FP + FN)$).

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

reading

Higgs Boson Search

- Download the Higgs boson data from Kaggle (programmatically within the notebook, see how in the Titanic notebook)
- Split the provided training data into a training and a test set. For each model calculate and discuss the training and test score results.
- Use a Random Forest and a Gradient Boosted Tree *Classifier model* to predict the label of the particles.
- Produce a confusion matrix for each model and compare them
- Use a Random Forest and a Gradient Boosted Tree *Regressor model* to predict the weight of the particles.
- Calculate the L1 and L2 metrics of each model and compare them.
- For the Random Forest classifier, select the 4 most important features (see how in the Titanic notebook) and explore the parameter space with the sklearn module `sklearn.model_selection.RandomizedSearchCV` for a model that uses only those features to predict the labels https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html#sphx-glr-auto-examples-model-selection-plot-randomized-search-py
- Generate an ROC curve plot for the best model and discuss it https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html or https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html#sphx-glr-auto-examples-model-selection-plot-roc-crossval-py
- EC and 667
- Download the script provided in the kaggle challenge to validate your model.
- Generate an output file as required by this script for your best model
- Report on the result

Visualization review

- Work on this alone

You will receive an email with 2 names and github handles of classmates. Review their plots according to the things we discussed last lecture. You can discuss your review with others but each of you should submit a review for each of the plots

There will be detailed instructions in the email on how to review, what structure the review should have, what to focus on, etc. Please comply to the instructions. Upload the review on your github DSPS HW9 repo AND ALSO in your classmate's HW8 repo, forking and submitting a pull request, NOT JUST your fork of their repo! Please note the numbers: I will grade what is in your HW9 repo and check that it is also in your classmate HW8 repo.

Each review will be reviewed and graded by me. (Please take this homework seriously: one sentence generic reviews will be graded 0)

homework