Ensemble Learning

APT 3025: APPLIED MACHINE LEARNING

Wisdom of the Crowd

- Suppose you pose a complex question to thousands of random people, then aggregate their answers.
- In many cases you will find that this aggregated answer is better than an expert's answer.
- This is called the wisdom of the crowd.
- Similarly, if you aggregate the predictions of a group of predictors (such as classifiers or regressors), you will often get better predictions than with the best individual predictor.

Ensemble

- A group of predictors is called an ensemble; thus, this technique is called Ensemble Learning, and an Ensemble Learning algorithm is called an Ensemble method.

Using Multiple Decision Trees

- As an example of an Ensemble method, you can train a group of Decision Tree classifiers, each on a different random subset of the training set.
- To make predictions, you obtain the predictions of all the individual trees, then predict the class that gets the most votes.

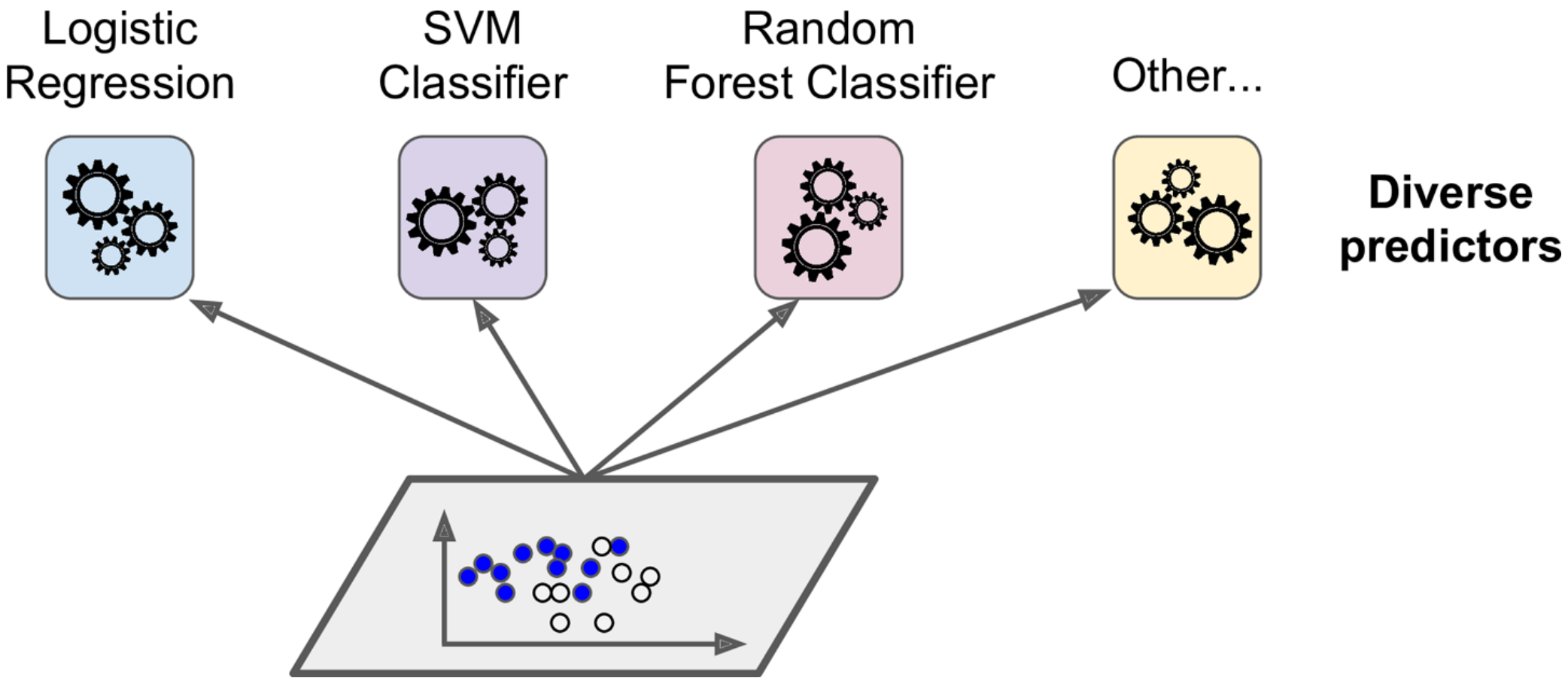
Random Forest

- Such an ensemble of Decision Trees is called a Random Forest, and despite its simplicity, this is one of the most powerful Machine Learning algorithms available today.

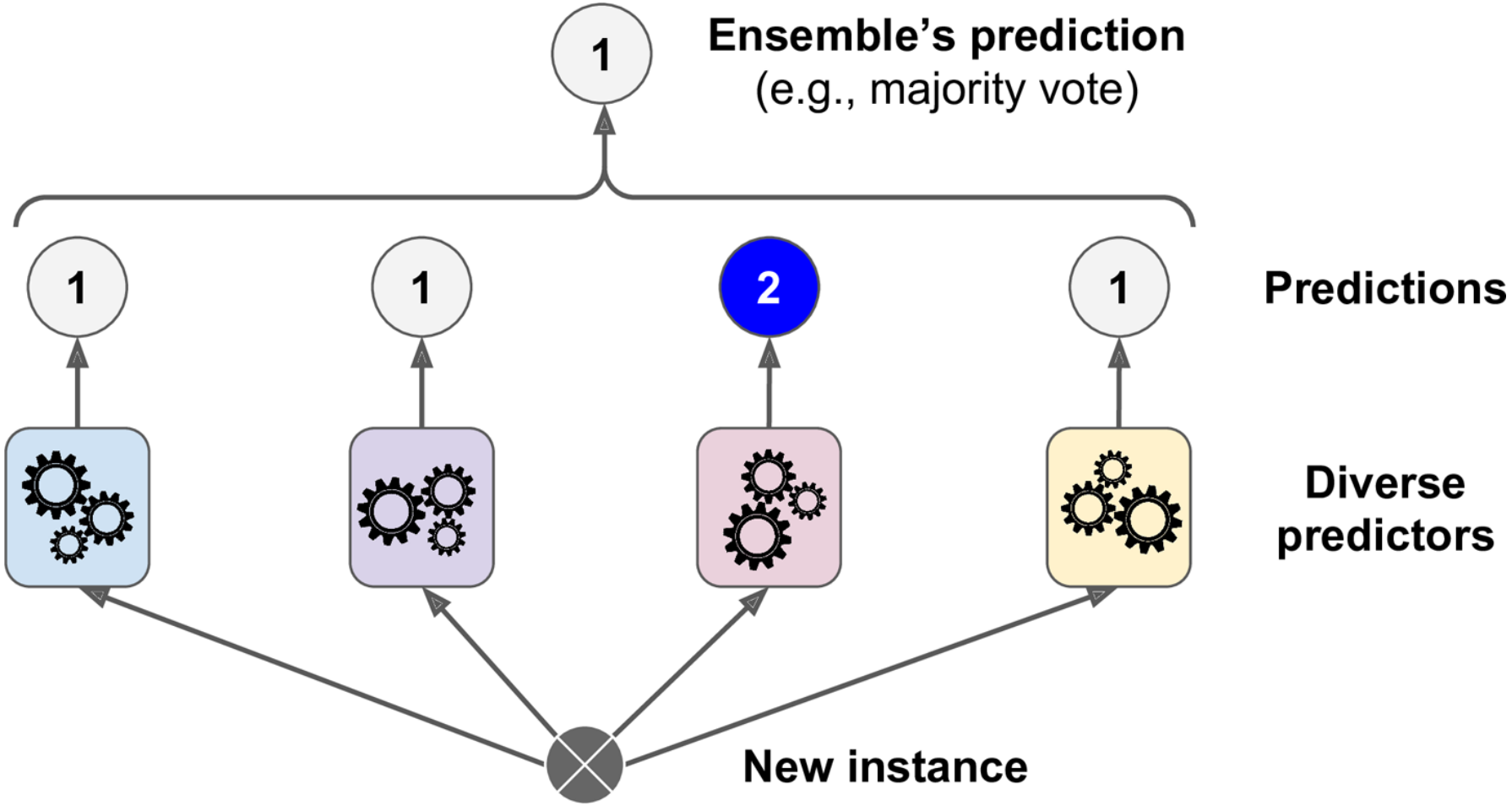
Hard Voting Classifier

- Suppose you have trained a few classifiers, each one achieving about 80% accuracy. You may have a Logistic Regression classifier, a Support Vector Machine classifier, a Random Forest classifier, a K-Nearest Neighbors classifier, and perhaps a few more.
- A very simple way to create an even better classifier is to aggregate the predictions of each classifier and predict the class that gets the most votes. This majority-vote classifier is called a hard voting classifier.

Training Diverse Classifiers



Hard Voting


1
Ensemble's prediction
(e.g., majority vote)
1
1
2
1
Predictions
Diverse
predictors
New instance

The whole is greater than the sum of its parts

- This voting classifier often achieves a higher accuracy than the best classifier in the ensemble.
- In fact, even if each classifier is a weak learner (meaning it does only slightly better than random guessing), the ensemble can still be a strong learner (achieving high accuracy), provided there are a sufficient number of weak learners and they are sufficiently diverse.

Ensemble Example

In [1]:
```
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [2]:
```
log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()
```

In [3]:
```
X, y = make_moons(n_samples=100, noise=0.15)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Ensemble Example continued

In [4]:
```
voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='hard')
voting_clf.fit(X_train, y_train)
```

Out[4]:
```
VotingClassifier(estimators=[('lr', LogisticRegression()),
                             ('rf', RandomForestClassifier()), ('svc', SVC())])
```

In [5]:
```
from sklearn.metrics import accuracy_score
for  clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.88
RandomForestClassifier 1.0
SVC 0.96
VotingClassifier 0.96
```

Soft Voting

- If all classifiers are able to estimate class probabilities (i.e., they all have a predict_proba() method), then you can tell Scikit-Learn to predict the class with the highest class probability, averaged over all the individual classifiers.
- This is called soft voting.

Soft Voting

- Soft voting often achieves higher performance than hard voting because it gives more weight to highly confident votes.
- All you need to do is replace voting="hard" with voting="soft" and ensure that all classifiers can estimate class probabilities.

Example

- Hard voting would classify this particular instance in class 1 because it's the majority.

classifier	predicted target label
classifier #1	class 1
classifier #2	class 1
classifier #3	class 0

Example continued

- In soft voting, every classifier is assigned a weight coefficient, which stands for the importance of the classifier in the voting procedure. For the sake of simplicity, let's assume all three classifiers have the same weight:
  - w1 = w2= w3 = 1

Example continued

- The three classifiers would then go on to predict the probability of a particular data point belonging to each of the available class labels:

classifier	class 0	class 1
classifier #1	0.3 w_1	0.7 w_1
classifier #2	0.5 w_2	0.5 w_2
classifier #3	0.4 w_3	0.6 w_3

Example continued

- The voting classifier would then compute the weighted average for each class label:
  - For class 0, we get a weighted average of (0.3 w1 + 0.5 w2 + 0.4 w3) / 3= 0.4
  - For class 1, we get a weighted average of (0.7 w1 + 0.5 w2 + 0.6 w3) / 3 = 0.6
- Because the weighted average for class 1 is higher than for class 0, the ensemble classifier would go on to predict class 1.

Soft vs. hard voting example 2

Model 1	Model 2	Model 3
0.6Y 0.4N	0.7Y 0.3N	0.1Y 0.9N
Model 4	**Model 5**	
0.2Y 0.8N	0.6Y 0.4N	

Hard voting: 3 yes vs 2 No. Conclusion: Yes

Yes (0.6 + 0.7 + 0.1 + 0.2 + 0.6)/5 = 0.44
No (0.4 + 0.3 + 0.9 + 0.8 + 0.4)/5 = 0.56
Soft voting conclusion: No

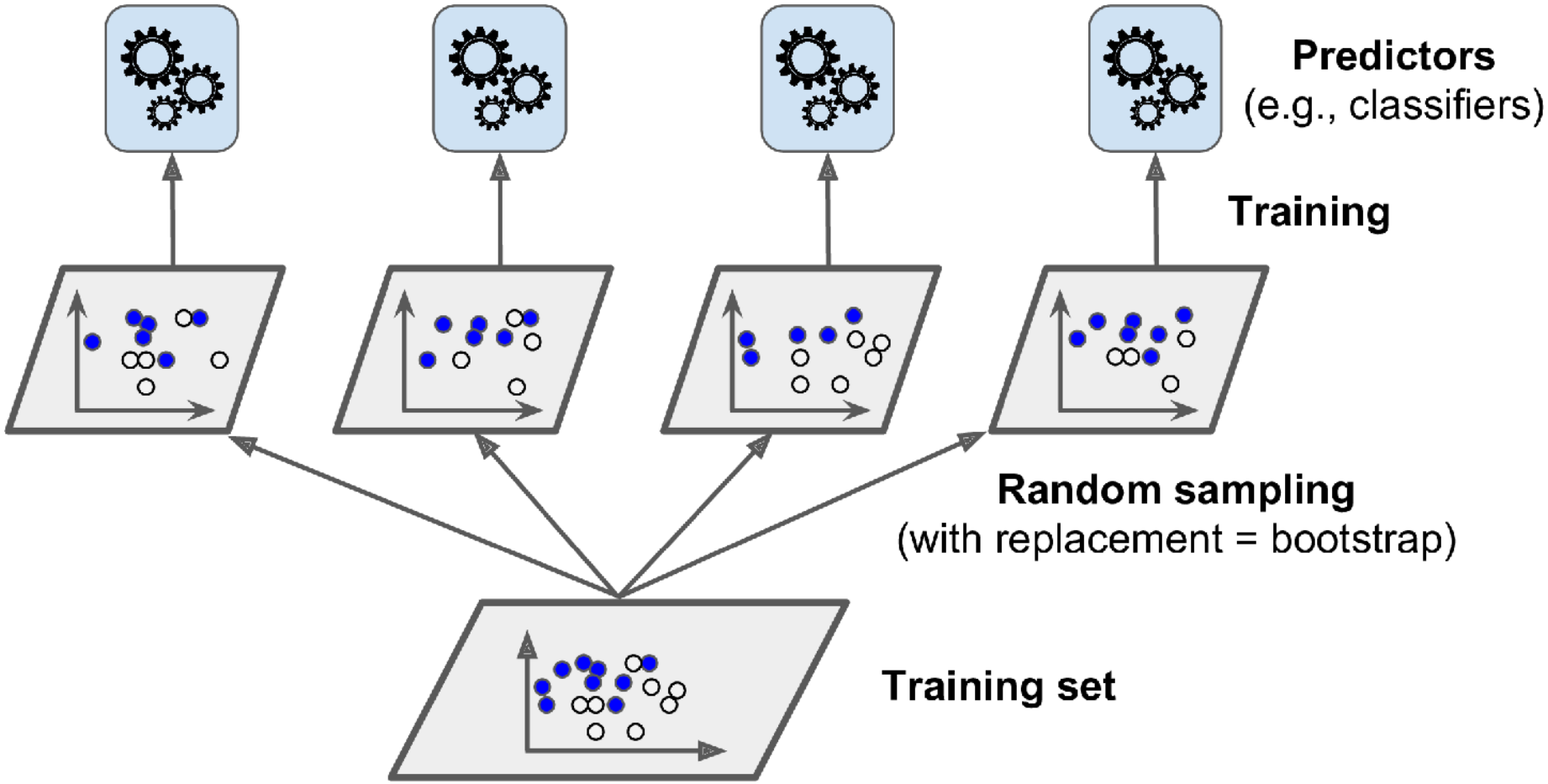
Bagging and Pasting

- One way to get a diverse set of classifiers is to use very different training algorithms, as just discussed.
- Another approach is to use the same training algorithm for every predictor and train them on different random subsets of the training set.

Bagging and Pasting

- When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating).
- When sampling is performed without replacement, it is called pasting.

Bagging and Pasting



Predicting with Bagging and Pasting

- Once all predictors are trained, the ensemble can make a prediction for a new instance by simply aggregating the predictions of all predictors.
- The aggregation function is typically the statistical mode (i.e., the most frequent prediction, just like a hard voting classifier) for classification, or the average for regression.

Training and Predicting in Parallel

- Predictors in an ensemble can all be trained in parallel, via different CPU cores or even different servers.
- Similarly, predictions can be made in parallel.
- This is one of the reasons bagging and pasting are such popular methods: they scale very well.

Bagging and Pasting in Scikit-Learn

- Scikit-Learn offers a simple API for both bagging and pasting with the BaggingClassifier class (or BaggingRegressor for regression).
- The following code trains an ensemble of 500 Decision Tree classifiers: each is trained on 100 training instances randomly sampled from the training set with replacement (this is an example of bagging, but if you want to use pasting instead, just set bootstrap=False).

Bagging and Pasting in Scikit-Learn

- The n_jobs parameter tells Scikit-Learn the number of CPU cores to use for training and predictions (–1 tells Scikit-Learn to use all available cores).

In [8]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=50, bootstrap=True, n_jobs=-1)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

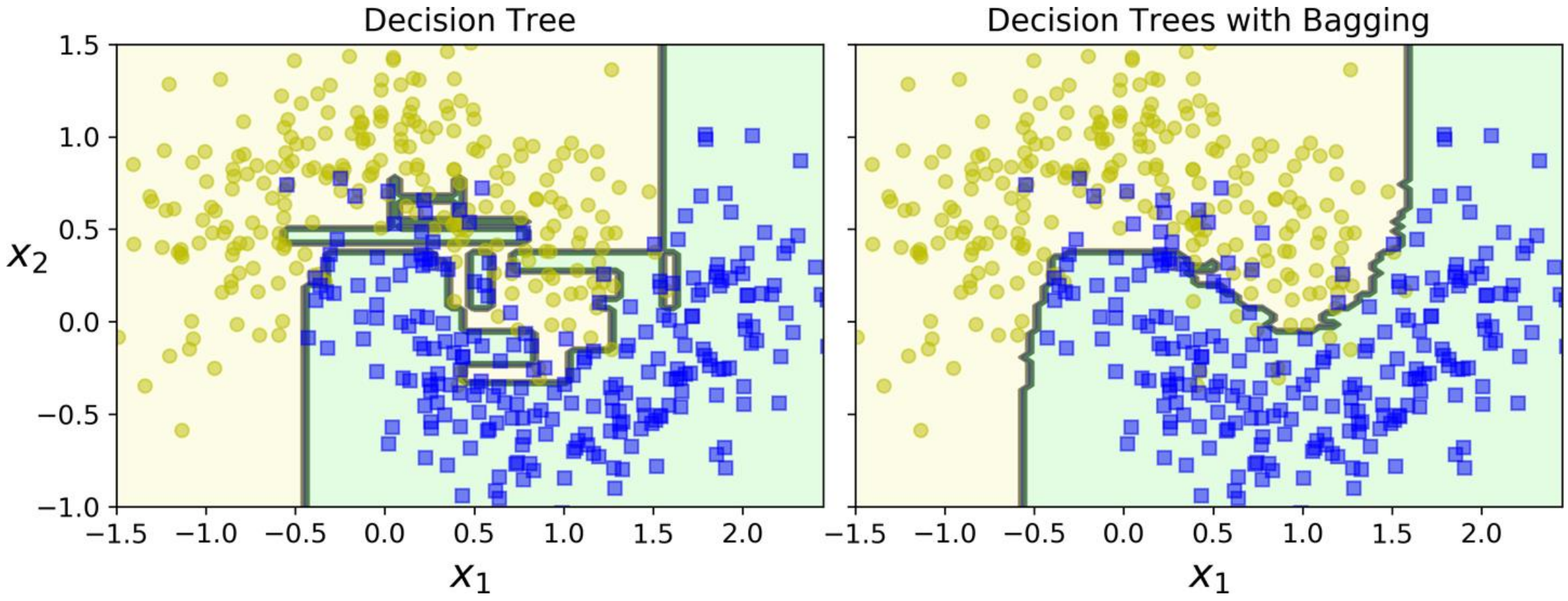
In [10]:
```
print(accuracy_score(y_test, y_pred))
```

```
0.96
```

Single Tree vs. Forest

- The figure below compares the decision boundary of a single Decision Tree with the decision boundary of a bagging ensemble of 500 trees (from the preceding code), both trained on the moons dataset.
- As you can see, the ensemble's predictions will likely generalize much better than the single Decision Tree's predictions.

Single Tree vs. Forest Decision Boundaries



Random Forests

- As we have discussed, a Random Forest is an ensemble of Decision Trees, generally trained via the bagging method (or sometimes pasting), typically with max_samples set to the size of the training set.
- Instead of building a BaggingClassifier and passing it a DecisionTreeClassifier, you can instead use the RandomForestClassifier class, which is more convenient and optimized for Decision Trees (similarly, there is a RandomForestRegressor class for regression tasks).

Random Forest

- The following code uses all available CPU cores to train a Random Forest classifier with 500 trees (each limited to maximum 16 nodes):

In [13]:
```
from sklearn.ensemble import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)
y_pred_rf = rnd_clf.predict(X_test)
```

In [14]:
```
print(accuracy_score(y_test, y_pred))
```

```
0.96
```

Boosting

- Boosting refers to any Ensemble method that can combine several weak learners into a strong learner.
- The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor.
- There are many boosting methods available, but by far the most popular are AdaBoost (short for Adaptive Boosting) and Gradient Boosting.
- Boosting is beyond the scope of this course and will not be discussed any further.

References

- Much of the content in these slides was taken from
  - *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd Edition by Aurélien Géron, O'Reilly Media, Inc., 2019
- A small part, specifically the example to illustrate hard and soft voting, was taken from
  - *Machine Learning for OpenCV by Michael Beyeler*, Packt Publishing, 2017