Ensemble methods: Bagging

Glenn Bruns
CSUMB

Many figures in this deck from Géron, Hands-on Machine Learning with Scikit-Learn and TensorFlow

Learning outcomes

After this lecture you should be able to:

1. Explain the concept of ensemble methods
2. Define the following concepts, and use them in Scikit-Learn:
   - bagging
   - pasting
   - out-of-bag evaluation
   - random subspaces
   - random patches

Ensemble methods

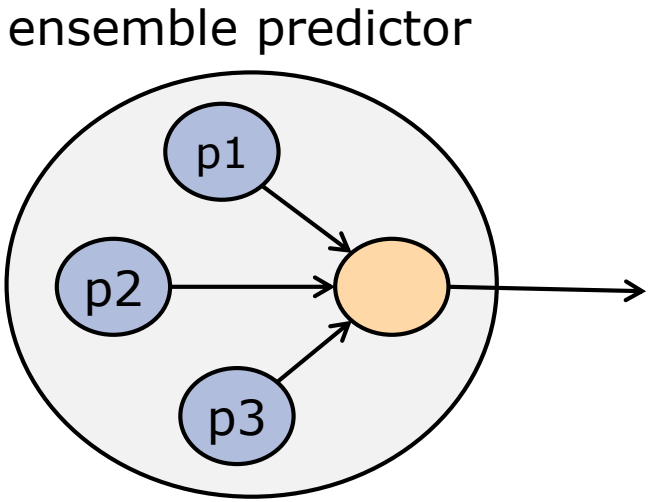
Why do you ask multiple doctors for a diagnosis?

Why do you ask the audience for their answer to a question?

Ensemble methods:

create a "predictor" by combining the answers of multiple predictors

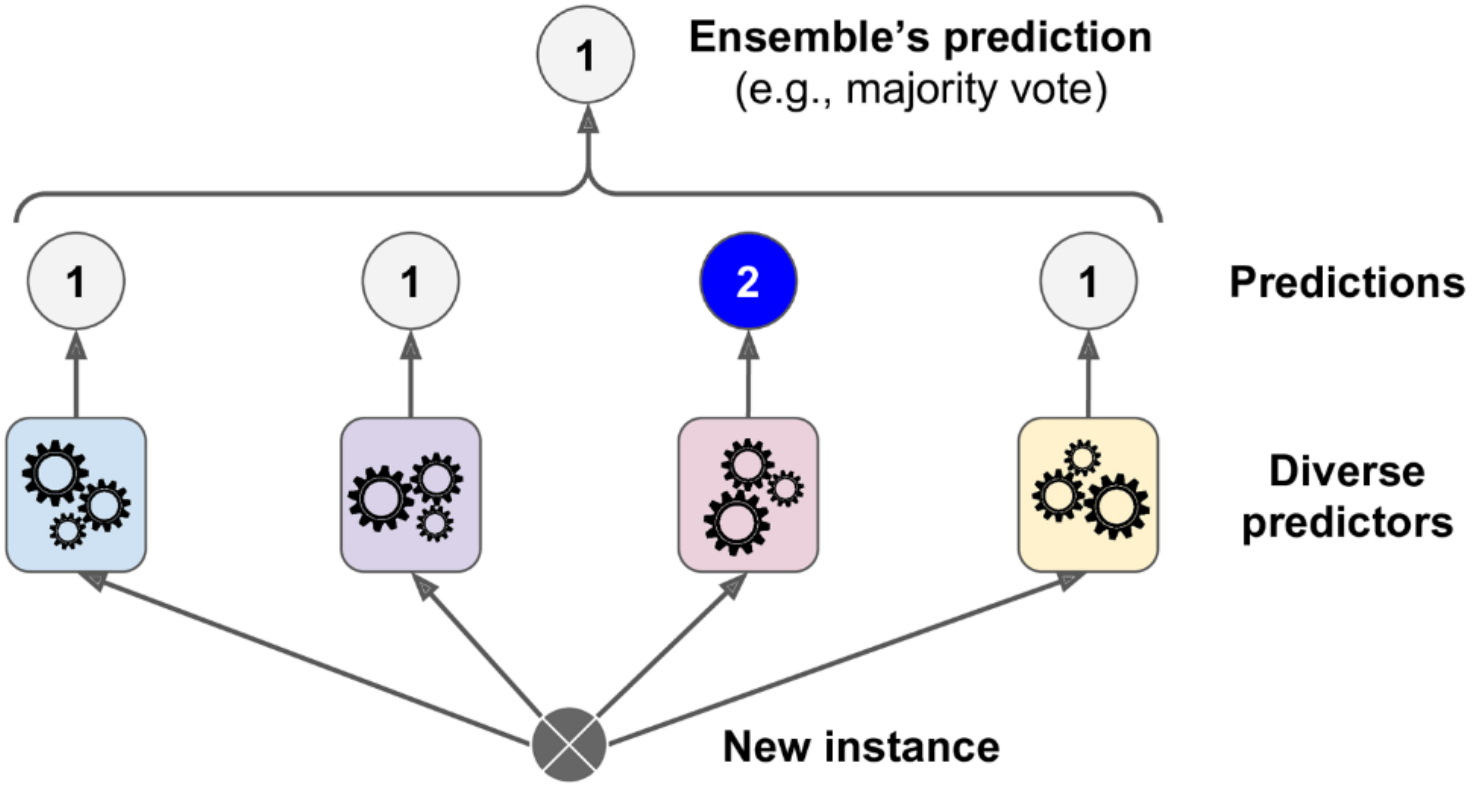
Ensemble methods are effective in practice – used by Netflix prize winners



Questions:

- are all predictors the same kind?
- are they all trained in the same way?
- how are their answers combined?

A simple voting classifier



The diverse predictors might be logistic regression, SVM classification, Naïve Bayes, etc.

All are trained on same training data set.

Why voting works

Conceptually, why should we expect an improvement by voting?

Question: What is the probability of getting a majority of heads in 1,000 flips, if the coin is weighted 0.51 to heads?

Answer: about 0.75

Important: the coin flips are independent
Question: are predictors trained on the same data set likely to be independent?

Voting in Scikit-Learn

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
voting_clf.fit(X_train, y_train)
```

This is called “hard voting”.

Soft voting

	class 1 prob	class 2 prob	class 3 prob
classifier A	0.6	0.2	0.2
classifier B	0.5	0.4	0.1
classifier C	0.3	0.4	0.3
average:	0.47	0.33	0.2

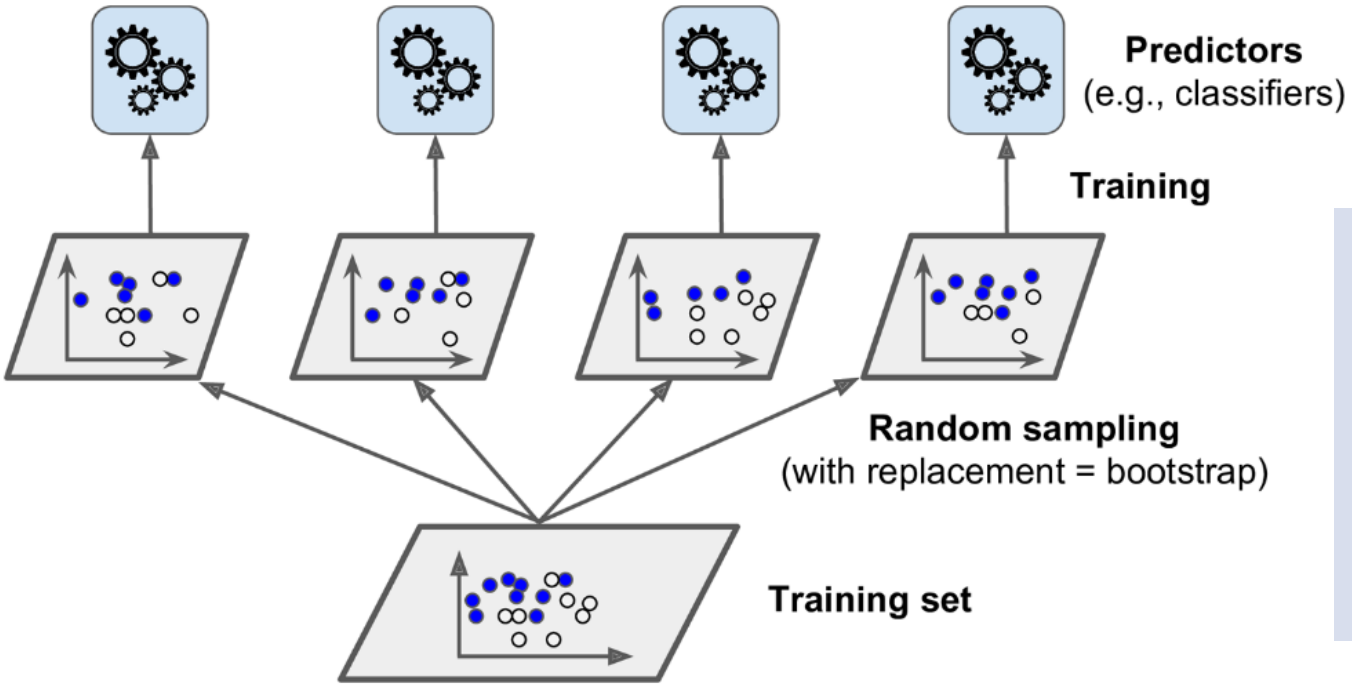
This works when classifiers give probabilities for each class.

Soft voting result here is class 1 – average probability is highest

Bagging = Bootstrap aggregating

In bagging, diversity of predictors is achieved by training them on different versions of the training data.

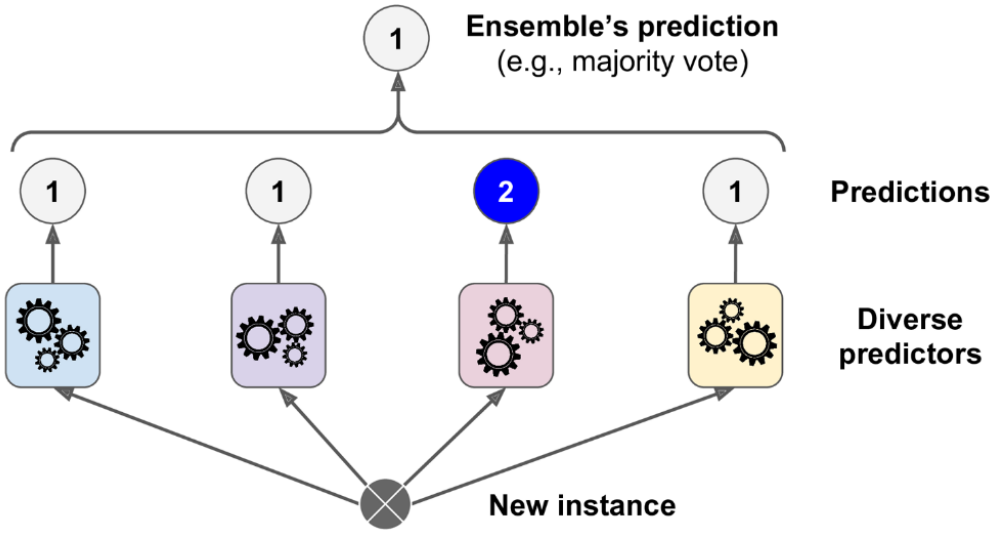
- for each predictor, train using m random samples from the training data set (with replacement)



Typically, a predictor will see about 63% of the training instances in its own training set.

How to aggregate predictor output?

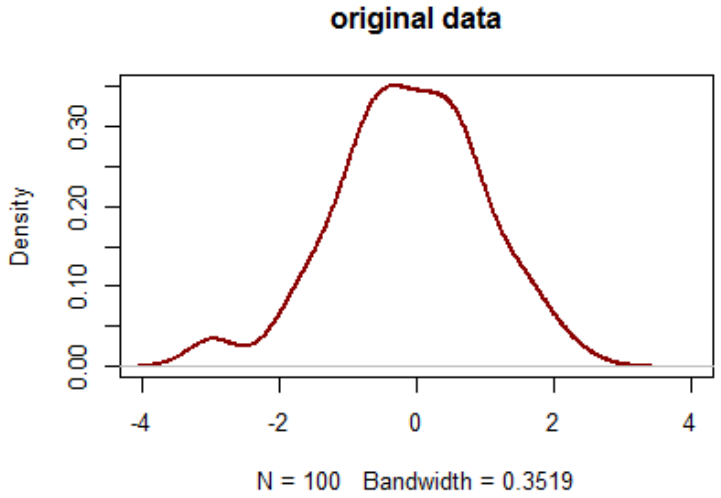
- classification: use the mode of the predictor classes (i.e. most frequent class)
- regression: take the mean of the predictor outputs

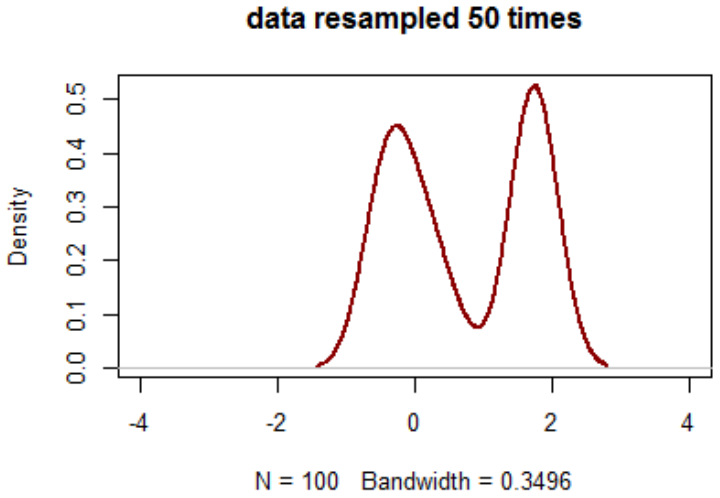


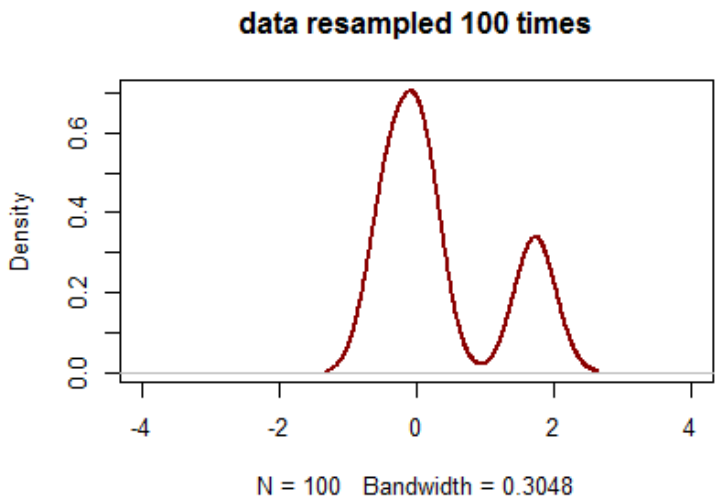
An individual predictor will have higher bias than if it were trained on the complete training set.

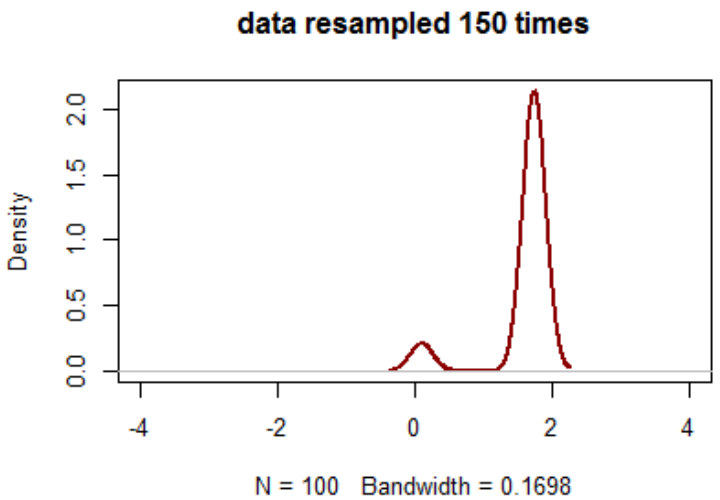
The ensemble will have lower bias and lower variance than an individual predictor trained on the original training set.

Resampling experiment









Pasting

Pasting is just like bagging, but sampling is done without replacement.

Bagging often does better than pasting, because the training sets in bagging are more diverse.

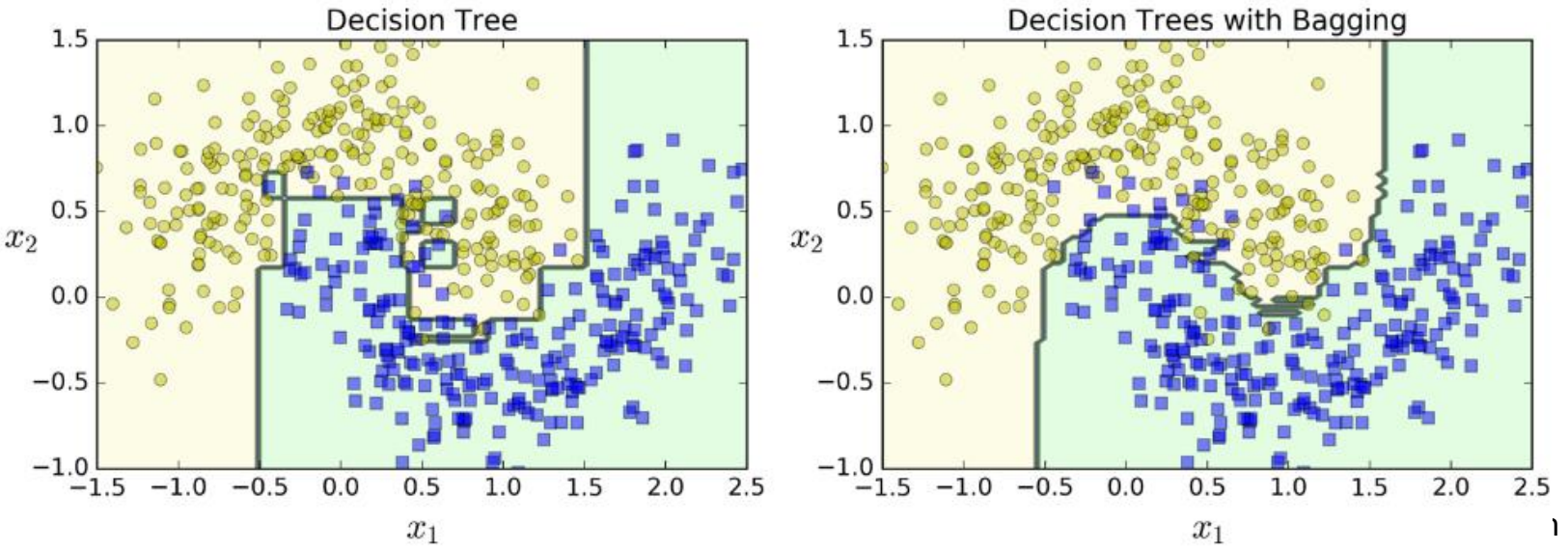
It’s easy to run the individual predictors in parallel, both with bagging and pasting.

Bagging and pasting in Scikit-Learn

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=100, bootstrap=True, n_jobs=-1)

bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```



Out-of-bag evaluation

Cool idea for bagging:

1. A predictor trains on samples from the full training set
2. Some instances (typically 37%) in the full training set aren't sampled
3. Use these "out-of-bag" instances as a validation set

```
>>> bag_clf = BaggingClassifier(
...     DecisionTreeClassifier(), n_estimators=500,
...     bootstrap=True, n_jobs=-1, oob_score=True)
...
>>> bag_clf.fit(X_train, y_train)
>>> bag_clf.oob_score_
0.9013333333333332
```

What about sampling features?

Random subspaces

- keep all training instances, but sample features

Random patches

- sample training instances and features

In sklearn's BaggingClassifier, use these options:

- max_features (default is to use all)
- bootstrap_features (default is to not bootstrap them)

Pros/cons of bagging

Pros:

- easy
- parallelizable
- results usually improve, at least incrementally
- bias and variance are reduced

Cons:

- doesn't help much with high bias models
- loss of explanatory power of the model (this can be very important)
- issues with unbalanced training sets

Summary

- Ensemble methods
- Bagging
- Pasting
- Out-of-bag evaluation
- random subspaces
- random patches