Trees & Forests

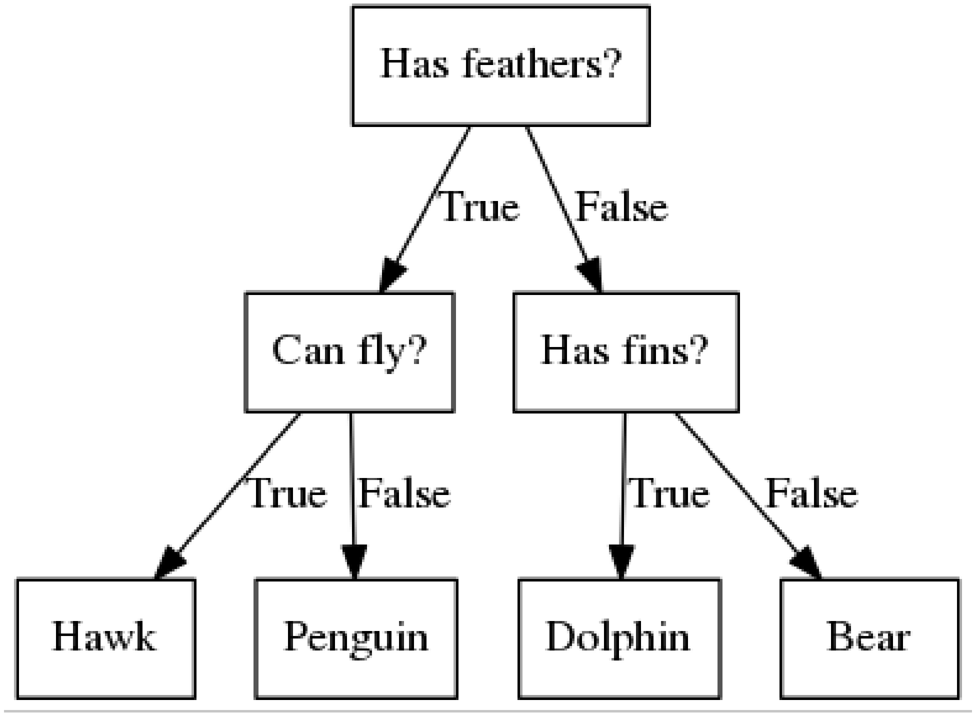
Andreas Müller

Why trees?

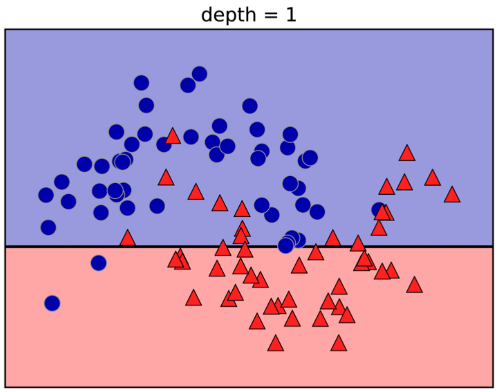
- Very powerful modeling method – non-linear!
- Doesn’t care about scaling of distribution of data!
- “Interpretable”
- Basis of very powerful models!

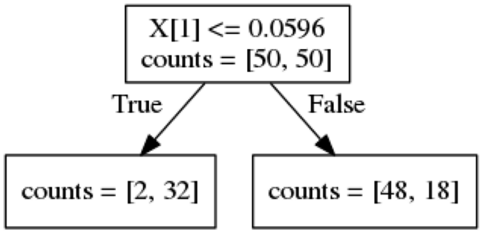
Decision Trees for Classification

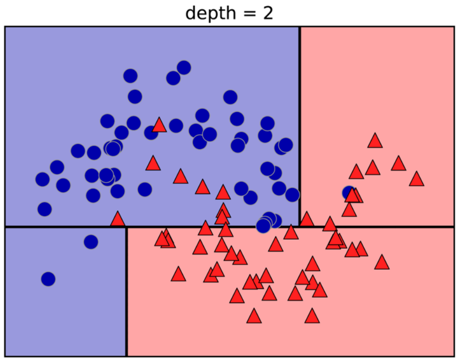
Idea: series of binary questions

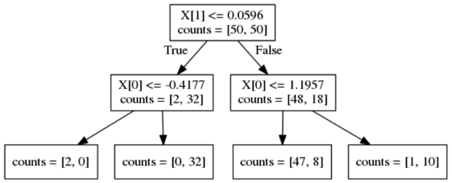


Building trees


depth = 1




depth = 2

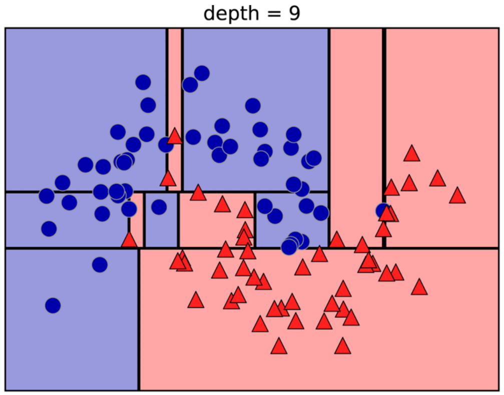


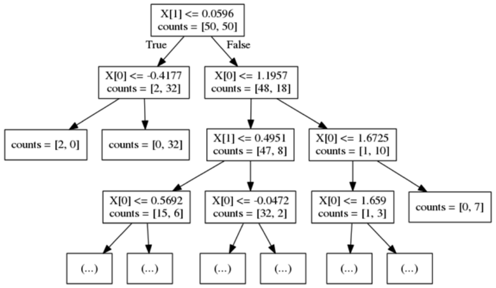
Continuous features:
“questions” are thresholds on single features.

[Other methods are possible but not as common]

For each split:
exhaustive search over all features and thresholds!

Minimize “impurity”


depth = 9



Criteria (for classification)

- Gini index:

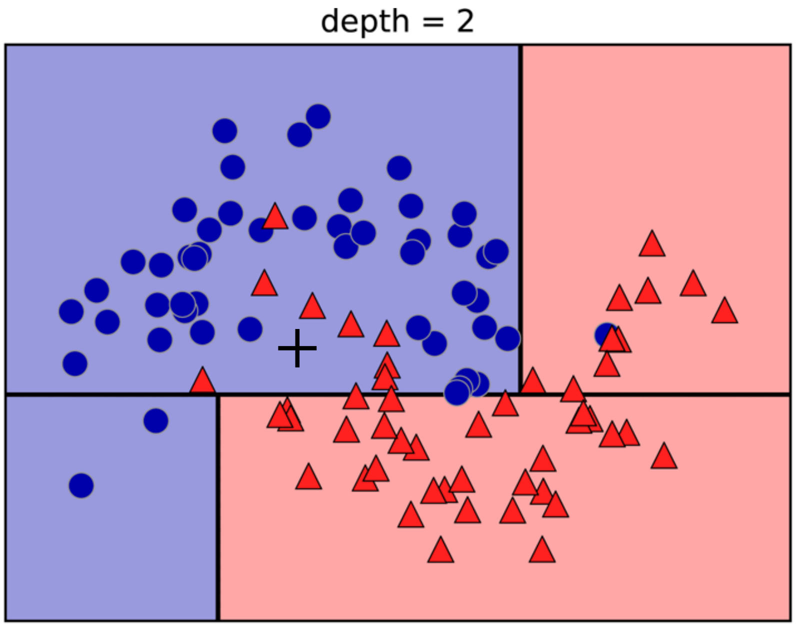
$$H_{\mathrm{gini}}(X_m) = \sum_{k \in \mathcal{Y}} p_{mk}(1 - p_{mk})$$

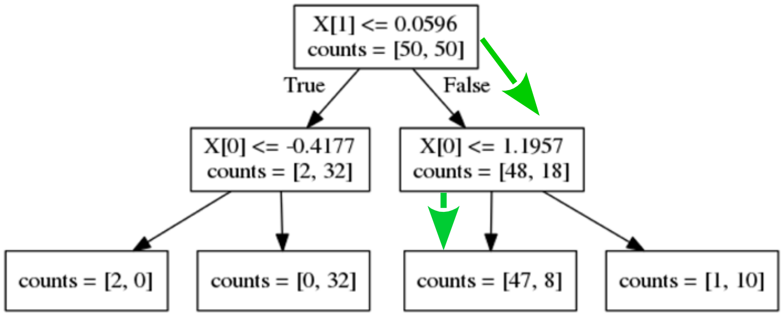
- Cross-entropy:

$$H_{\mathrm{CE}}(X_m) = -\sum_{k \in \mathcal{Y}} p_{mk} \log(p_{mk})$$

X_m	observations in node m
$\mathcal{Y}$	classes
$p_{m\cdot}$	distribution over classes in node m

Prediction




X[1] <= 0.0596
counts = [50, 50]

True

False

X[0] <= -0.4177
counts = [2, 32]

X[0] <= 1.1957
counts = [48, 18]

counts = [2, 0]

counts = [0, 32]

counts = [47, 8]

counts = [1, 10]

y=0

Traverse tree based on feature tests
Predict most common class in leaf

Regression trees

- Impurity Criteria:

  Mean Squared Error

  Mean Absolute Error

- Prediction:

  Predict mean.

- Without regularization / pruning:

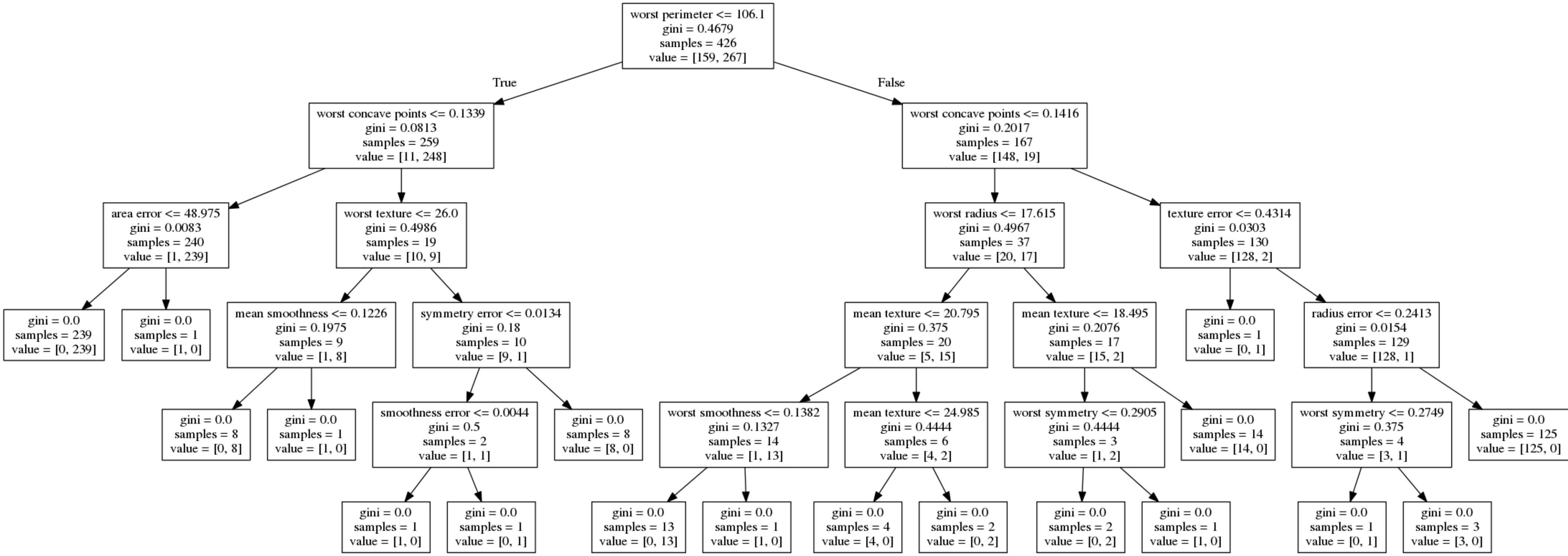
  Each leaf often contains a single point to be “pure”

Parameter tuning

- Pre-pruning and post-pruning (not in sklearn yet)
- Limit tree size (pick one):

  max_depth
  max_leaf_nodes
  min_samples_split
  (and more)

No pruning



worst perimeter <= 106.1
gini = 0.4679
samples = 426
value = [159, 267]

True

False

worst concave points <= 0.1339
gini = 0.0813
samples = 259
value = [11, 248]

worst concave points <= 0.1416
gini = 0.2017
samples = 167
value = [148, 19]

area error <= 48.975
gini = 0.0083
samples = 240
value = [1, 239]

worst texture <= 26.0
gini = 0.4986
samples = 19
value = [10, 9]

worst radius <= 17.615
gini = 0.4967
samples = 37
value = [20, 17]

texture error <= 0.4314
gini = 0.0303
samples = 130
value = [128, 2]

gini = 0.0
samples = 239
value = [0, 239]

gini = 0.0
samples = 1
value = [1, 0]

mean smoothness <= 0.1226
gini = 0.1975
samples = 9
value = [1, 8]

symmetry error <= 0.0134
gini = 0.18
samples = 10
value = [9, 1]

mean texture <= 20.795
gini = 0.375
samples = 20
value = [5, 15]

mean texture <= 18.495
gini = 0.2076
samples = 17
value = [15, 2]

gini = 0.0
samples = 1
value = [0, 1]

radius error <= 0.2413
gini = 0.0154
samples = 129
value = [128, 1]

gini = 0.0
samples = 8
value = [0, 8]

gini = 0.0
samples = 1
value = [1, 0]

smoothness error <= 0.0044
gini = 0.5
samples = 2
value = [1, 1]

gini = 0.0
samples = 8
value = [8, 0]

worst smoothness <= 0.1382
gini = 0.1327
samples = 14
value = [1, 13]

mean texture <= 24.985
gini = 0.4444
samples = 6
value = [4, 2]

worst symmetry <= 0.2905
gini = 0.4444
samples = 3
value = [1, 2]

gini = 0.0
samples = 14
value = [14, 0]

worst symmetry <= 0.2749
gini = 0.375
samples = 4
value = [3, 1]

gini = 0.0
samples = 125
value = [125, 0]

gini = 0.0
samples = 1
value = [1, 0]

gini = 0.0
samples = 1
value = [0, 1]

gini = 0.0
samples = 13
value = [0, 13]

gini = 0.0
samples = 1
value = [1, 0]

gini = 0.0
samples = 4
value = [4, 0]

gini = 0.0
samples = 2
value = [0, 2]

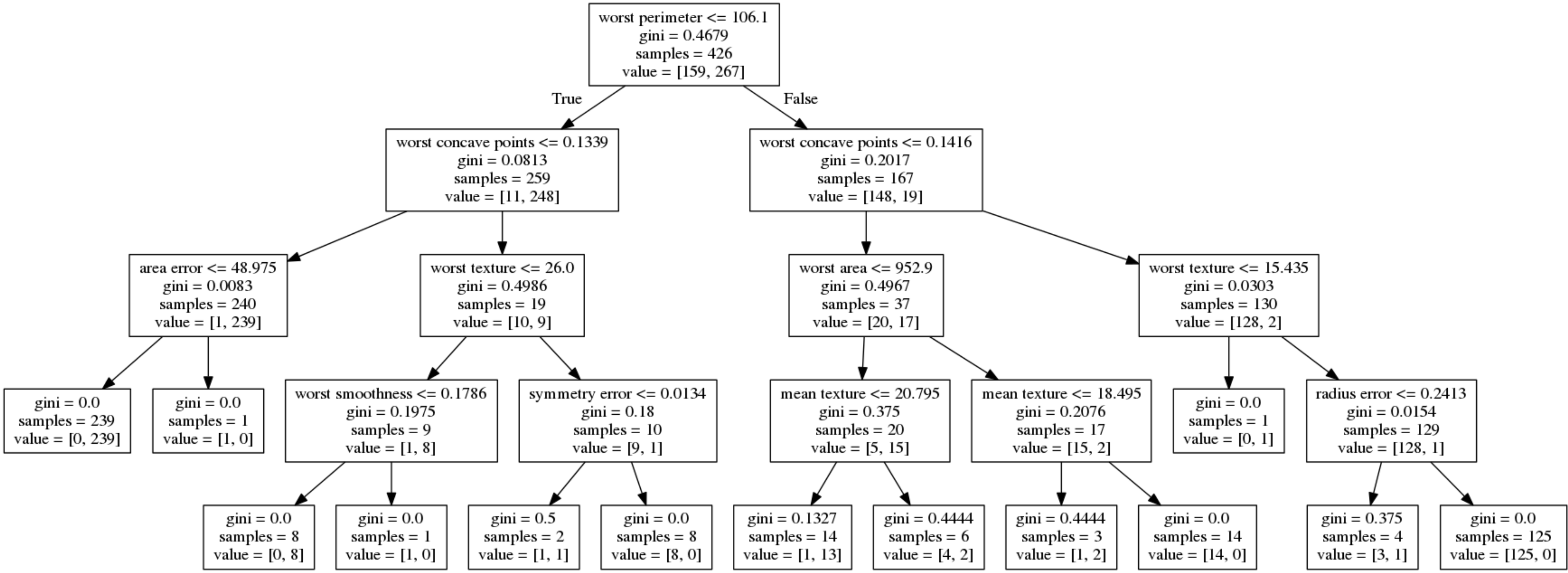
gini = 0.0
samples = 2
value = [0, 2]

gini = 0.0
samples = 1
value = [1, 0]

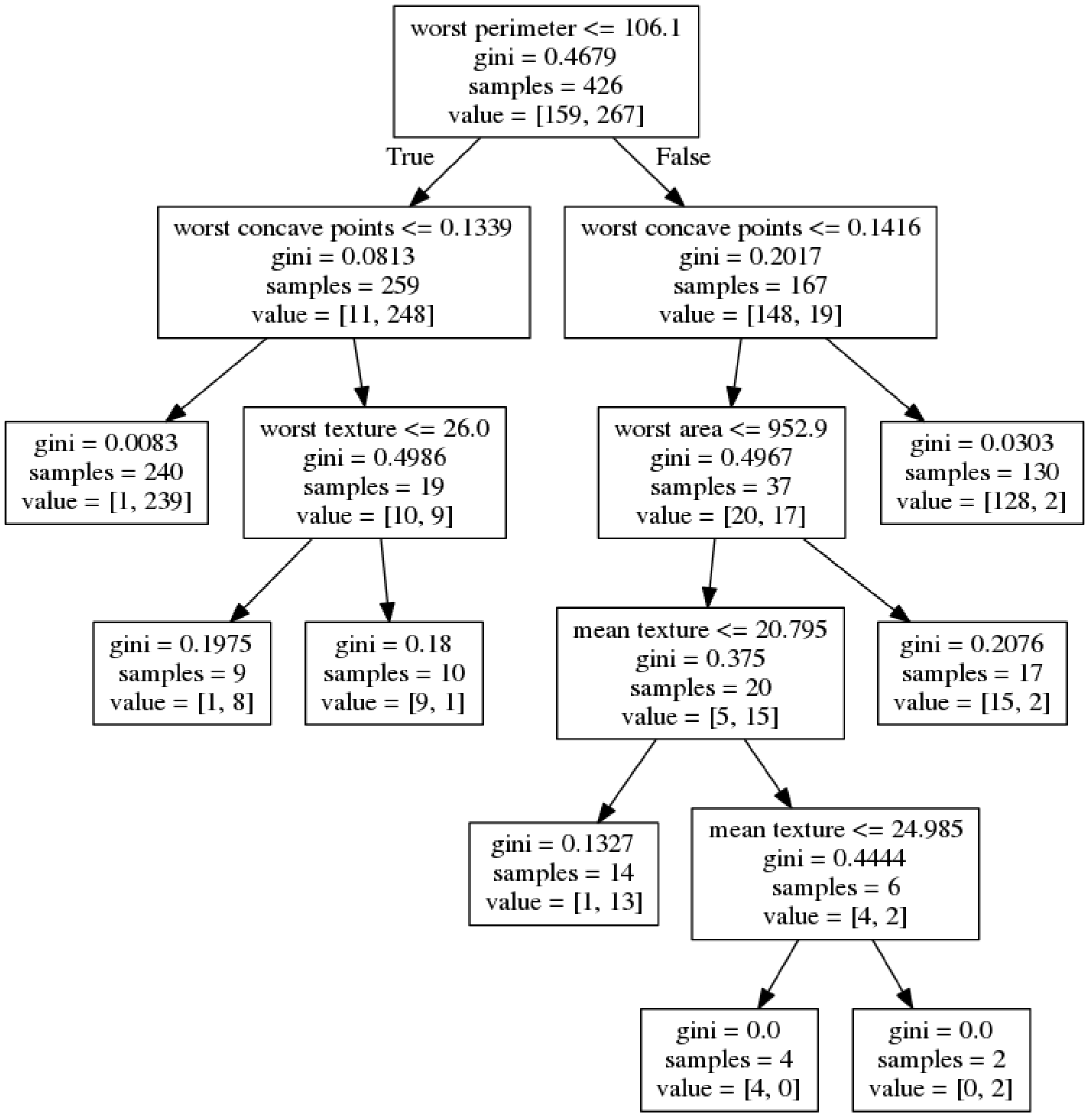
gini = 0.0
samples = 1
value = [0, 1]

gini = 0.0
samples = 3
value = [3, 0]

max_depth=4


worst perimeter <= 106.1
gini = 0.4679
samples = 426
value = [159, 267]
True
False
worst concave points <= 0.1339
gini = 0.0813
samples = 259
value = [11, 248]
worst concave points <= 0.1416
gini = 0.2017
samples = 167
value = [148, 19]
area error <= 48.975
gini = 0.0083
samples = 240
value = [1, 239]
worst texture <= 26.0
gini = 0.4986
samples = 19
value = [10, 9]
worst area <= 952.9
gini = 0.4967
samples = 37
value = [20, 17]
worst texture <= 15.435
gini = 0.0303
samples = 130
value = [128, 2]
gini = 0.0
samples = 239
value = [0, 239]
gini = 0.0
samples = 1
value = [1, 0]
worst smoothness <= 0.1786
gini = 0.1975
samples = 9
value = [1, 8]
symmetry error <= 0.0134
gini = 0.18
samples = 10
value = [9, 1]
mean texture <= 20.795
gini = 0.375
samples = 20
value = [5, 15]
mean texture <= 18.495
gini = 0.2076
samples = 17
value = [15, 2]
gini = 0.0
samples = 1
value = [0, 1]
radius error <= 0.2413
gini = 0.0154
samples = 129
value = [128, 1]
gini = 0.0
samples = 8
value = [0, 8]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 8
value = [8, 0]
gini = 0.1327
samples = 14
value = [1, 13]
gini = 0.4444
samples = 6
value = [4, 2]
gini = 0.4444
samples = 3
value = [1, 2]
gini = 0.0
samples = 14
value = [14, 0]
gini = 0.375
samples = 4
value = [3, 1]
gini = 0.0
samples = 125
value = [125, 0]

max_leaf_nodes=8



worst perimeter <= 106.1
gini = 0.4679
samples = 426
value = [159, 267]

True

False

worst concave points <= 0.1339
gini = 0.0813
samples = 259
value = [11, 248]

worst concave points <= 0.1416
gini = 0.2017
samples = 167
value = [148, 19]

gini = 0.0083
samples = 240
value = [1, 239]

worst texture <= 26.0
gini = 0.4986
samples = 19
value = [10, 9]

worst area <= 952.9
gini = 0.4967
samples = 37
value = [20, 17]

gini = 0.0303
samples = 130
value = [128, 2]

gini = 0.1975
samples = 9
value = [1, 8]

gini = 0.18
samples = 10
value = [9, 1]

mean texture <= 20.795
gini = 0.375
samples = 20
value = [5, 15]

gini = 0.2076
samples = 17
value = [15, 2]

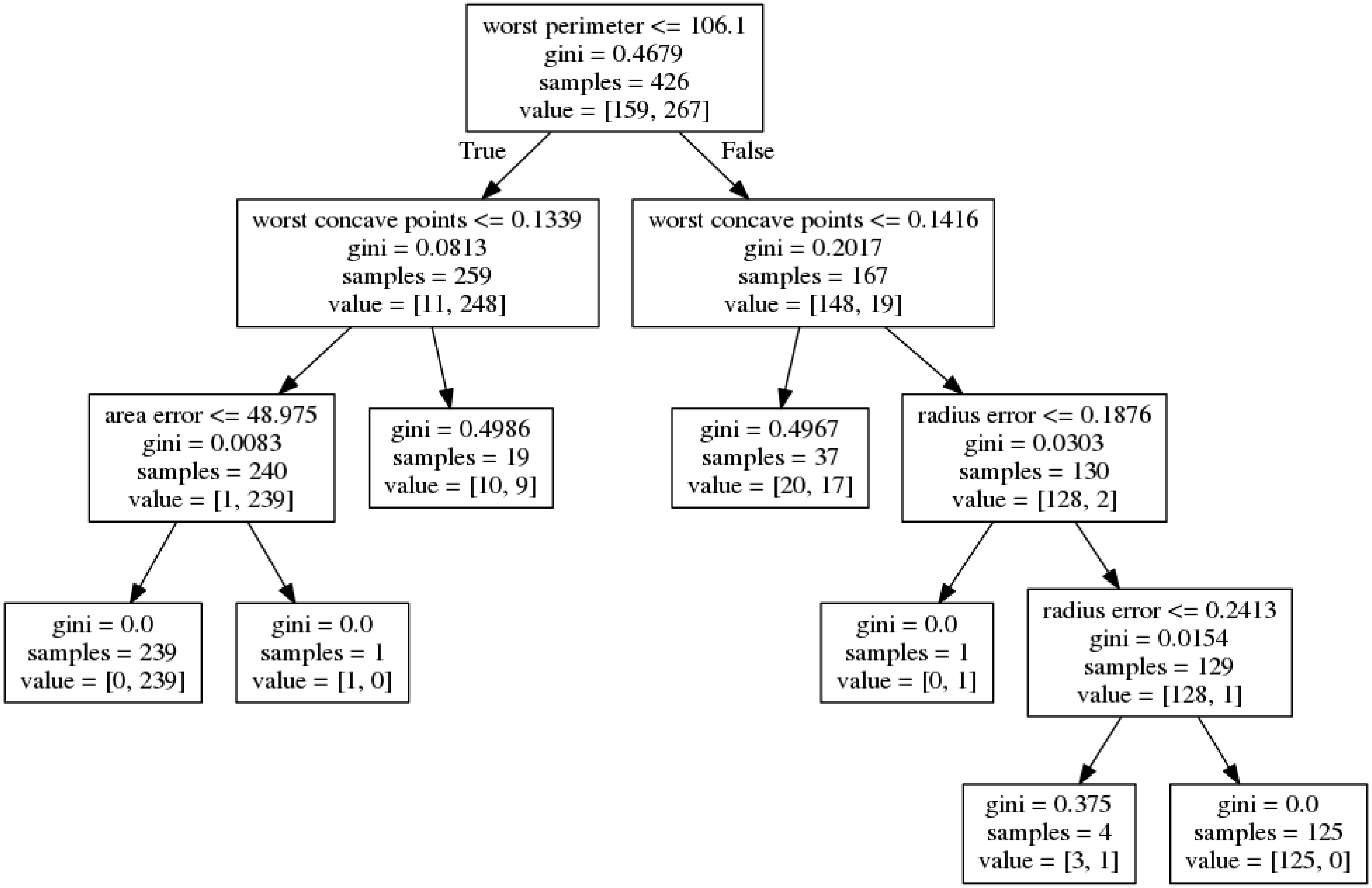
gini = 0.1327
samples = 14
value = [1, 13]

mean texture <= 24.985
gini = 0.4444
samples = 6
value = [4, 2]

gini = 0.0
samples = 4
value = [4, 0]

gini = 0.0
samples = 2
value = [0, 2]

min_samples_split=50



worst perimeter <= 106.1
gini = 0.4679
samples = 426
value = [159, 267]

True

False

worst concave points <= 0.1339
gini = 0.0813
samples = 259
value = [11, 248]

worst concave points <= 0.1416
gini = 0.2017
samples = 167
value = [148, 19]

area error <= 48.975
gini = 0.0083
samples = 240
value = [1, 239]

gini = 0.4986
samples = 19
value = [10, 9]

gini = 0.4967
samples = 37
value = [20, 17]

radius error <= 0.1876
gini = 0.0303
samples = 130
value = [128, 2]

gini = 0.0
samples = 239
value = [0, 239]

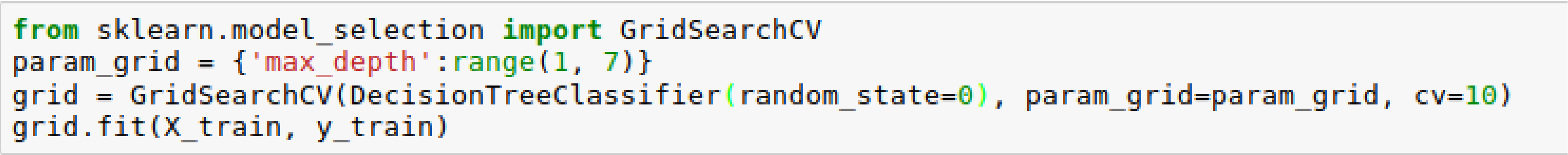
gini = 0.0
samples = 1
value = [1, 0]

gini = 0.0
samples = 1
value = [0, 1]

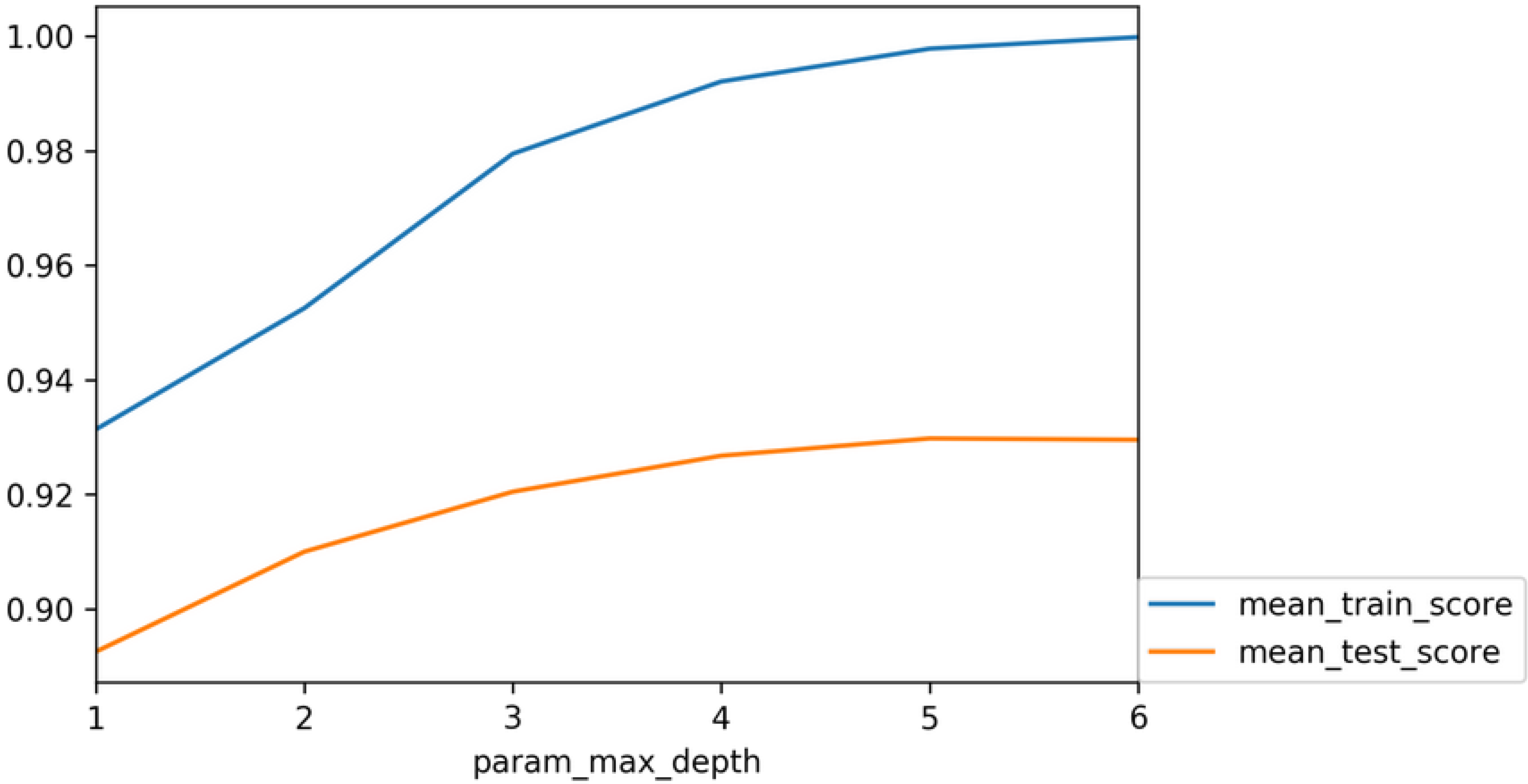
radius error <= 0.2413
gini = 0.0154
samples = 129
value = [128, 1]

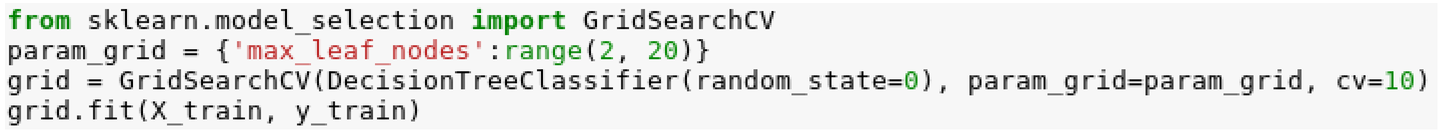
gini = 0.375
samples = 4
value = [3, 1]

gini = 0.0
samples = 125
value = [125, 0]

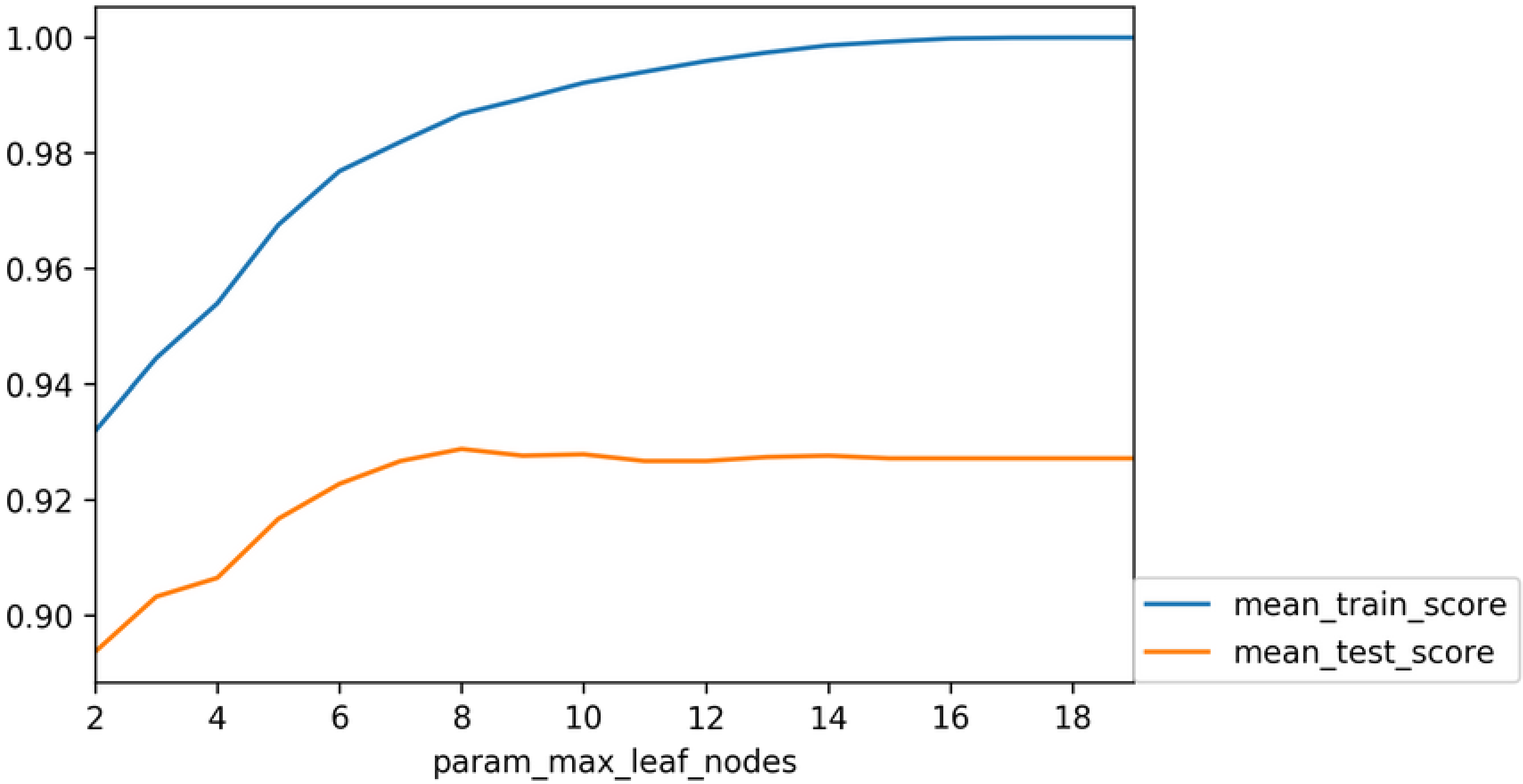


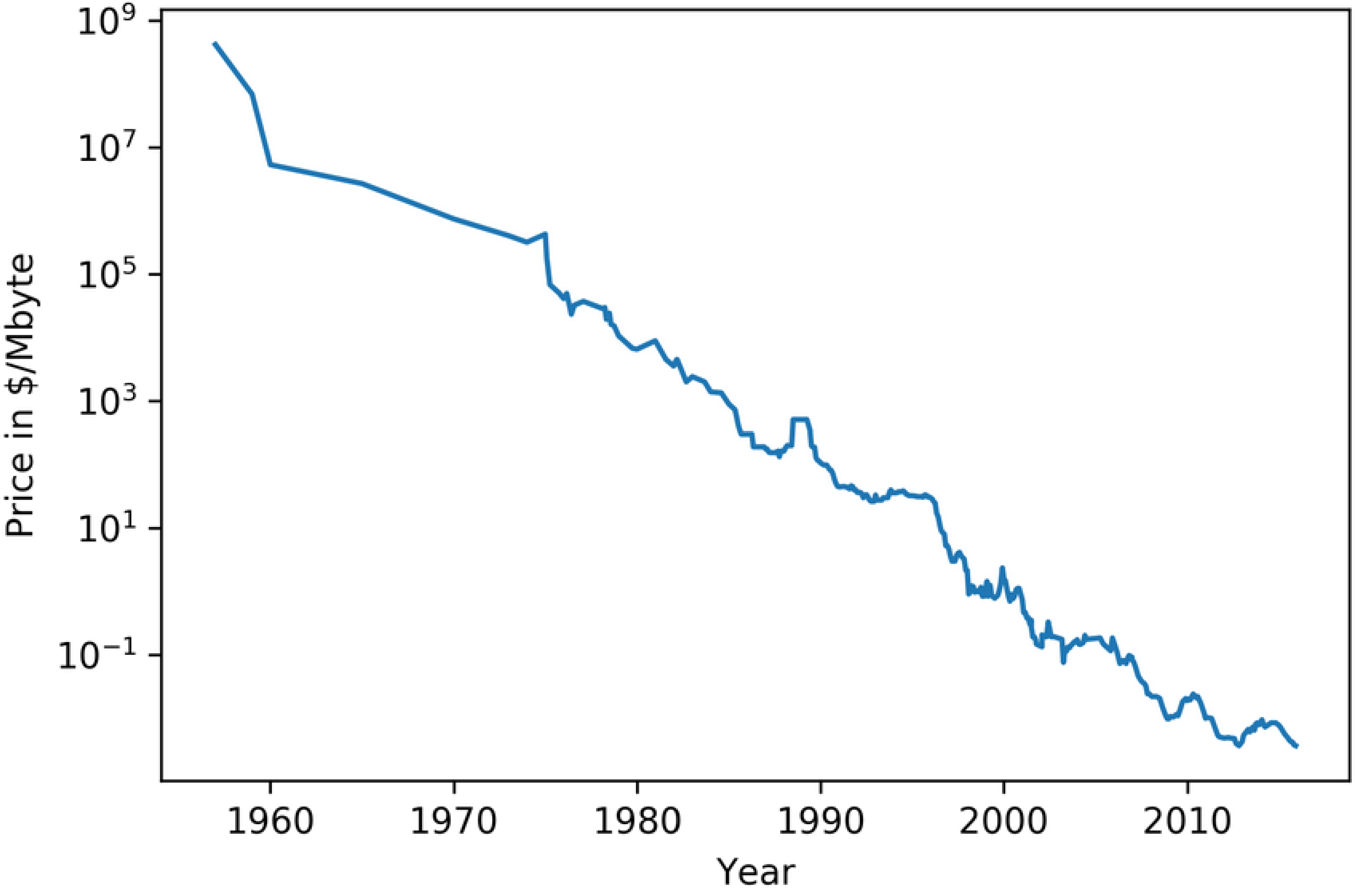
```
from sklearn.model_selection import GridSearchCV
param_grid = {'max_depth':range(1, 7)}
grid = GridSearchCV(DecisionTreeClassifier(random_state=0), param_grid=param_grid, cv=10)
grid.fit(X_train, y_train)
```

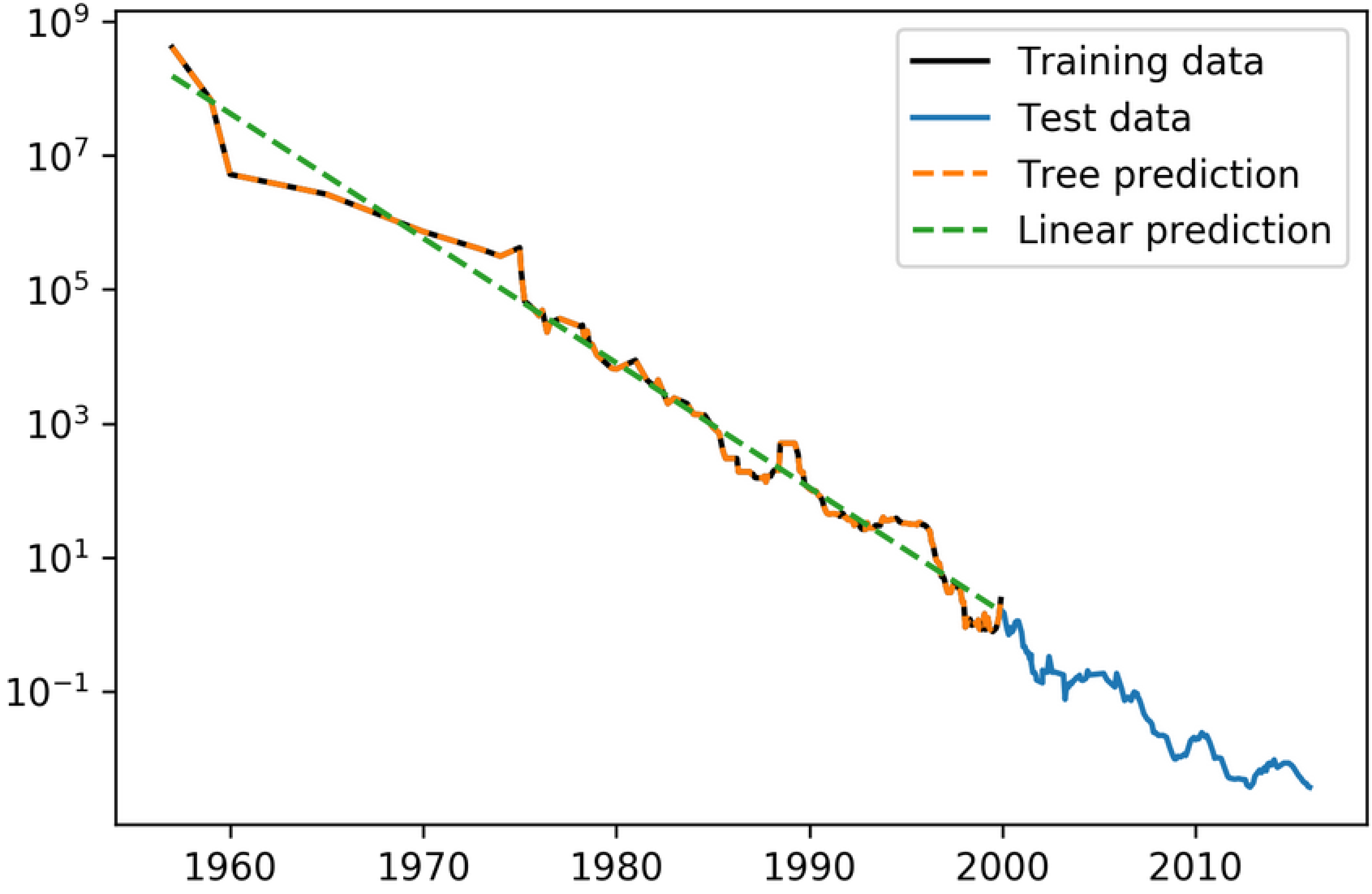




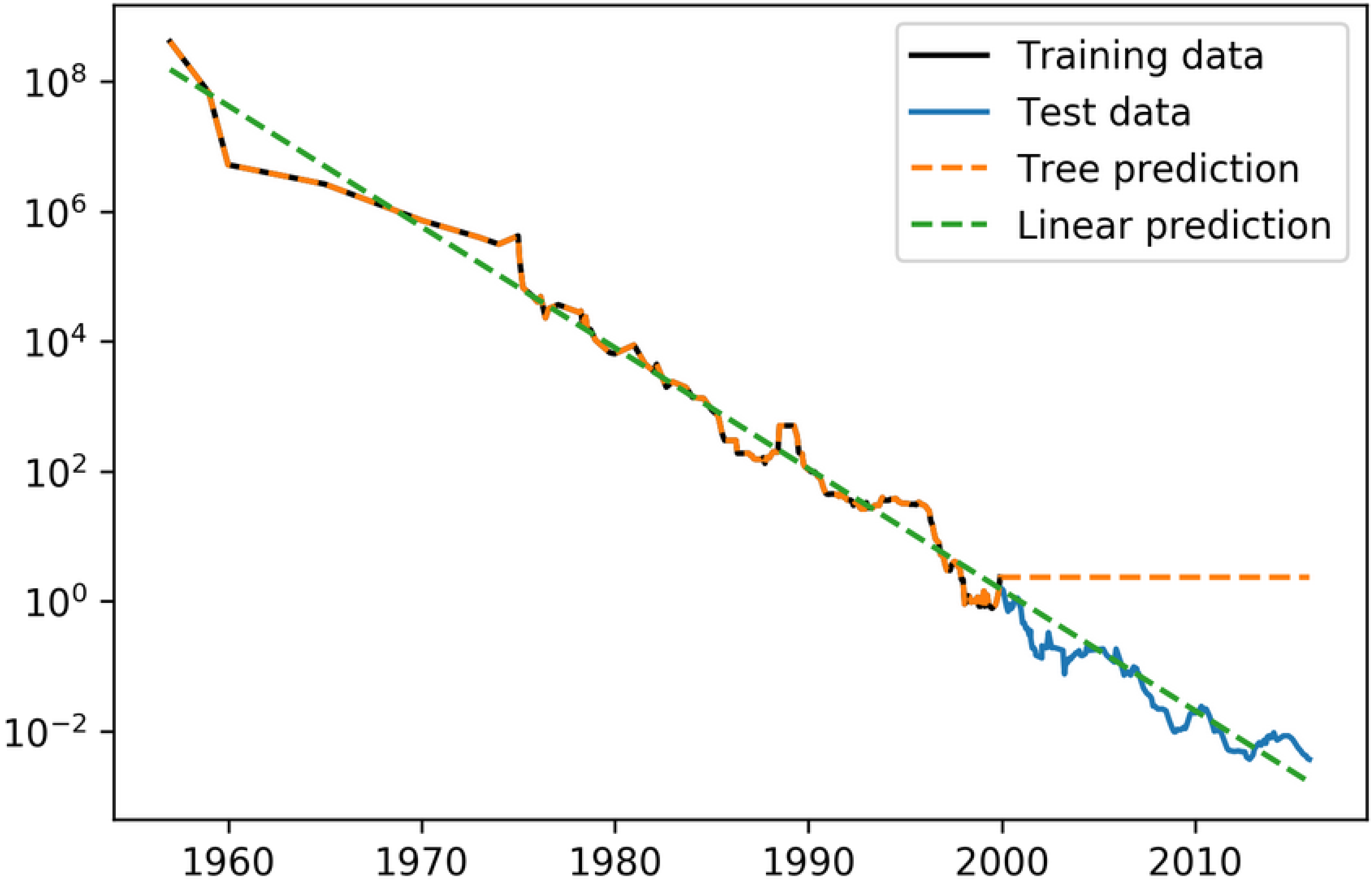
```python
from sklearn.model_selection import GridSearchCV
param_grid = {'max_leaf_nodes':range(2, 20)}
grid = GridSearchCV(DecisionTreeClassifier(random_state=0), param_grid=param_grid, cv=10)
grid.fit(X_train, y_train)
```




10^9
10^7
10^5
10^3
10^1
10^{-1}
Price in $/Mbyte
1960
1970
1980
1990
2000
2010
Year

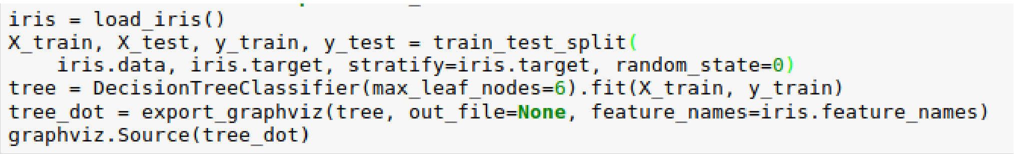

Training data
Test data
Tree prediction
Linear prediction
10^9
10^7
10^5
10^3
10^1
10^{-1}
1960
1970
1980
1990
2000
2010

Extrapolation

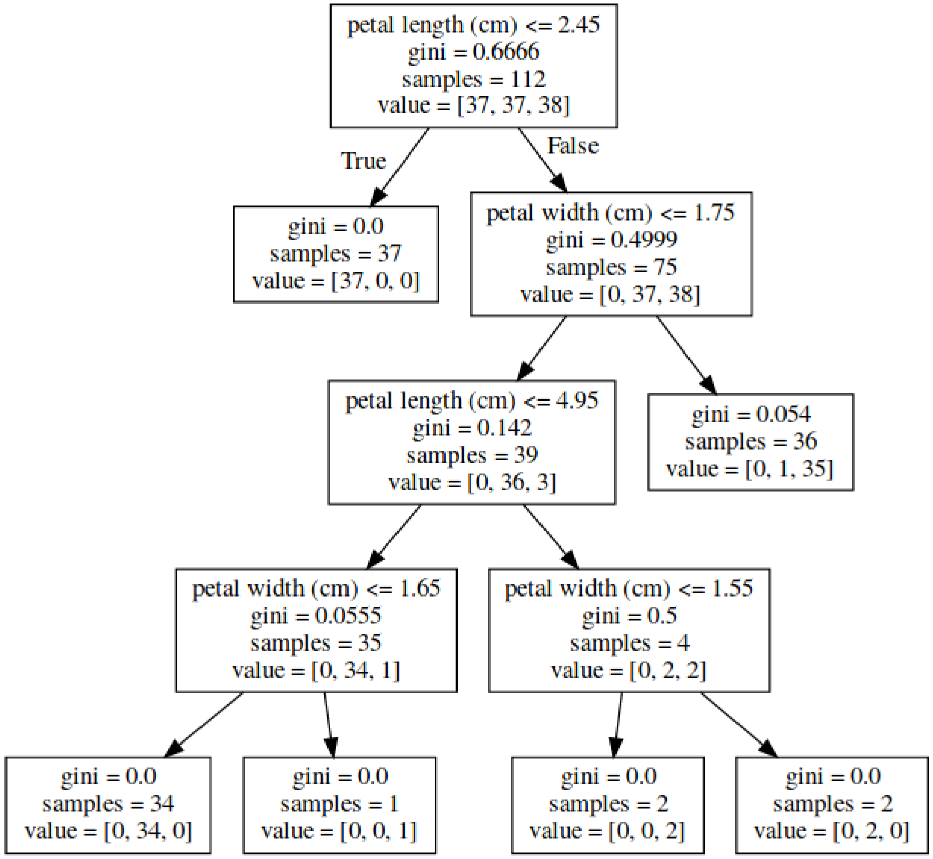


Would be the same for nearest neighbor regression!

Instability



```
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, stratify=iris.target, random_state=0)
tree = DecisionTreeClassifier(max_leaf_nodes=6).fit(X_train, y_train)
tree_dot = export_graphviz(tree, out_file=None, feature_names=iris.feature_names)
graphviz.Source(tree_dot)
```



petal length (cm) <= 2.45
gini = 0.6666
samples = 112
value = [37, 37, 38]

True

False

gini = 0.0
samples = 37
value = [37, 0, 0]

petal width (cm) <= 1.75
gini = 0.4999
samples = 75
value = [0, 37, 38]

petal length (cm) <= 4.95
gini = 0.142
samples = 39
value = [0, 36, 3]

gini = 0.054
samples = 36
value = [0, 1, 35]

petal width (cm) <= 1.65
gini = 0.0555
samples = 35
value = [0, 34, 1]

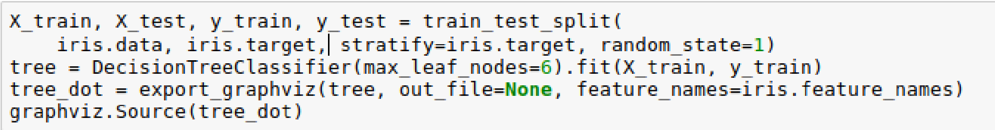
petal width (cm) <= 1.55
gini = 0.5
samples = 4
value = [0, 2, 2]

gini = 0.0
samples = 34
value = [0, 34, 0]

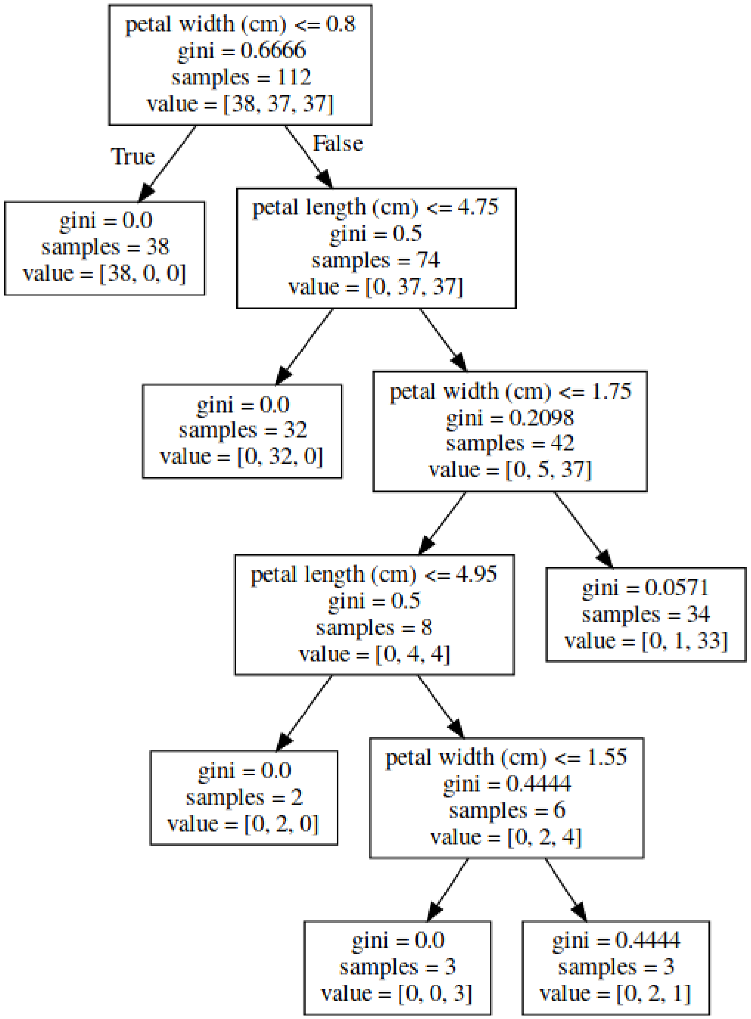
gini = 0.0
samples = 1
value = [0, 0, 1]

gini = 0.0
samples = 2
value = [0, 0, 2]

gini = 0.0
samples = 2
value = [0, 2, 0]



```
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, stratify=iris.target, random_state=1)
tree = DecisionTreeClassifier(max_leaf_nodes=6).fit(X_train, y_train)
tree_dot = export_graphviz(tree, out_file=None, feature_names=iris.feature_names)
graphviz.Source(tree_dot)
```



petal width (cm) <= 0.8
gini = 0.6666
samples = 112
value = [38, 37, 37]

True

False

gini = 0.0
samples = 38
value = [38, 0, 0]

petal length (cm) <= 4.75
gini = 0.5
samples = 74
value = [0, 37, 37]

gini = 0.0
samples = 32
value = [0, 32, 0]

petal width (cm) <= 1.75
gini = 0.2098
samples = 42
value = [0, 5, 37]

petal length (cm) <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]

gini = 0.0571
samples = 34
value = [0, 1, 33]

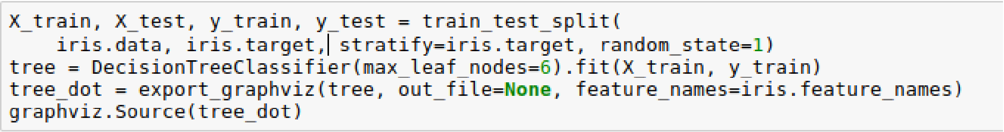
gini = 0.0
samples = 2
value = [0, 2, 0]

petal width (cm) <= 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]

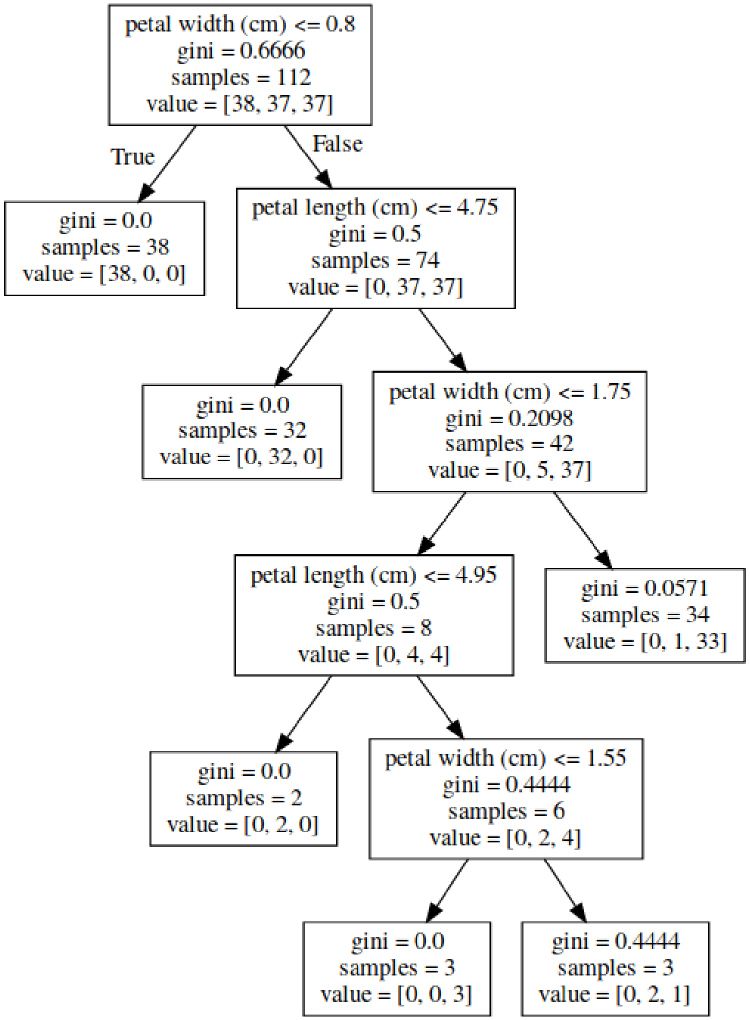
gini = 0.0
samples = 3
value = [0, 0, 3]

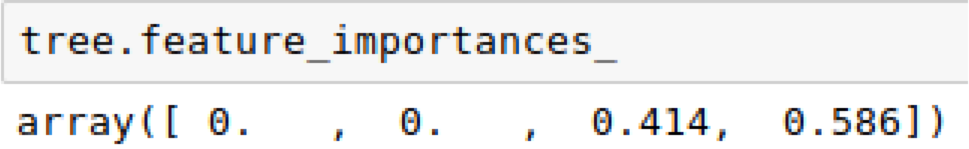
gini = 0.4444
samples = 3
value = [0, 2, 1]

Feature importance



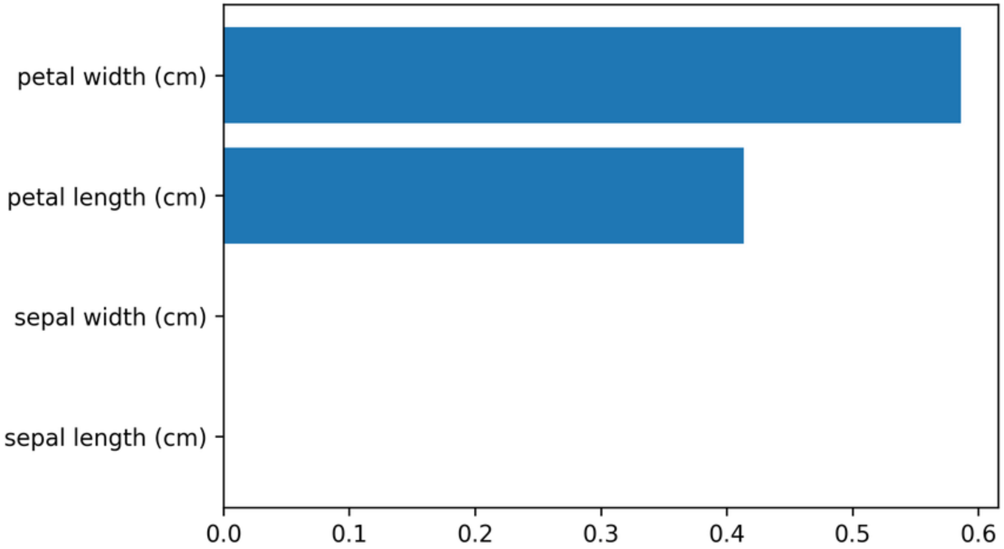
```
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, stratify=iris.target, random_state=1)
tree = DecisionTreeClassifier(max_leaf_nodes=6).fit(X_train, y_train)
tree_dot = export_graphviz(tree, out_file=None, feature_names=iris.feature_names)
graphviz.Source(tree_dot)
```





```
tree.feature_importances_
```

```
array([ 0.   ,  0.   ,  0.414,  0.586])
```



Unstable tree → unstable feature importances.

Might take one or multiple from a group of correlated features.

Categorical Data

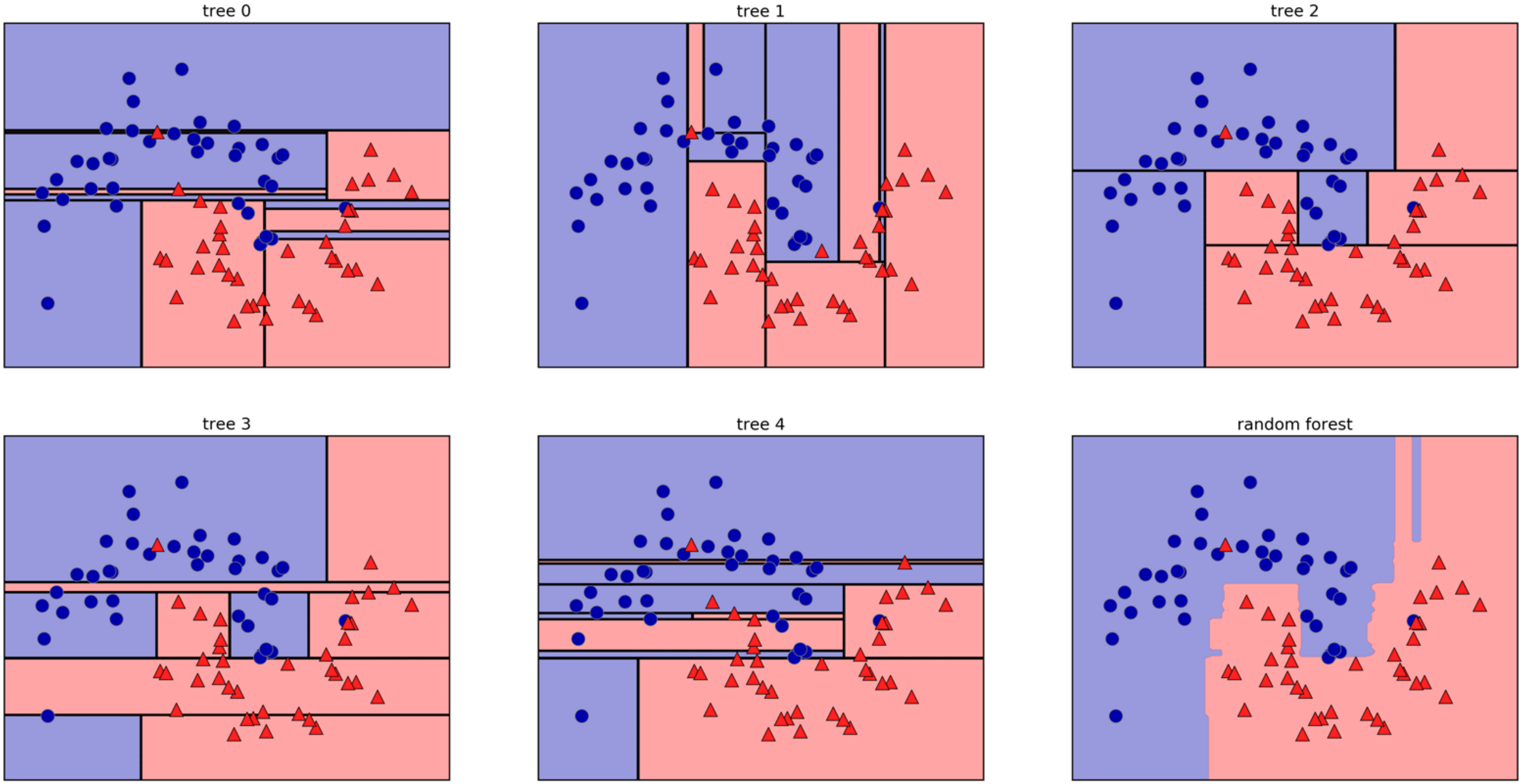
- Can split on categorical data directly
- Intuitive way to split: split in two subsets
- 2 ^ n_values many possibilities
- Possible to do in linear time exactly for gini index and binary classification.
- Heuristics done in practice for multi-class.
- Not in sklearn release version :(

Predicting probabilities

- Fraction of class in leaf.
- Without pruning: Always 100% certain!
- Even with pruning might be too certain.

Random Forests

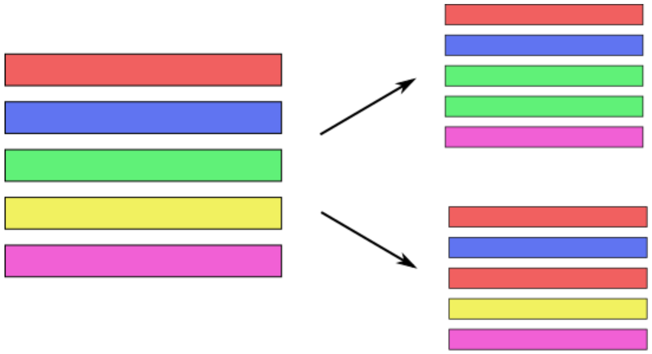
- Smarter bagging for trees!



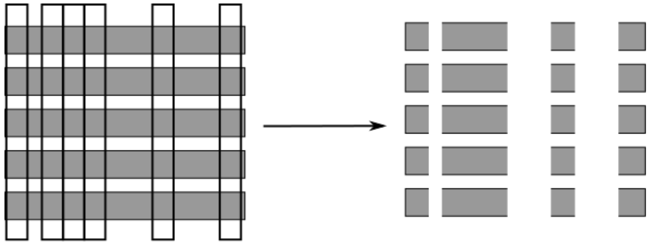
Randomize in two ways

- For each **tree**:

  Pick bootstrap sample of data



- For each **split**:
  Pick random sample of features



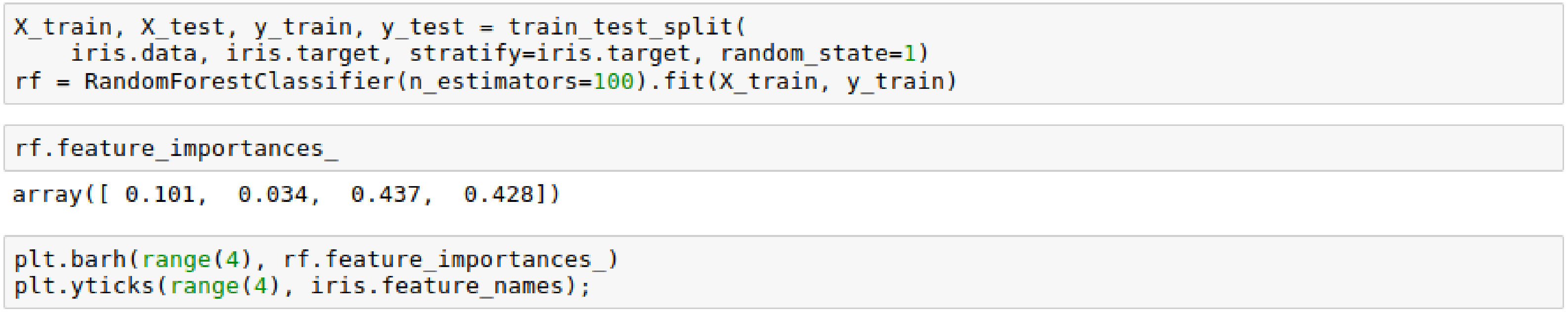
- More tree are always better

Tuning Random Forests

- Main parameter: max_features
  - around sqrt(n_features) for classification
  - Around n_features for regression

- n_estimators > 100
- Prepruning might help, definitely helps with model size!
- max_depth, max_leaf_nodes, min_samples_split again

Variable Importance

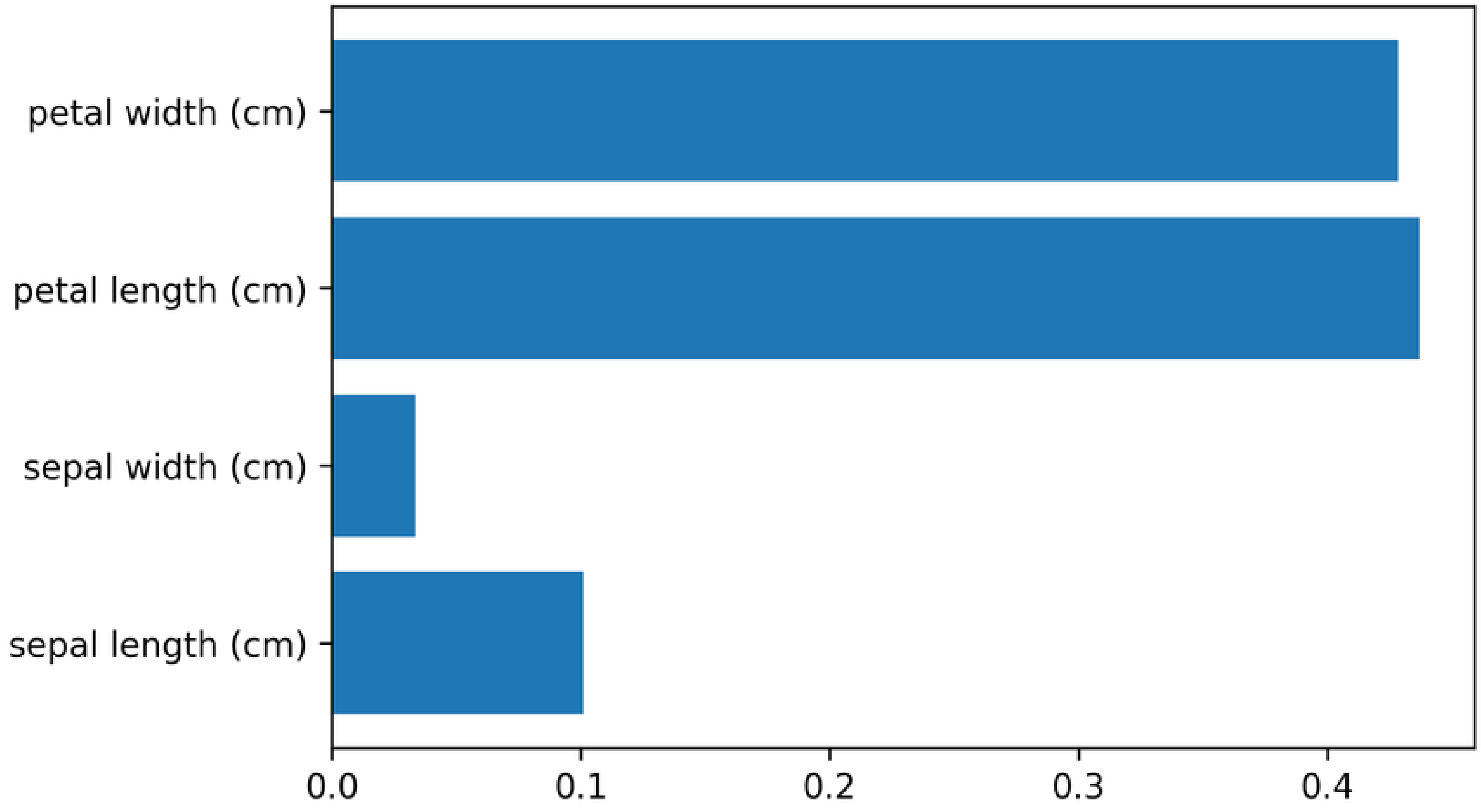


```
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, stratify=iris.target, random_state=1)
rf = RandomForestClassifier(n_estimators=100).fit(X_train, y_train)
```

```
rf.feature_importances_
```

```
array([ 0.101,  0.034,  0.437,  0.428])
```

```
plt.barh(range(4), rf.feature_importances_)
plt.yticks(range(4), iris.feature_names);
```


petal width (cm)
petal length (cm)
sepal width (cm)
sepal length (cm)