Should we trust an expert or a crowd?

You are at an auction trying to determine whether or not a specific painting is a fraud. You decide to ask around and get opinions from people at the auction site.

Option A:

An expert art inspector has an accuracy of 80%. There is only one on site.

Option B:

Other attendees, who are not quite as knowledgeable, accurately identify whether or not a piece of art is fake 55% of the time. There are at most 100 other attendees you can ask.

Assuming you can only have time for one option, which would you choose? Why?

Hint: Think about how you might be able to use a binomial distribution perhaps….

Bagging Methods with Random Forests

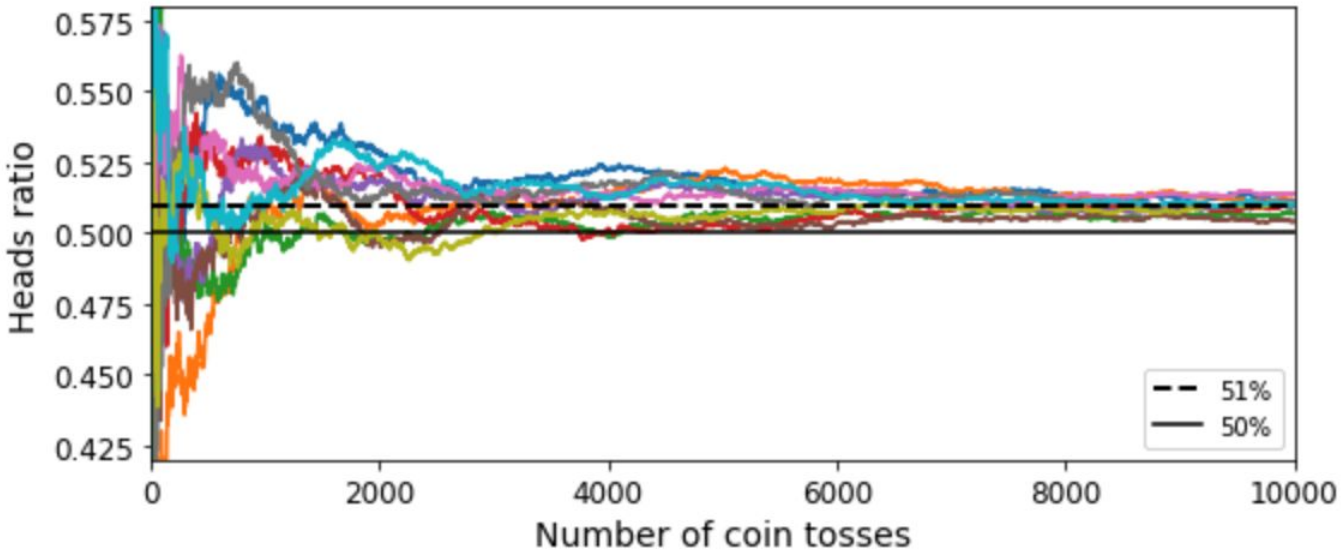
Data Science Immersive

Wisdom of the Crowd!

At a 1906 fair in Plymouth, England, statistician Francis Galton noticed how when 800 people guessed how much a “dressed” ox weighed. It turns out that the actual guess had only a 1% error from the median guess

Why Ensemble Methods

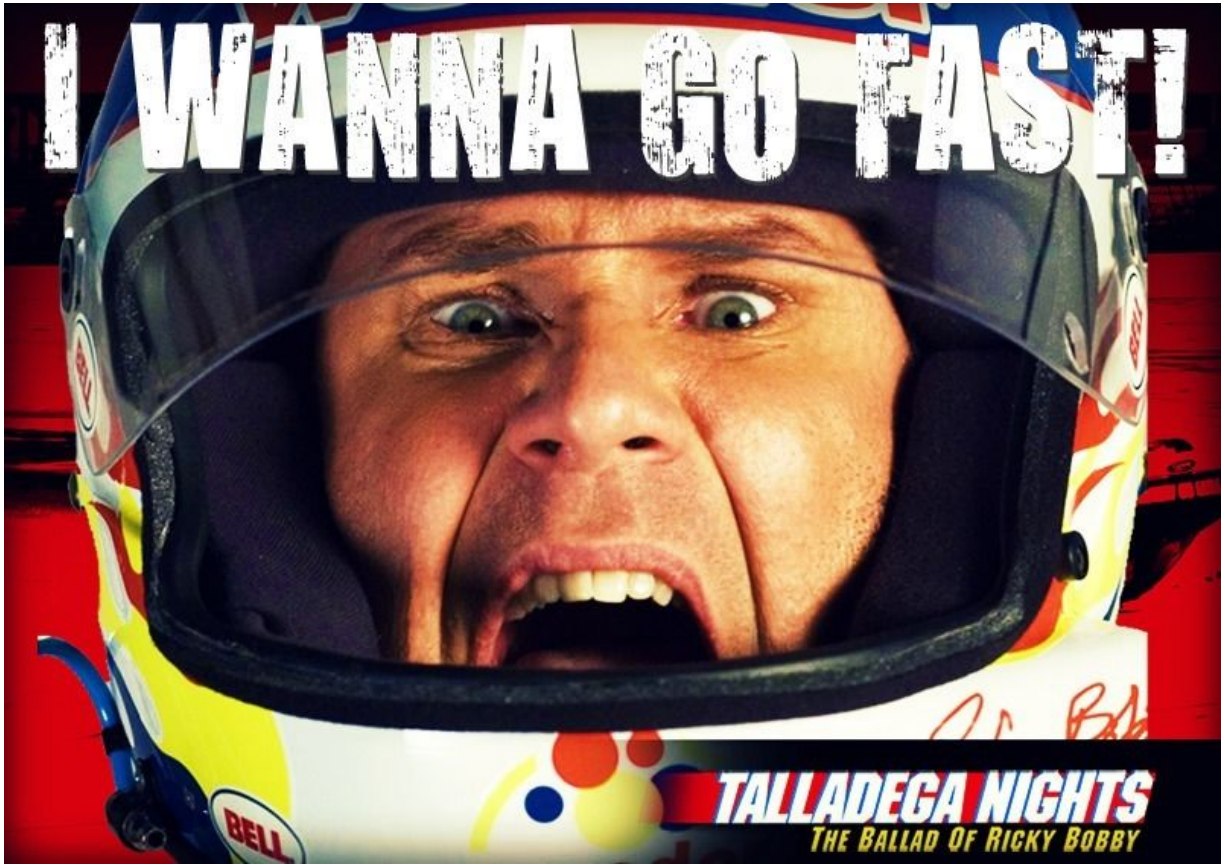
Which would you rather use to solve a problem, a bunch of simple models that aren't that accurate or one complex model that is very accurate?



You have a biased coin that has a 51% chance of coming up heads and a 49% chance of coming up tails. If you toss it 1,000 times the probability of having a majority of heads is 75%. If you increase your tosses to 10,000, that probability rise to 97%

Objectives

- Understanding Bootstrapping and Aggregating (Bagging) of models
- Understanding the Random Forests algorithm

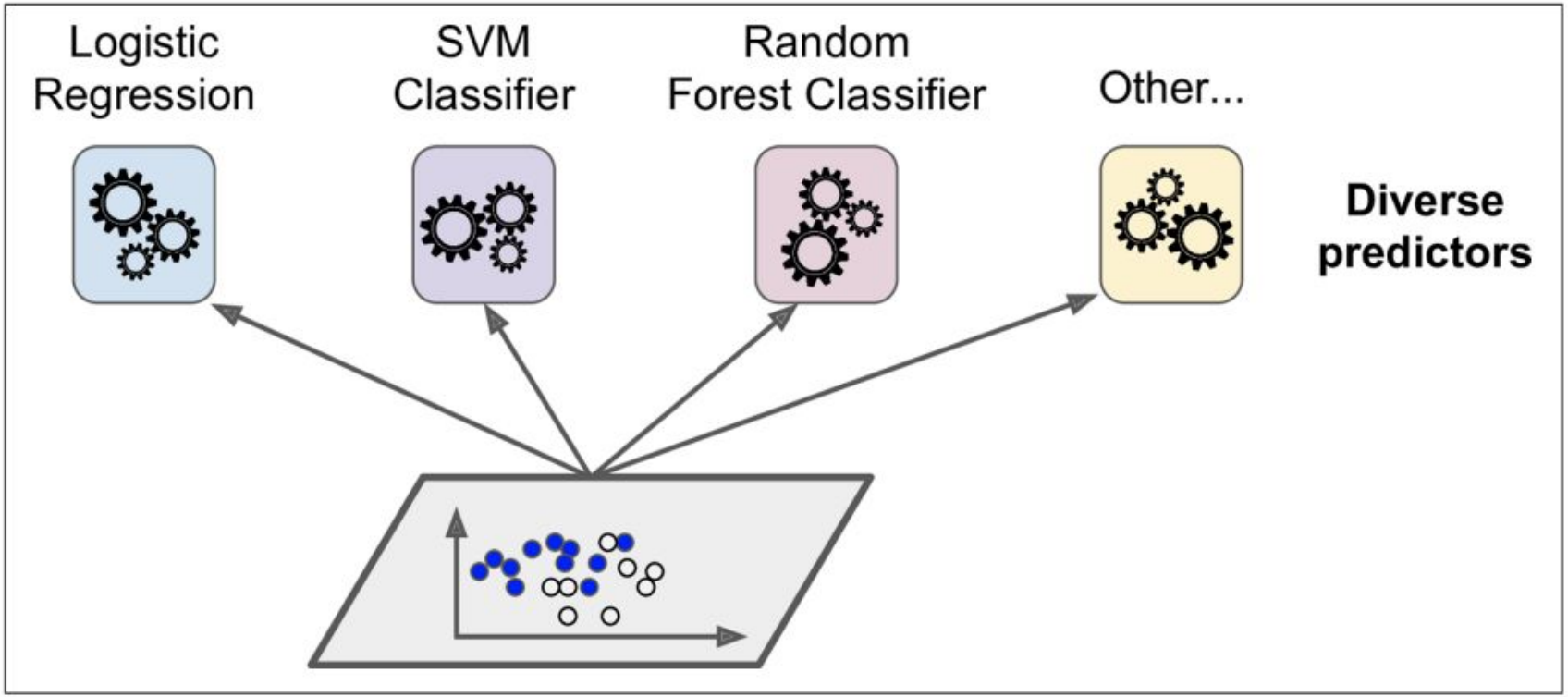

I WANNA GO FAST!
TALLADEGA NIGHTS
THE BALLAD OF RICKY BOBBY
BELL

Ensemble Methods

We can learn a lot by combining many different learners. This is called an *ensemble method*.

In essence, we can combine the output of models that are not accurate enough on their own. We can get wisdom from the crowd!

Ensemble Methods

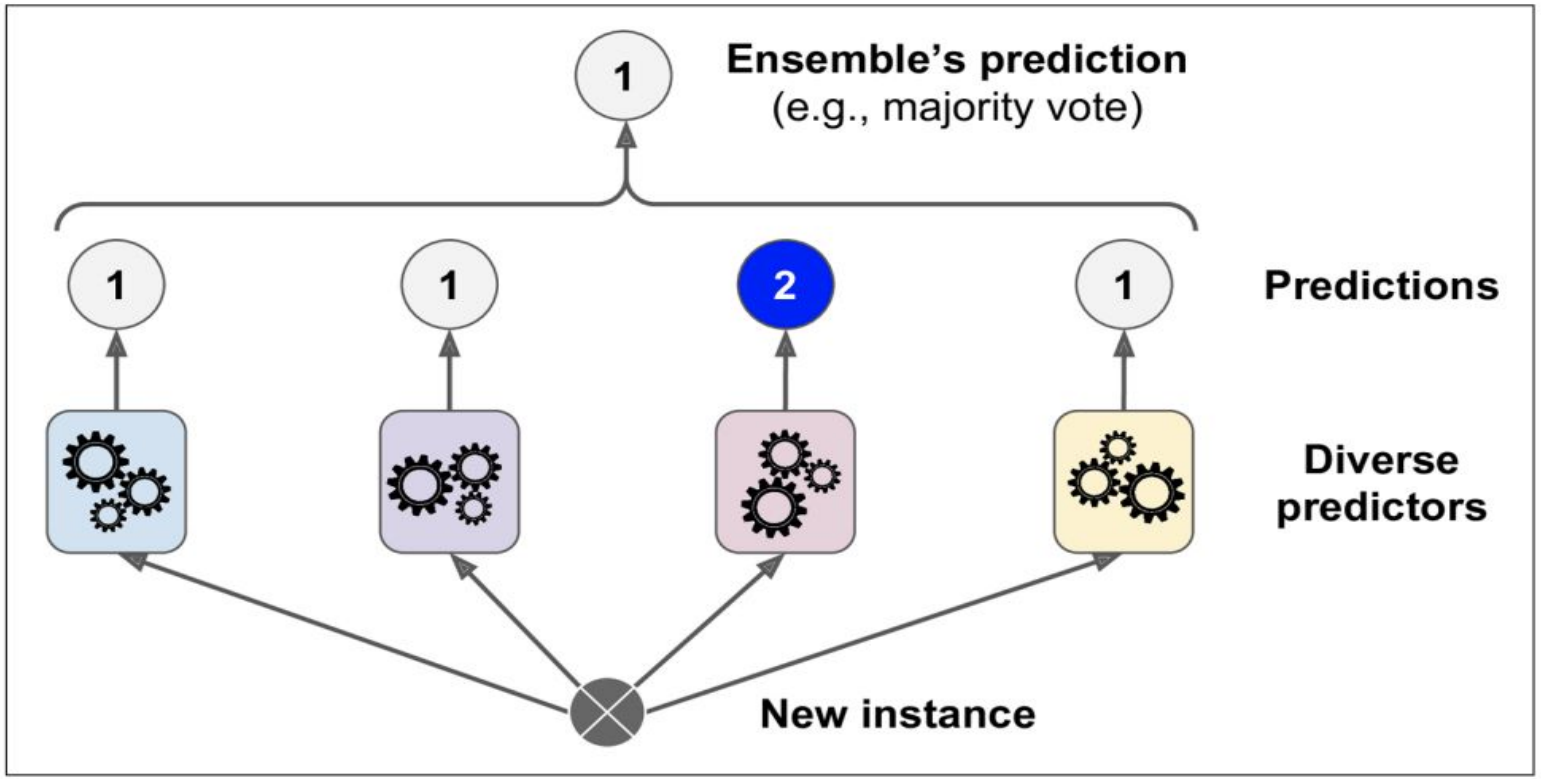


Ensemble Methods in SKlearn

In sklearn, you can create custom ensemble models by making use of the VotingClassifier/VotingRegressor. This is intended to be used with conceptually different models.

- Within the VotingClassifier, you can specify “Majority Class” vote or “Soft” vote.
  - Hard: Select the class that is predicted most
  - Soft: Take an average of the probabilities for each class, make decision based off of that

Hard Voting Example



Coding up a Voting Classifier

```
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)
```

https://github.com/ageron/handson-ml/blob/master/07_ensemble_learning_and_random_forests.ipynb

Individual Model vs. Ensemble Model

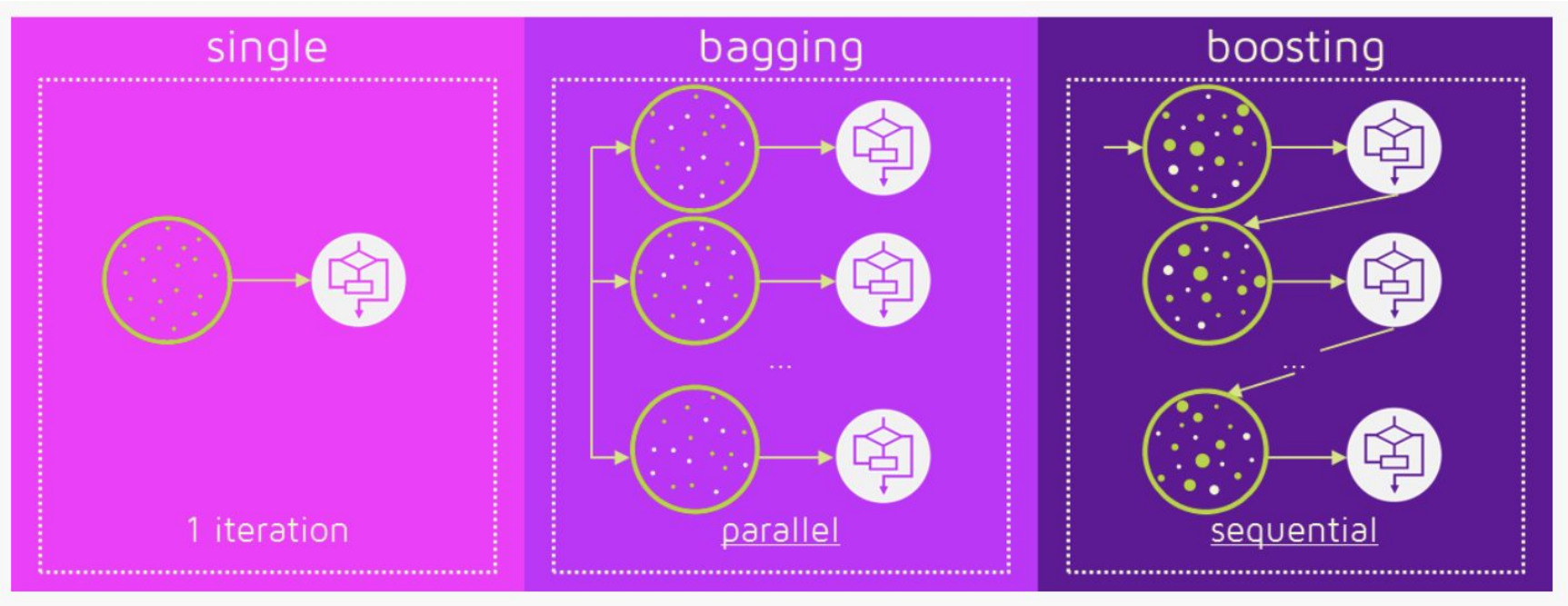
```
>>> from sklearn.metrics import accuracy_score
>>> for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
>>>     clf.fit(X_train, y_train)
>>>     y_pred = clf.predict(X_test)
>>>     print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
LogisticRegression 0.864
RandomForestClassifier 0.872
SVC 0.888
VotingClassifier 0.896
```

https://github.com/ageron/handson-ml/blob/master/07_ensemble_learning_and_random_forests.ipynb

Ensemble Methods

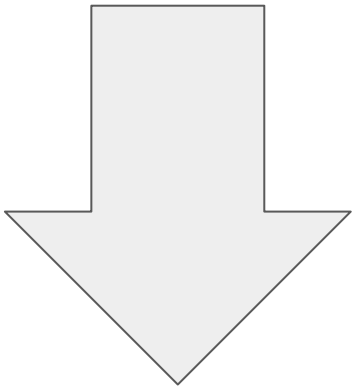
Instead of taking multiple different training algorithms, we can use the same algorithm multiple times.

There are two main types of ensemble methods: **Bagging** and **Boosting**



Instead of taking multiple different training algorithms, we can use the same algorithm multiple times. This is called **Bagging.**

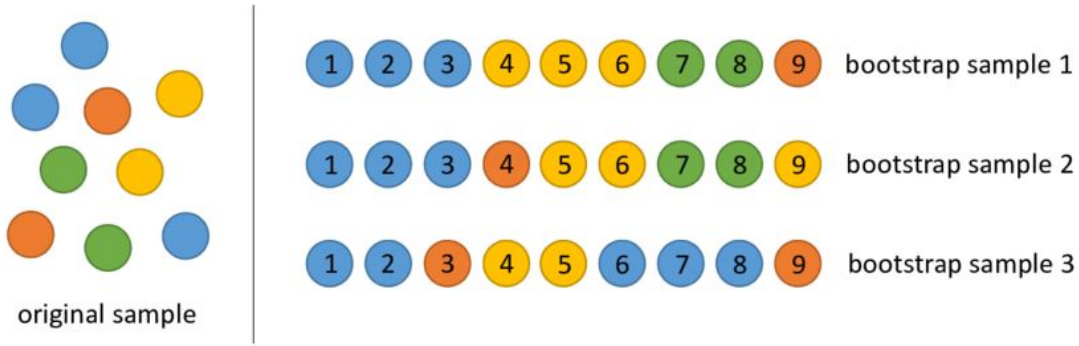
Bootstrapping + **Agg**regating



Bagging

Bootstrapping

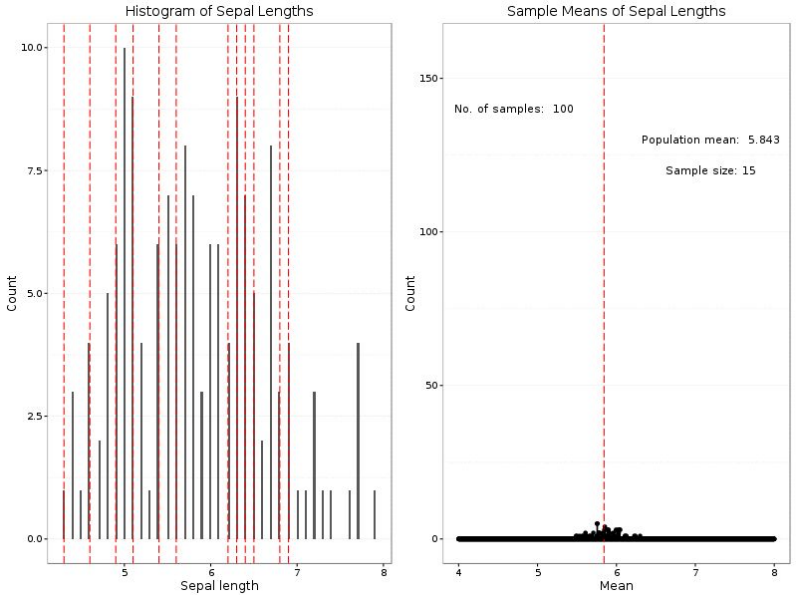
- Some models are prone to overfitting. (They have a high variance). This is especially true for decisions trees that are built out to full “purity” in each of the leaves.
- To help prevent overfitting we take bootstrap samples **with replacement** from our training data that is the same size as our training data.



Bootstrapping

What principle allows us to use bootstrapping? Hint what principle, from our knowledge of statistics, talk about what will happen if we take multiple samples from the same dataset?

Central Limit Theorem



Aggregating

- Once we have taken bootstrapped samples, we fit a model to each sample.

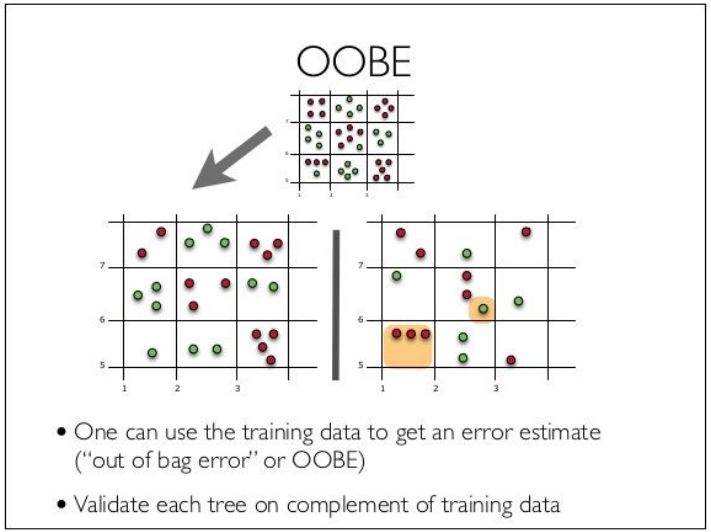
  *This individual model might have a **low bias and high variance.***
- Repeat this process for however many models you want in your ensemble.
- In order to predict a data point we feed the data through all of the bootstrapped models.
- For classification, the models will vote either using hard voting or soft voting.
- For regression, the algorithm will take an average of all the models.

Out-of-Bag Error

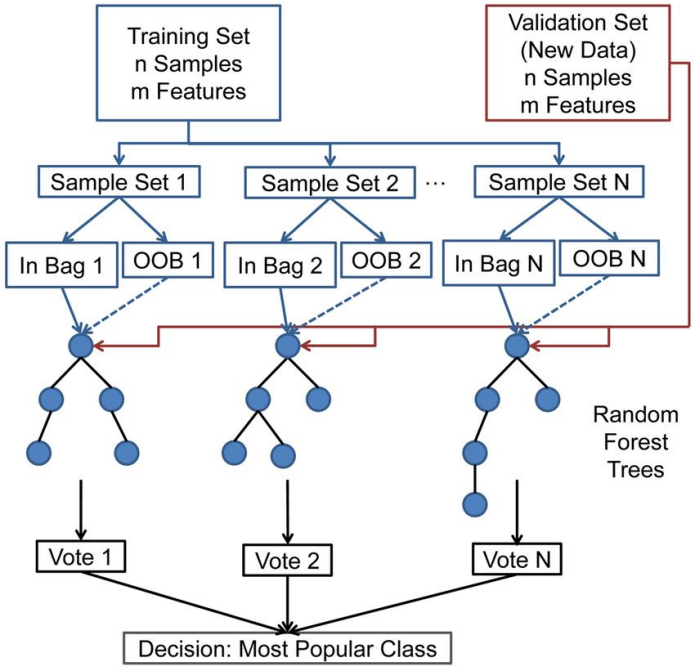
When using bootstrapping our sample, only ⅔ of the observations will be trained on each model. (Why 2/3?)

Performing cross validation on bagged classifiers/regressors is challenging because it can be computationally expensive.

Rather, we can look at every observation and make a prediction for each of the data points for which that data points was not used to create the tree.



Out-of-Bag Error



It's essentially just a form of cross-validation!

Decision Tree Review

Can be used for both Regression and Classification problems (Classification and Regression Trees)

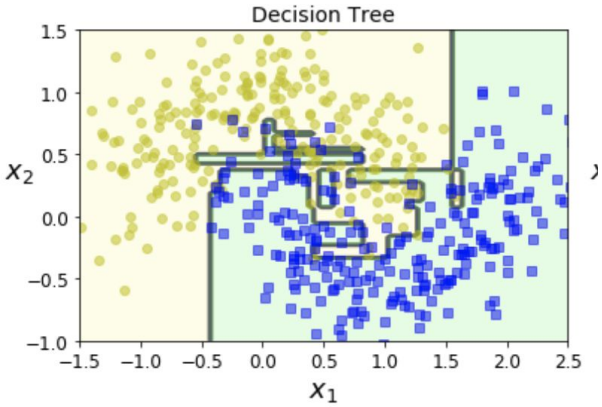
Advantages of Decision Trees:

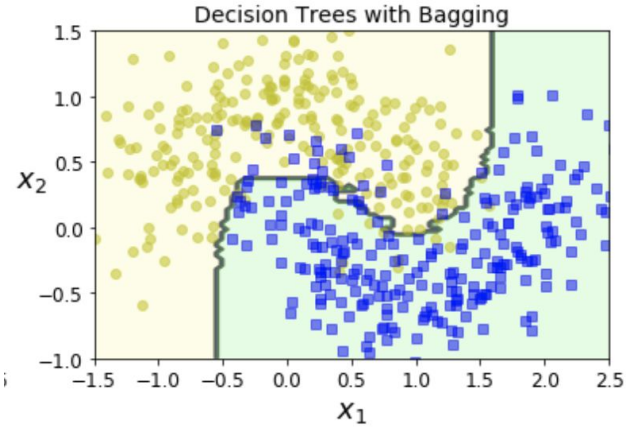
- Work well with non-linear relationships
- Easy to interpret!
- Implicit feature selection
- They are fairly robust to outliers

Disadvantages of Decision Trees:

- Decision trees tend to overfit, especially if they are completely pure.
- Instability: Small changes in the input data can cause large changes to the structure of the tree.

Decision Tree v. Decision Trees with Bagging





Potential Issue with Bagging Decision Trees?

Will the trees created after bootstrapping be diverse? Why or why not?

Think about how a decision tree makes each of its splits.

The Issue with Bagging Decision Trees

The issue with bagging is that each one of the trees might be correlated to each other. There might be a feature that is powerful in generating a separation between different categories, which results in trees that are correlated to one another despite being from bootstrapped samples.

We need to do something to ensure that the trees of our bootstrapped samples are not correlated with one another…..

Random Patches and Random Subspace

Similar to how we took a sample of data points, we can take a sample of features for each model we create. This is particularly useful when you are dealing with high-dimensional inputs (such as images).

Bootstrapping: Takes random sample of observations.

Random Subspace: Takes a random sample of the features.

Random Patches: Takes a random sample of both observations and features.

In SKLearn

The BaggingClassifier class supports sampling the features as well. This is controlled by two hyperparameters: **max_features** and **bootstrap_features**. Thus, each predictor can be trained on a random subset of the input features.

Bootstrapping: bootstrap=True, max_samples = (x>=1)

Random Subspace: bootstrap=False, bootstrap_features=True, max_features= (x>1.0)

Random Patches = bootstrap=True, bootstrap_features=True, max_features= (x>1.0)

Random Forests

A specific implementation of a bagging method is a **Random Forest**.

Random Forest is a uses multiple decision trees to try and predict the output variable.

Random Forest uses both a subsample of the observations and a subsamples of the features.

Note: The subsample of the features happens at each decision node, not for each entire tree.

Random Forests: Max Features

The max number of features considered when finding best split. By increasing the number of features your processing speed will decrease and your trees will be more diverse.

max_features-(*integer, float, string, or None*)-Default=”*auto*”

An *integer* then you should think carefully about max_features at each split because the number is basically up to you.

A *float* is treated as a percentage (max_features x number of features).

auto or *sqrt* will set it to the square root of the number of features (sqrt(n_features)).

log2 will set it to equal *log2(n_features).*

Random Forest Hyperparameters

n_estimators : the number of trees in the forest

criterion: “gini”,”entropy”

max_features: the number of random features to be considered when looking for the best split

max_depth: the maximum number of levels of a tree

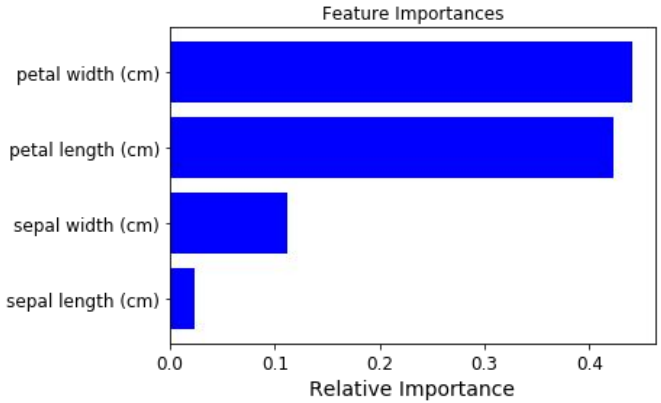
bootstrap: whether or not bootstrap samples are used to build trees

oob_score: whether or not to use out-of-bag samples to estimate the generalization accuracy

n_jobs: how many cores you want to use when training your trees

Feature Importances

We are able to obtain “feature importance” for each of the variables. This is calculated by averaging the total amount that the gini index is decreased/mean decrease impurity increased by splits over a given predictor, averaged over all B trees.

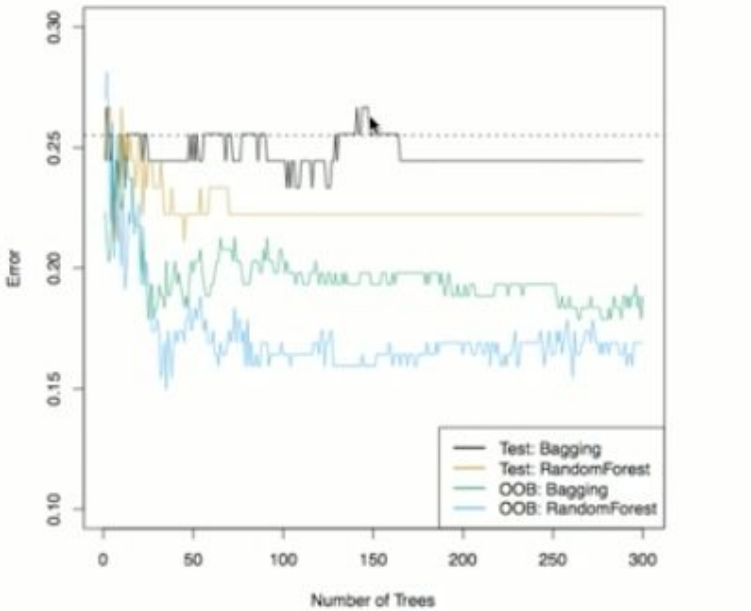


There are many other ways to determine the feature importances of Random Forest. Check them out here
https://papers.nips.cc/paper/4928-understanding-variable-importances-in-forests-of-randomized-trees.pdf
https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e

Comparing Performances

In general:

Random Forest > Bagging > Decision Trees



Random Forest Advantages/Disadvantages

Advantages

- A very powerful model. Will nearly always outperform decision trees
- Able to detect non-linear relationships well
- Harder than other models to overfit

Disadvantages

- Not as interpretable as decision trees
- Many hyperparameters to tune (GridSearch is your friend!)

Question to ponder…….

How do Random Forests handle the bias-variance tradeoff?

Additional Resources

https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm