Module-4.9-Ensemble Models

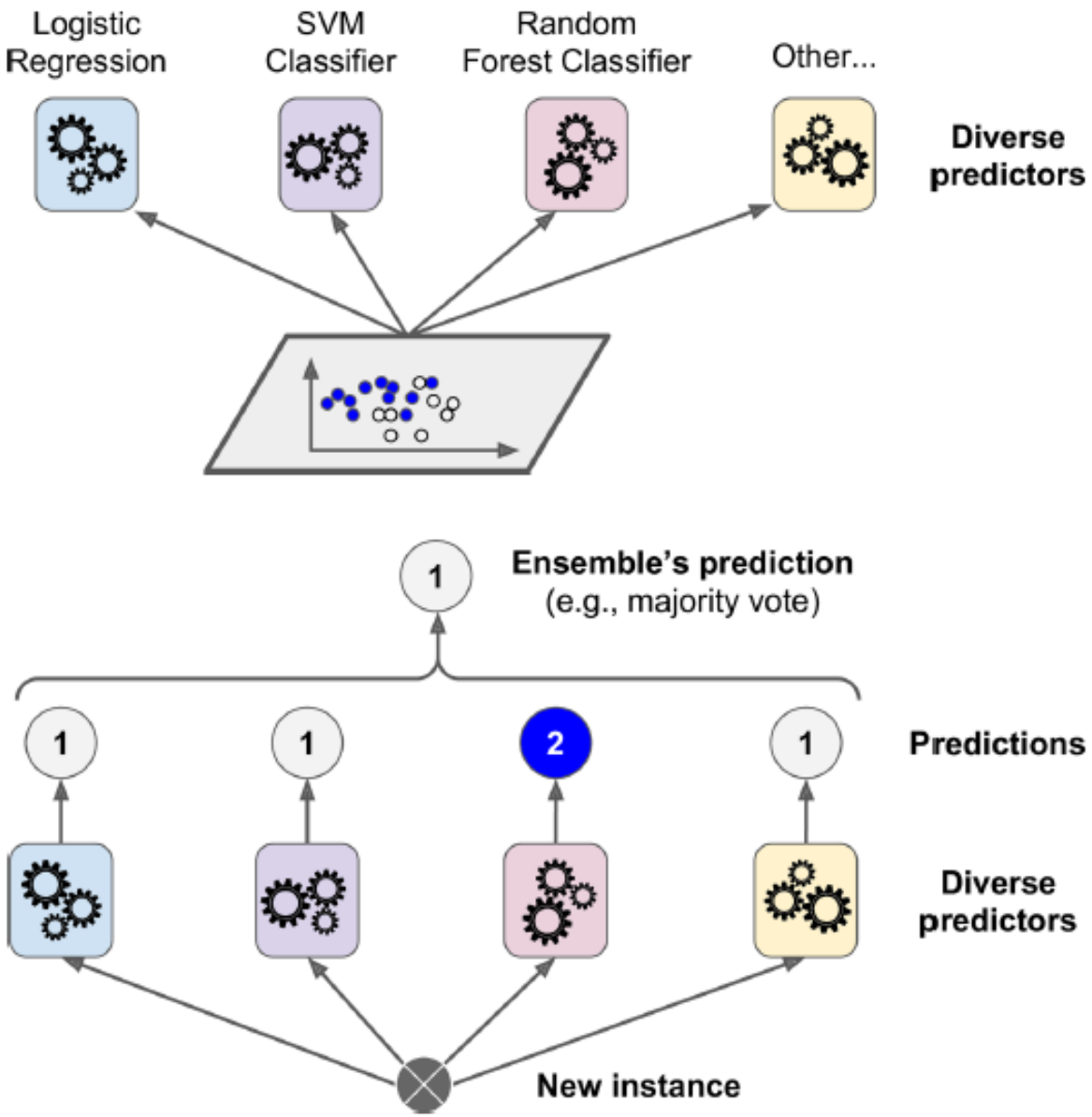
Presented by Yasin Ceran

Table of Contents

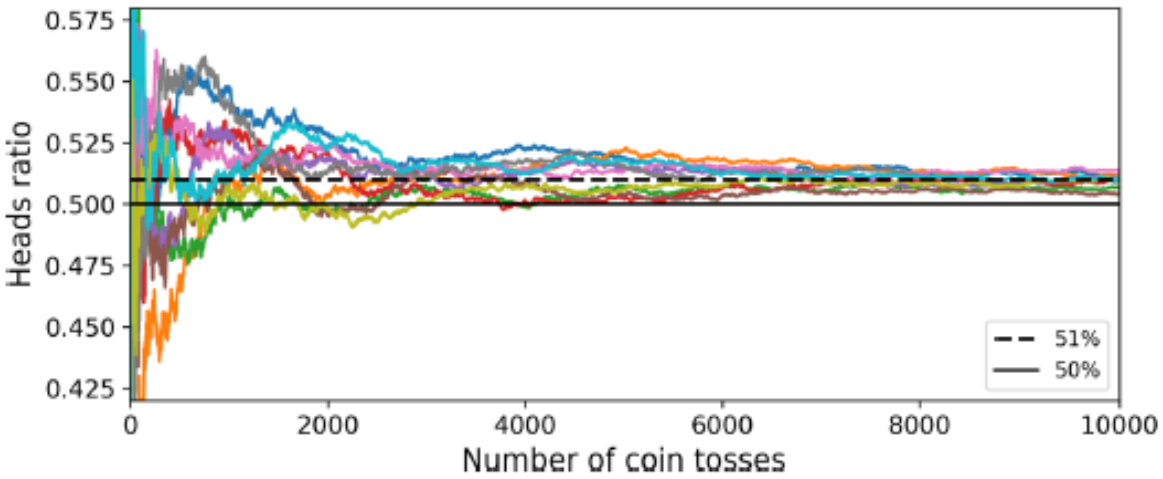
Poor Man's Ensembles

- Build different models
- Average the result
- More models are better – if they are not correlated.
- Also works with neural networks
- You can average any models as long as they provide calibrated ("good") probabilities

Voting Classifiers



Voting Classifier and Law of Large Numbers

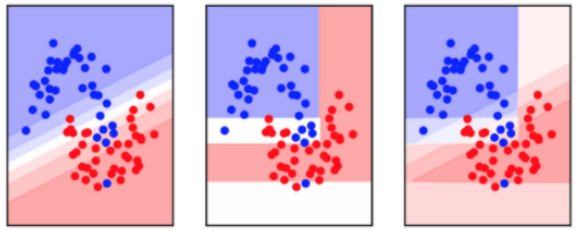


- A slightly biased coin with 51% chance of coming heads
- Toss it 1,000 times, it is 75% more likely to have heads as majority
- Toss it 10,000 times, it is 97% more likely to have heads as majority
- Keep in mind that each toss is independent from the others

Voting Classifier

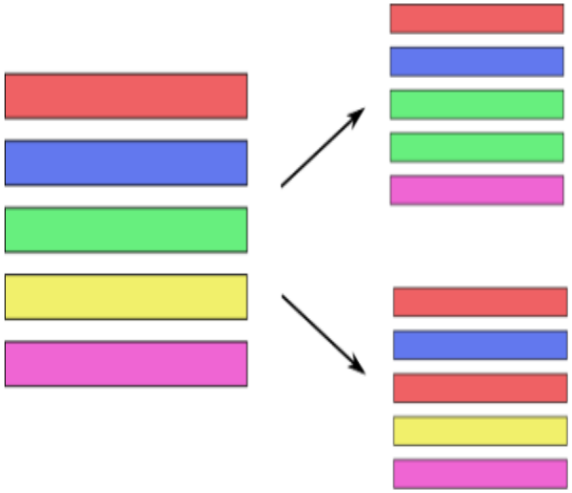
```
voting = VotingClassifier(
    [('logreg', LogisticRegression(C=100)),
     ('tree', DecisionTreeClassifier(max_depth=3, random_state=0))],
    voting='soft')
voting.fit(X_train, y_train)
lr, tree = voting.estimators_
voting.score(X_test, y_test), lr.score(X_test, y_test), tree.score(X_test, y_test)
```

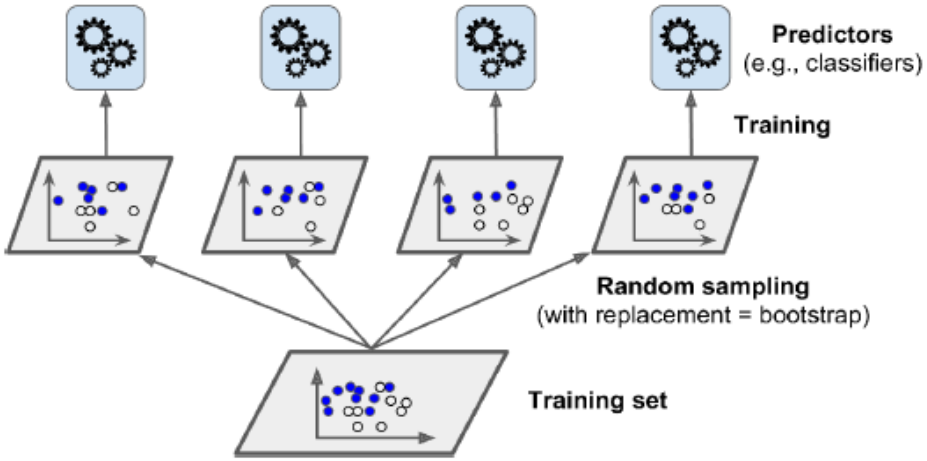
```
0.88 0.84 0.80
```



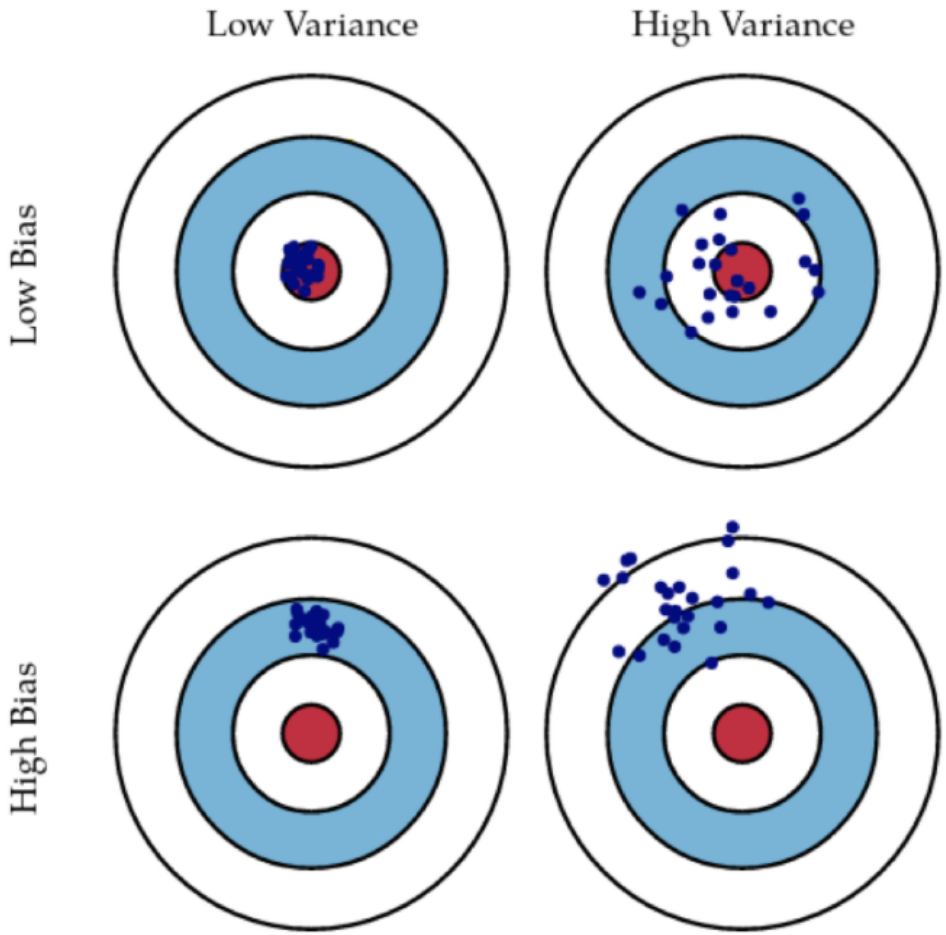
Bagging (Bootstrap AGGregation)

- Generic way to build "slightly different" models
- Use the same training algorithm for every predictor and train them on different random subsets of the training set

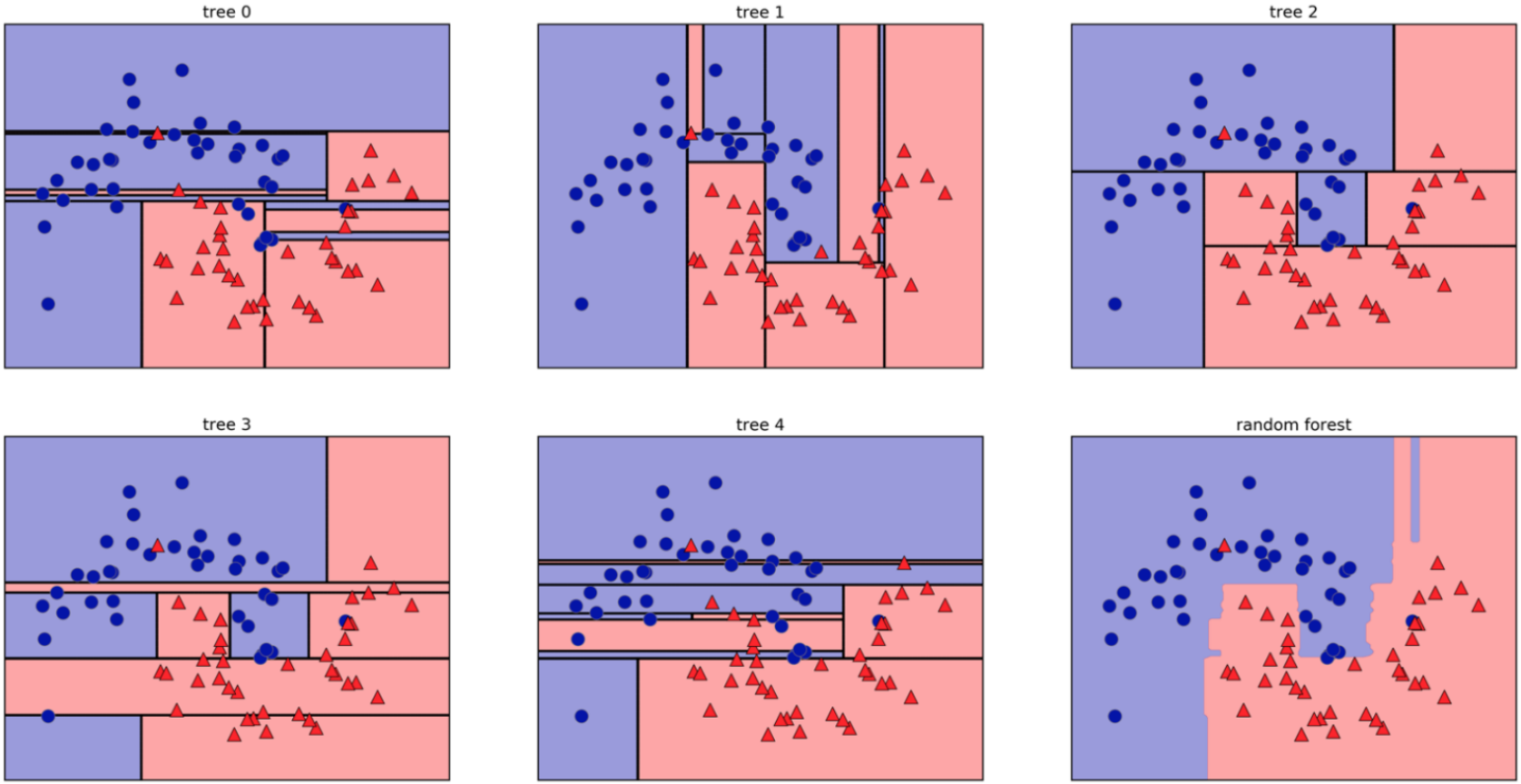




Bias and Variance

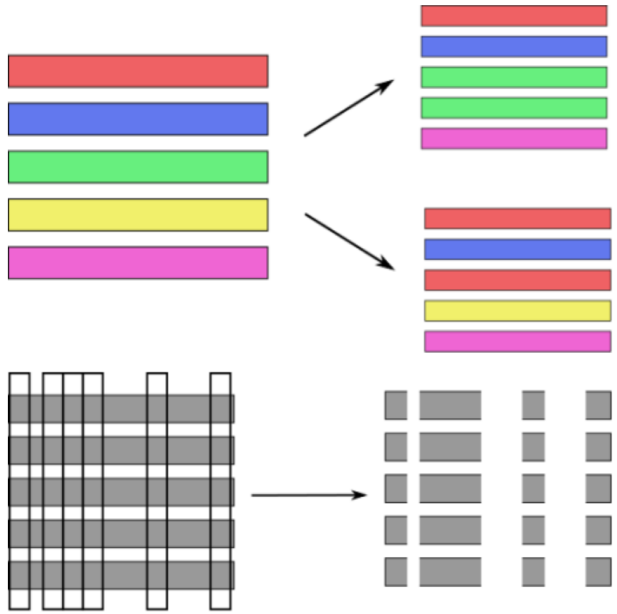


Random Forest



Randomize in Two Ways

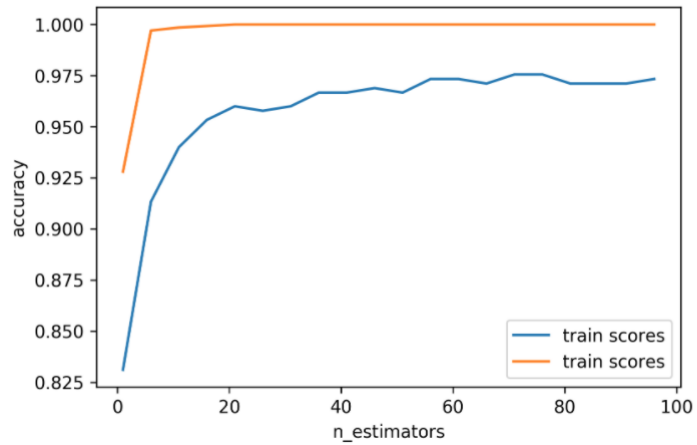
- For each tree: Pick bootstrap sample of data
- For each split: Pick random sample of features
- More trees are always better



A Simple Example for Random Forest

```
train_scores = []
test_scores = []

rf = RandomForestClassifier(warm_start=True)
estimator_range = range(1, 100, 5)
for n_estimators in estimator_range:
    rf.n_estimators = n_estimators
    rf.fit(X_train, y_train)
    train_scores.append(rf.score(X_train, y_train))
    test_scores.append(rf.score(X_test, y_test))
```



Out of Bag Estimates

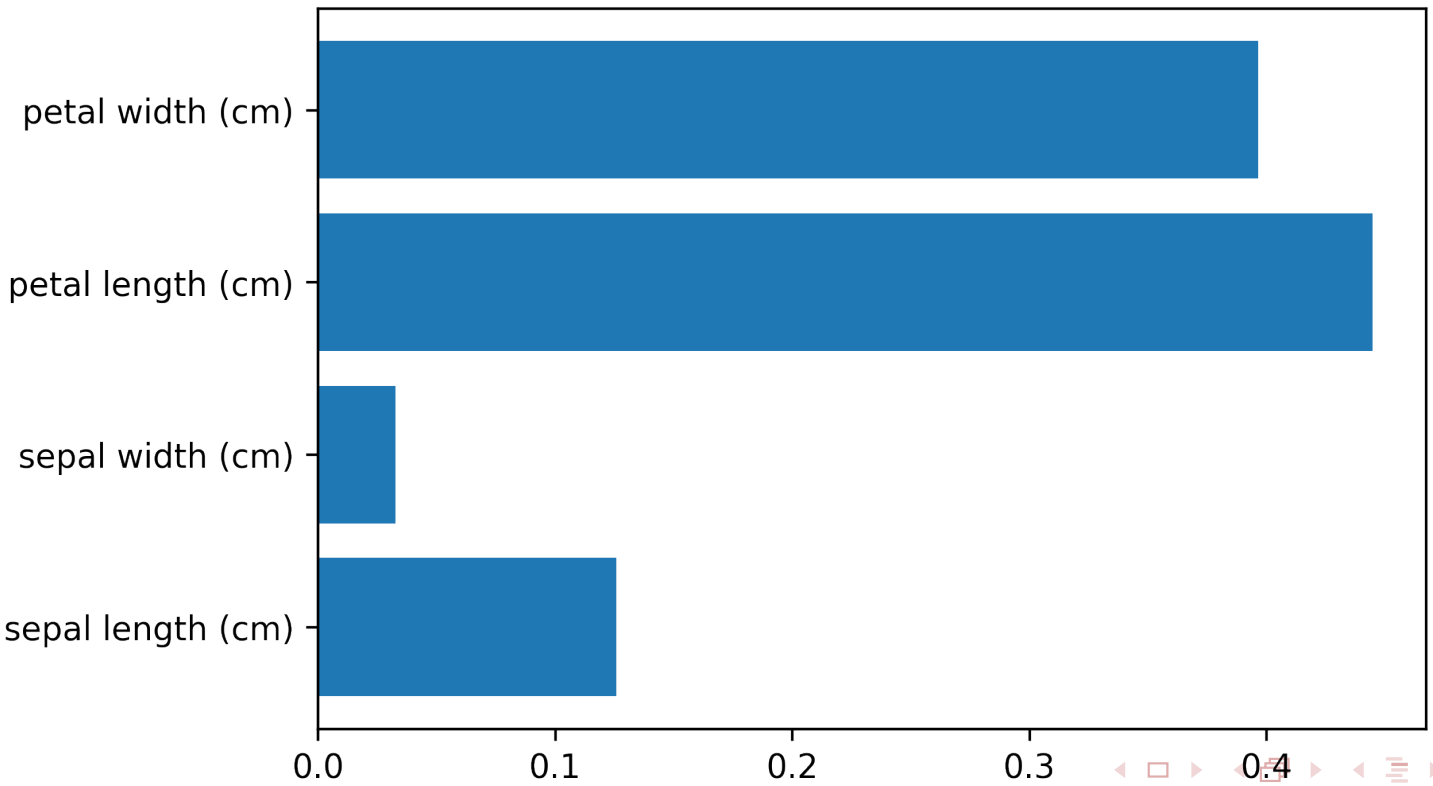
```
train_scores = []
test_scores = []
oob_scores = []
feature_range = range(1, 64, 5)
for max_features in feature_range:
    rf = RandomForestClassifier(max_features=max_features, oob_score=True,
                                n_estimators=200, random_state=0)
    rf.fit(X_train, y_train)
    train_scores.append(rf.score(X_train, y_train))
    test_scores.append(rf.score(X_test, y_test))
    oob_scores.append(rf.oob_score_)
```

- Each tree only uses 66% of data
- Can evaluate it on the rest!
- Make predictions for out-of-bag, average, score.
- Each prediction is an average over different subset of trees

Out of Bag Example

Variable Importance

```
1  X_train, X_test, y_train, y_test = train_test_split(
2      iris.data, iris.target, stratify=iris.target, random_state=1)
3  rf = RandomForestClassifier().fit(X_train, y_train)
4  rf.feature_importances_
5  plt.barh(range(4), rf.feature_importances_)
6  plt.yticks(range(4), iris.feature_names);
```



Summary

- Decision trees are robust to data scaling, but are very unstable models
- To cope with the high variance in the decision trees, we use Random Forest models