

A photograph of a forest with tall, slender trees. Sunlight is streaming through the canopy from the upper right, creating a warm, golden glow. The ground is covered in brown pine needles and some green moss. A large, thick tree trunk is visible on the right side of the frame.

# Lesson #19 Ensemble Learning I

- Ensembles (introduction)
- Voting classifiers
- Bagging & Pasting
  - Random Forest
  - Feature Importance (XAI)
- Case Study



# Wisdom of the crowd



A NEW YORK TIMES BUSINESS BESTSELLER

"As entertaining and thought-provoking as *The Tipping Point* by Malcolm Gladwell. . . . *The Wisdom of Crowds* ranges far and wide."

—*The Boston Globe*

# THE WISDOM OF CROWDS

JAMES  
SUROWIECKI

WITH A NEW AFTERWORD BY THE AUTHOR

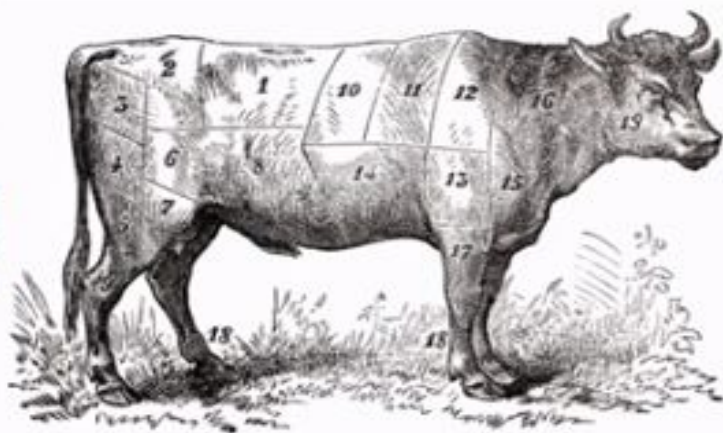


Paradoxically, the way for a group to be smart is for each person in it to think and act as independently as possible.

# The Wisdom of Crowds

A NEW YORK TIMES BESTSELLING BOOK  
"An extraordinary and thought-provoking work. The Wisdom of Crowds is a masterpiece." —The New York Times

## THE WISDOM OF CROWDS JAMES SUROWIECKI



average of 800 guesses = 1,197  
actual weight of the ox = 1,198



Logistic  
Regression



SVM  
Classifier



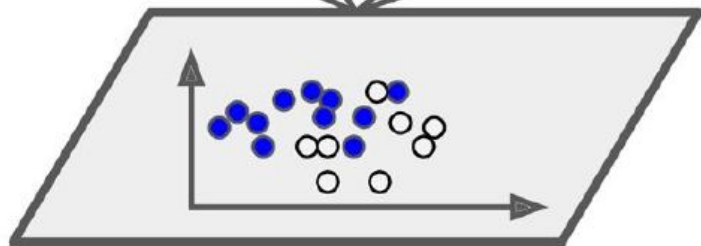
Random  
Forest Classifier



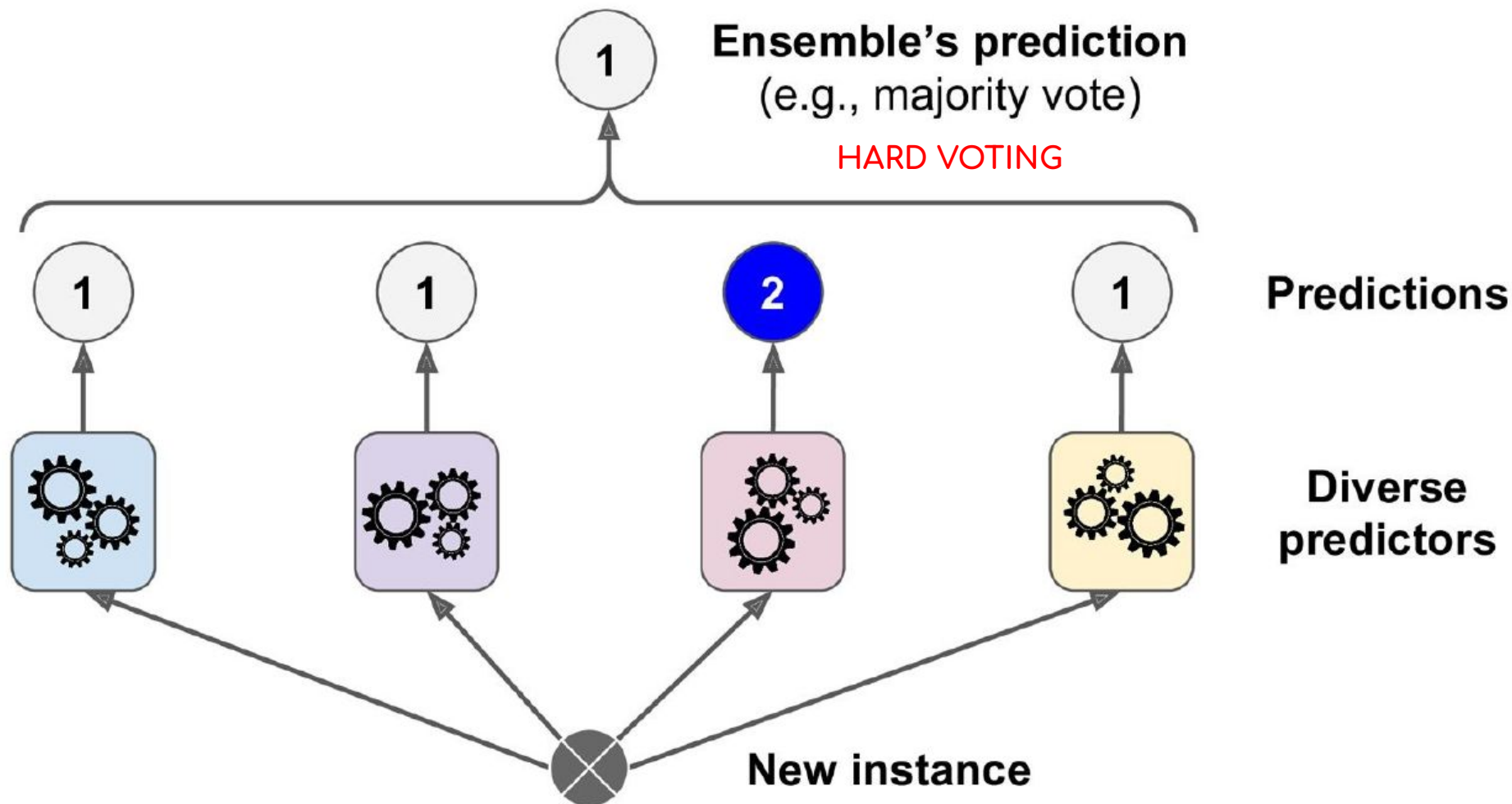
Other...



**Diverse  
predictors**





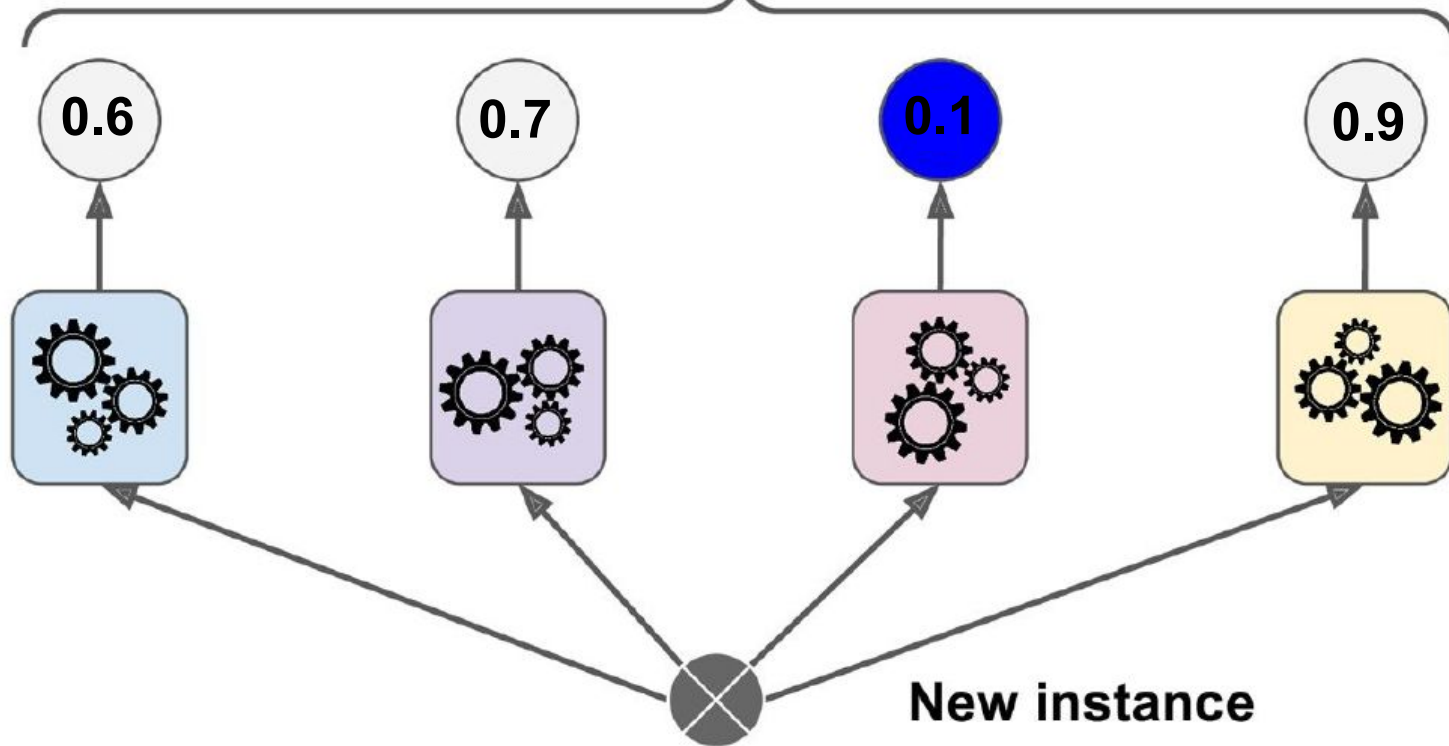


```
>> mean([0.6,0.7,0.1,0.9])  
>> 0.575
```

1

**Ensemble's prediction**  
(e.g., majority vote)

SOFT VOTING



**Predictions**

**Diverse  
predictors**

**New instance**



# Netflix Prize

**COMPLETED**[Home](#)[Rules](#)[Leaderboard](#)[Update](#)[Browse](#)[Recommendations](#)[Friends](#)[Queue](#)[Buy DVDs](#)[Home](#)[Genres](#)[New Releases](#)[Previews](#)[Netflix Top 100](#)[C](#)

## Movies For You

Randy, the following movies were chosen based on your interest in:  
[Working for Suck](#)  
[The Mentalist: Season 1](#)  
[Greenbelt](#)

### The Big One

★★★★★

More of our favorite

subversive

from

by

on /

Original

OT

Less

and

Season 2.0

Disc Series

★★★★★

Daniel Knau

riveting

series

document

of a motley

series who've

made the

show their

Read More

All Discs  
Guaranteed

You really  
liked it..

Now own it for just \$5.98

Shop

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

as low

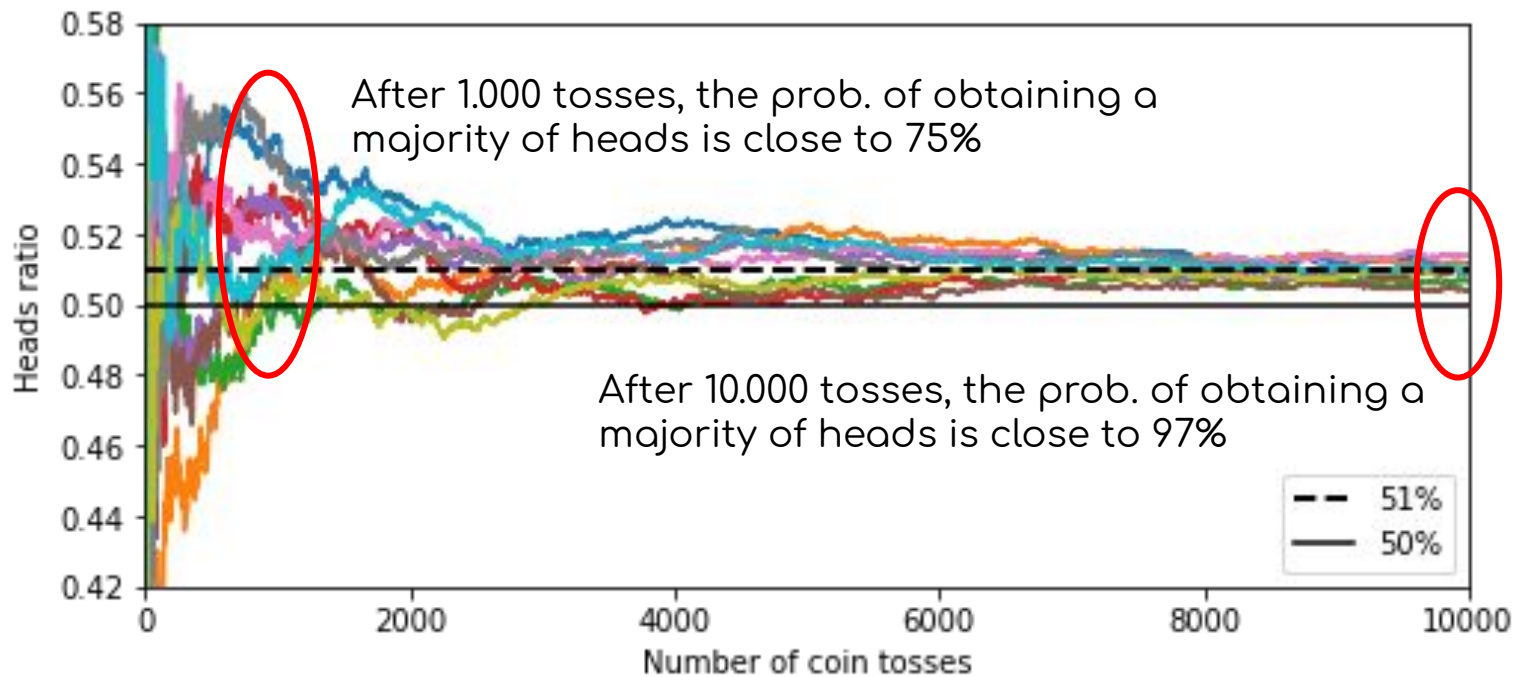
## Congratulations!

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the \$1M Grand Prize to team "BellKor's Pragmatic Chaos". Read about [their algorithm](#), checkout team scores on the [Leaderboard](#), and join the discussions on the [Forum](#).

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.

How is this  
possible?



```
# suppose you have a slightly biased coin
```

```
heads_proba = 0.51
```

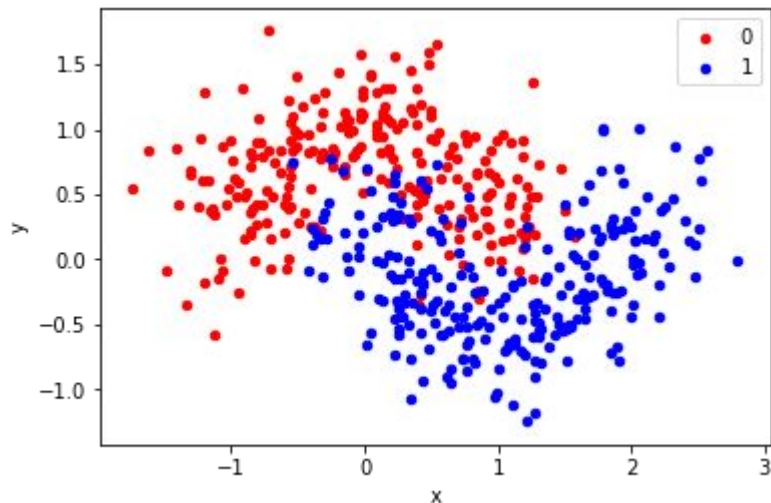
```
# 10 series of biased coin tosses (10.000)
```

```
coin_tosses = (np.random.rand(10000, 10) < heads_proba).astype(np.int32)
```

```
# cumulative heads ratio for each series
```

```
cumulative_heads_ratio = np.cumsum(coin_tosses, axis=0) / np.arange(1, 10001).reshape(-1, 1)
```





```
# create classifiers
```

```
log_clf = LogisticRegression(solver="lbfgs", random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=100, random_state=42)
svm_clf = SVC(gamma="scale", random_state=42)
```

```
# soft voting
```

```
voting_clf_hard = VotingClassifier(
    estimators=[('lr', log_clf),
                 ('rf', rnd_clf),
                 ('svc', svm_clf)],
    voting='soft')
```

LogisticRegression 0.85

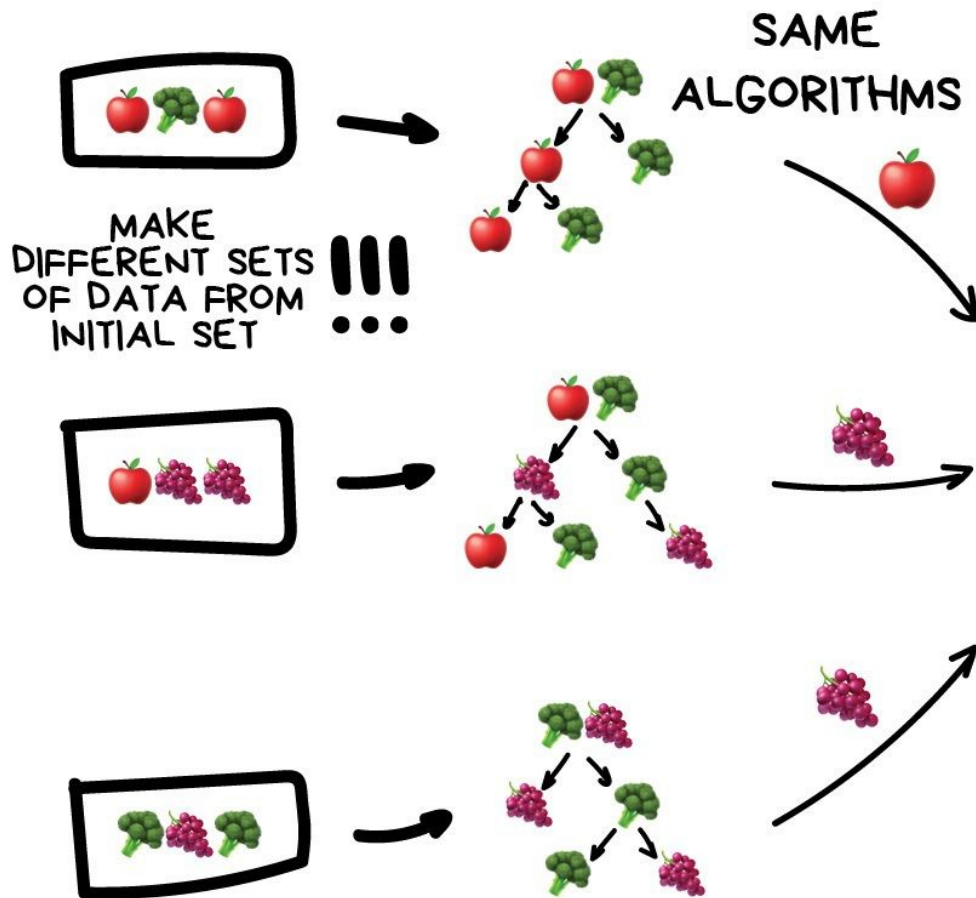
RandomForestClassifier 0.88

SVC 0.87

VotingClassifier 0.89

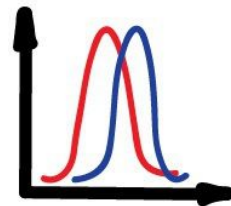
# ENSEMBLE METHODS

BAGGING & PASTING  
BOOSTING  
STACKING



BAGGING ON TREES  
//  
RANDOM FOREST

JUST AVERAGING  
ALL THE RESULTS



ANSWER

BAGGING



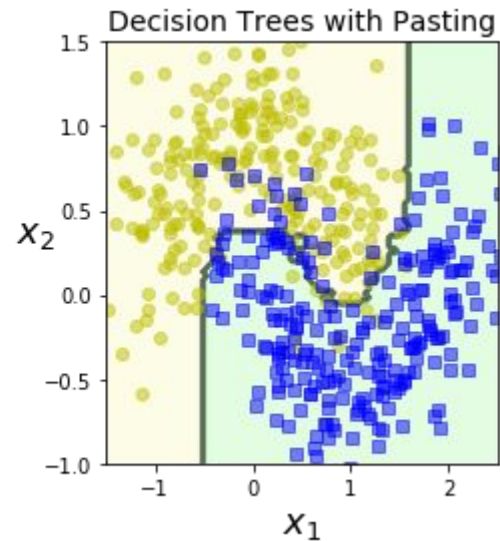
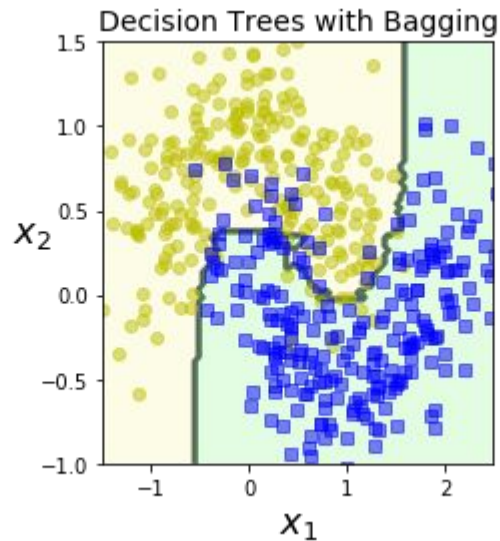
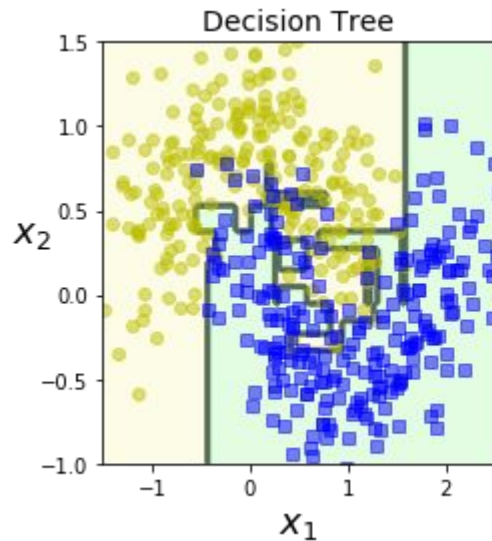
```
# instantiate a bagging object (model,size,samples,bootstrap,rnd)
bag_clf = BaggingClassifier(DecisionTreeClassifier(random_state=42),
                           n_estimators=500,
                           max_samples=100,
                           #bootstrap=False -> pasting
                           bootstrap=True,
                           random_state=42)

# training the bagging model
bag_clf.fit(X_train, y_train)
```



**max\_samples**: int or float, optional (default=1.0). The number of samples to draw from X to train each base estimator.

- If int, then draw **max\_samples** samples.
- If float, then draw **max\_samples \* X.shape[0]** samples.



Accuracy.bagging: 0.90

Accuracy.pasting: 0.89

Accuracy.decision\_tree: 0.82

# Out-of-Bag Evaluation

```
bag_clf = BaggingClassifier(DecisionTreeClassifier(random_state=42),  
                             n_estimators=500,  
                             bootstrap=True,  
                             oob_score=True,  
                             random_state=40)  
  
bag_clf.fit(X_train, y_train)  
bag_clf.oob_score_
```

0.9075

$$\lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m = e^{-1} = 0.3678$$





# Random decision forests

**Publisher:** IEEE

**1 Author(s)** Tin Kam Ho [View All Authors](#)

**215**  
Paper  
Citations

**5**  
Patent  
Citations

**3099**  
Full  
Text Views



## Abstract

## Authors

## References

## Citations

## Keywords

## Metrics

### Abstract:

Decision trees are attractive classifiers due to their high execution speed. But trees derived with traditional methods often cannot be grown to arbitrary complexity for possible loss of generalization accuracy on unseen data. The limitation on complexity usually means suboptimal accuracy on training data. Following the principles of stochastic modeling, we propose a method to construct tree-based classifiers whose capacity can be arbitrarily expanded for increases in accuracy for both training and unseen data. The essence of the method is to build multiple trees in randomly selected subspaces of the feature space. Trees in, different subspaces generalize their classification in complementary ways, and their combined classification can be monotonically improved. The validity of the method is demonstrated through experiments on the recognition of handwritten digits.

**Published in:** [Proceedings of 3rd International Conference on Document Analysis and Recognition](#)

**Date of Conference:** 14-16 Aug. 1995

**INSPEC Accession Number:** 5628989

**Date Added to IEEE Xplore:** 06 August 2002

**DOI:** [10.1109/ICDAR.1995.598994](#)

**Print ISBN:** 0-8186-7128-9

**Publisher:** IEEE

```
# Random Forest using BaggingClassifier
```

```
bag_clf = BaggingClassifier(DecisionTreeClassifier(max_features="auto",  
                                                    max_leaf_nodes=16,  
                                                    random_state=42),  
                            n_estimators=500,  
                            max_samples=1.0,  
                            bootstrap=True,  
                            random_state=42)
```

```
# Random Forest using RandomForestClassifier
```

```
rnd_clf = RandomForestClassifier(n_estimators=500,  
                                max_leaf_nodes=16,  
                                random_state=42)
```

- Random Forest introduces extra randomness when growing trees
- Instead of searching for the very best feature, it is possible to search among a random subset of features
- This results in a greater tree diversity (trades a higher bias for a low variance)

```
# Random Forest using BaggingClassifier
```

```
bag_clf = BaggingClassifier(DecisionTreeClassifier(max_features="auto",  
#added this hyperparameter  
splitter="random",  
max_leaf_nodes=16,  
random_state=42),  
  
n_estimators=500,  
max_samples=1.0,  
bootstrap=True,  
random_state=42)
```





## *Feature selection*

- Univariate Selection
- Recursive Feature Elimination
- **Feature Importance**
- Principal Component Analysis (PCA)

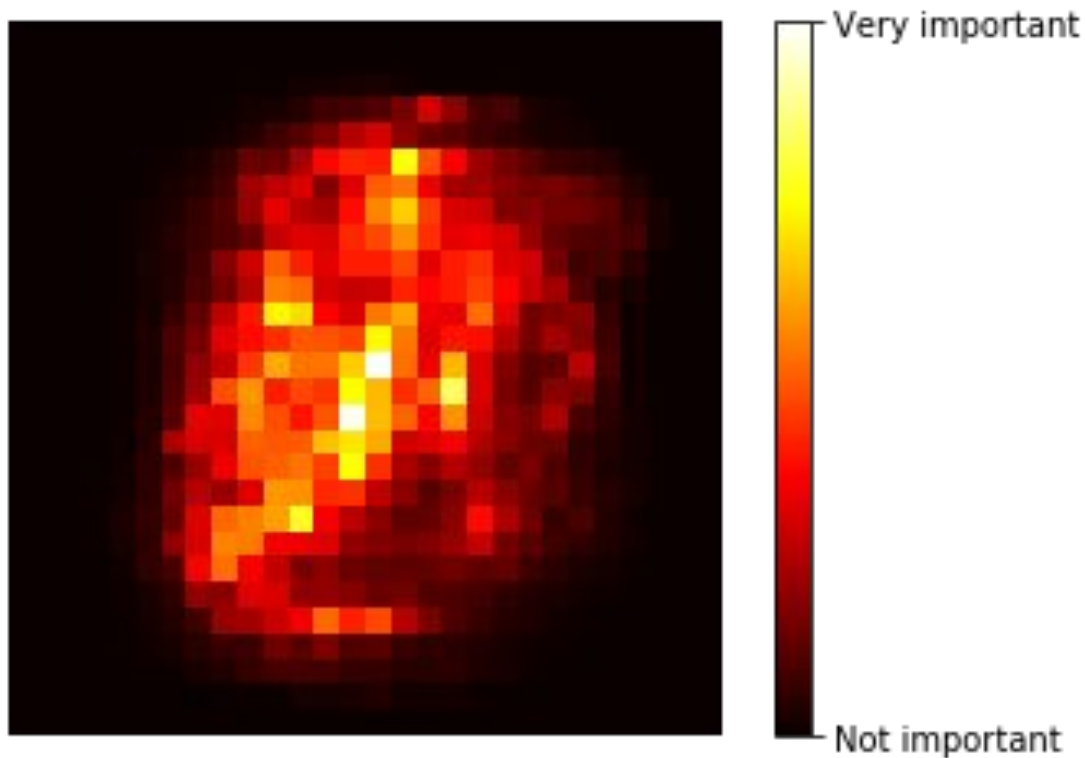
Feature Importance >> Explainable Artificial  
Intelligence (XAI)

# Feature Importance

```
from sklearn.datasets import load_iris
# load dataset
iris = load_iris()
# instantiate the model
rnd_clf = RandomForestClassifier(n_estimators=500, random_state=42)
# training
rnd_clf.fit(iris["data"], iris["target"])
# find the feature importance
for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
    print(name, score)
```

```
sepal length (cm) 0.11249225099876375
sepal width (cm) 0.02311928828251033
petal length (cm) 0.4410304643639577
petal width (cm) 0.4233579963547682
```

# Feature Importance





# When to use Random Forest

---

- Strengths of a Random Forest
  - Very accurate predictions
  - Resistance to overfitting
- Weakness
  - They are difficult to interpret
  - They take longer to create

## Section 2.3.3

- Tuning RandomForestClassifier
- GridSearchCV (progress bar?)
- StratifiedKFold?
- BaggingClassifier
  - splitter="random"?
  - Impact on overfitting?

