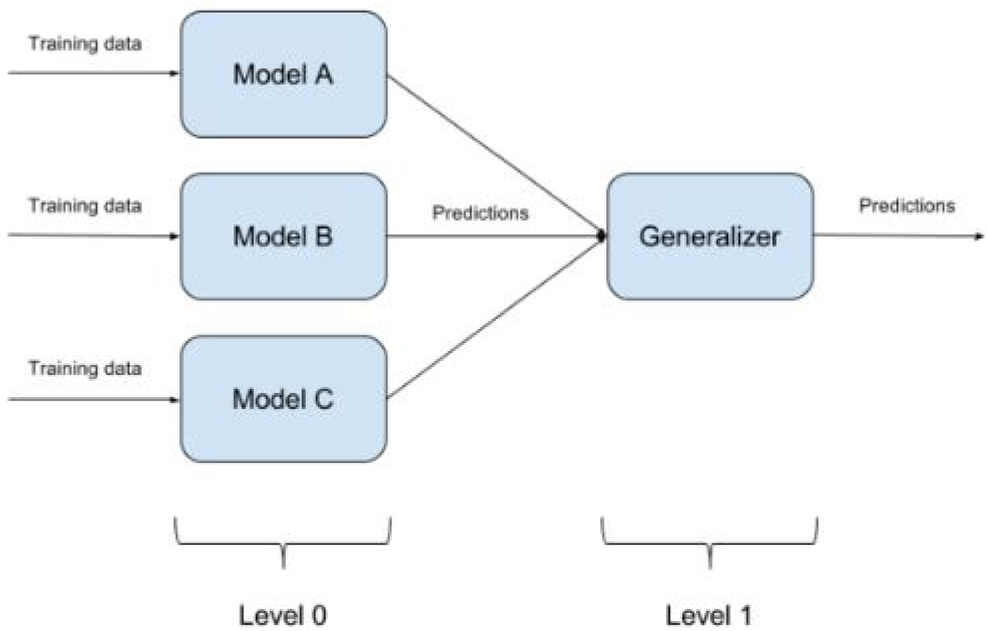
ENSEMBLE

CLASSIFIERS

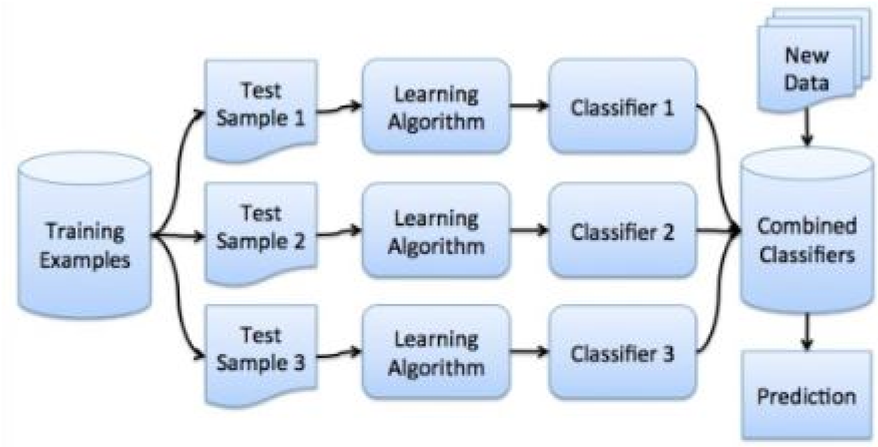
Ensemble Methods



- generate (”base learners”)
- combine learners
- goal: increase accuracy

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Sequential Ensemble

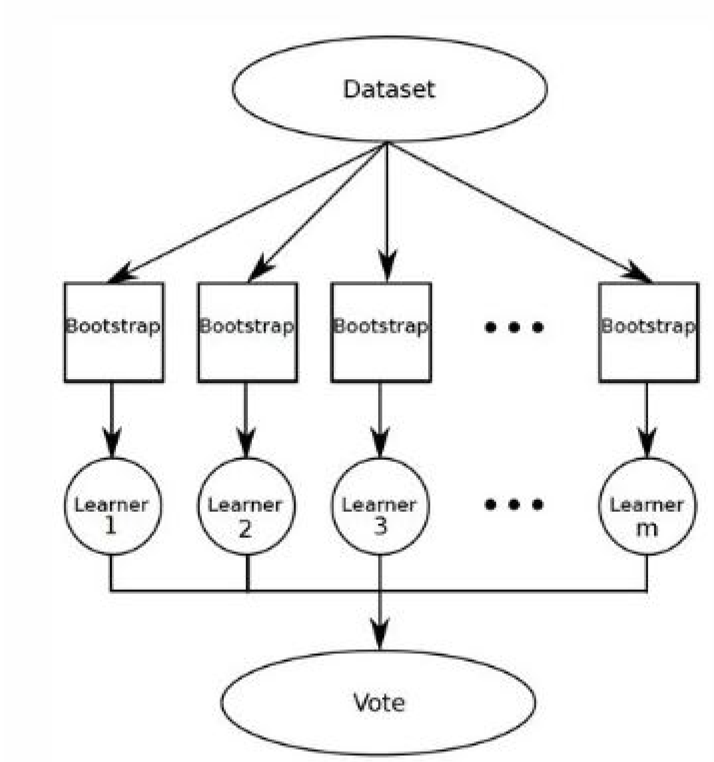


- learners are generated sequentially (Adaboost)

---

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Parallel Ensemble



- learners are generated in parallel (random forest)

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Random Forests vs. Decision Trees

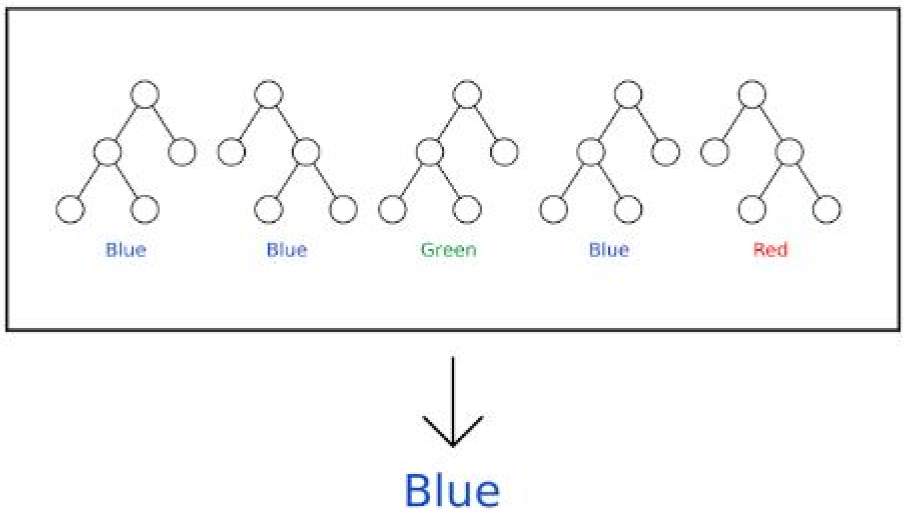
Decision Trees:

- inputs and outputs
- want classification for labels
- decision tree classifier

Random Forests

- build partial decision (sub)trees
- decide by ensemble of (sub)trees
- conceptually "similar" to kNN

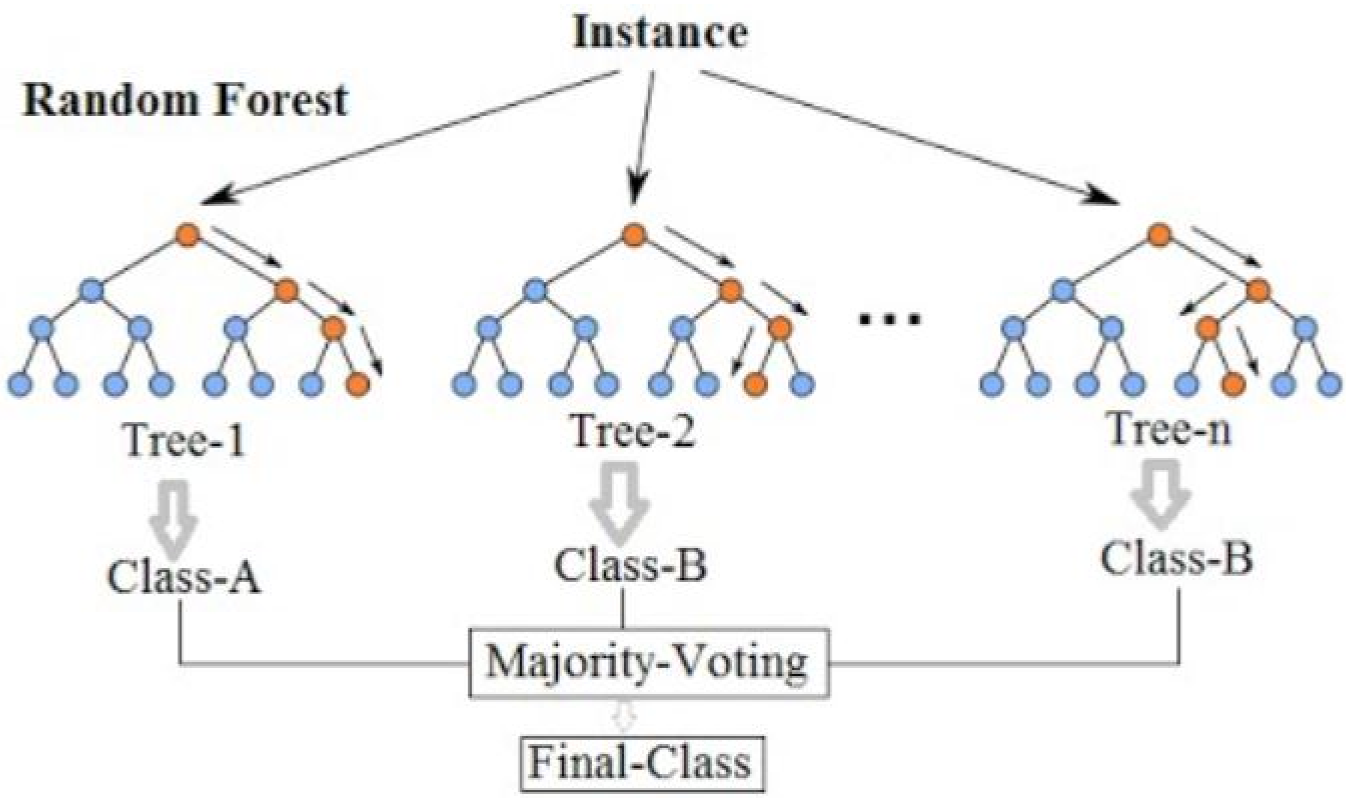
Random Forest (RF)



- a random sampling of data is used for each tree
- random subsets of features in splitting

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Training & Predictions



- randomly sample subsets ("bagging")
- fit models and predict

figure reprinted from www.kdnuggets.com with explicit permission of the editor

Bagging vs. Boosting

- *bagging:*

(a) samples with replacements
(b) equal weights of models
(c) majority voting
(d) ex: Random Forest

- *boosting:*

(a) new models from older ones
(b) weighted voting
(c) ex: AdaBoost and LogBoost

Advantages/Disadvantages

- advantages:

(a) lower variance
(b) better predictions

- disadvantages:

(a) do not train well on small datasets
(b) results hard to interpret
(c) computationally expensive

Hyper-parameters for Ensemble Classifiers

- performance depends on hyper-parameters:

(a) n_estimators: number of "weak" learners to use

(b) max_features: maximum number of features for each learner

(c) additional hyper-parameters on learners (e.g. max_depth of trees in random forest)

A Numerical Dataset

object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

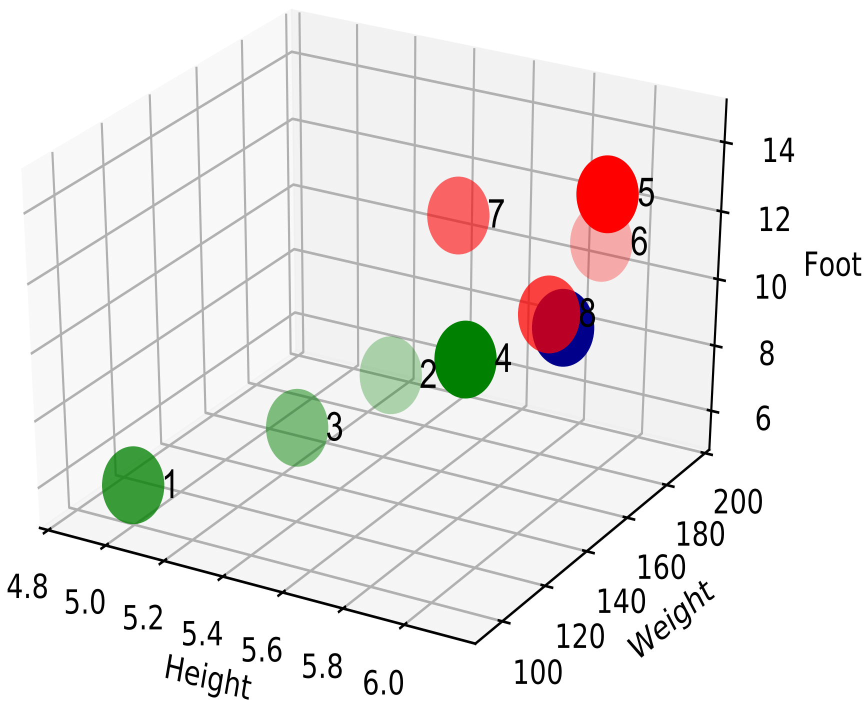
- $N = 8$ items
- $M = 3$ (unscaled) attributes

Code for the Dataset

```python
import pandas as pd
data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                    'Foot', 'Label'] )
```

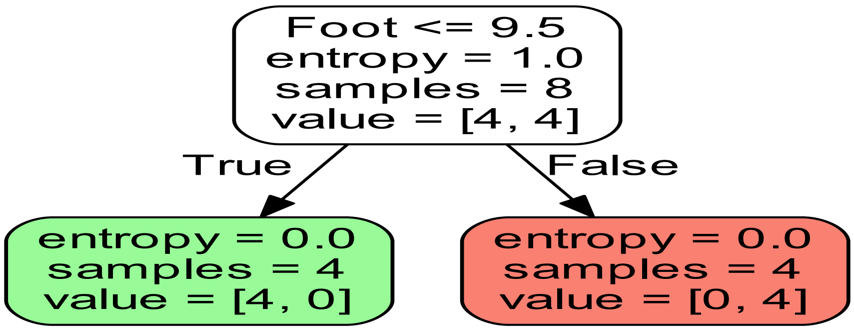
```
ipdb> data
  id Height Weight Foot Label
0  1   5.00    100    6 green
1  2   5.50    150    8 green
2  3   5.33    130    7 green
3  4   5.75    150    9 green
4  5   6.00    180   13   red
5  6   5.92    190   11   red
6  7   5.58    170   12   red
7  8   5.92    165   10   red
```

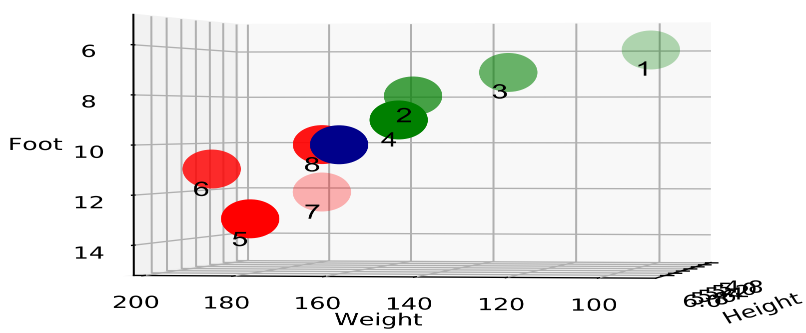
A New Instance



$$(\mathrm{H}=6,\ \mathrm{W}=160,\ \mathrm{F}=10) \mapsto ?$$

Decision Tree





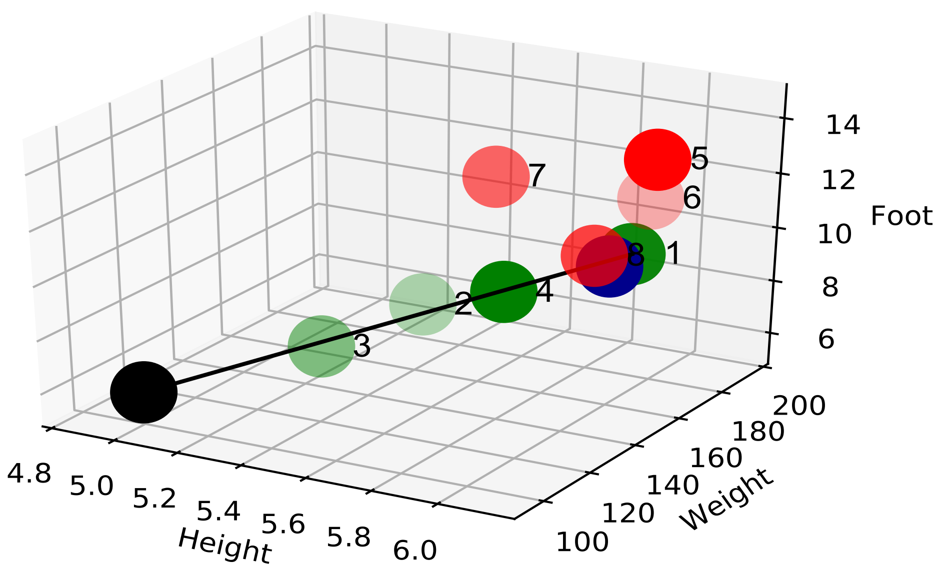
$$(\mathrm{H}=6,\ \mathrm{W}=160,\ \mathrm{F}=10) \mapsto \text{red}$$

Decision Tree in Python

```
import numpy as np
import pandas as pd
from sklearn import tree

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                         'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                            6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X,Y)

prediction = clf.predict(np.asmatrix([6, 160, 10]))
```

```
ipdb> prediction[0]

'red'
```

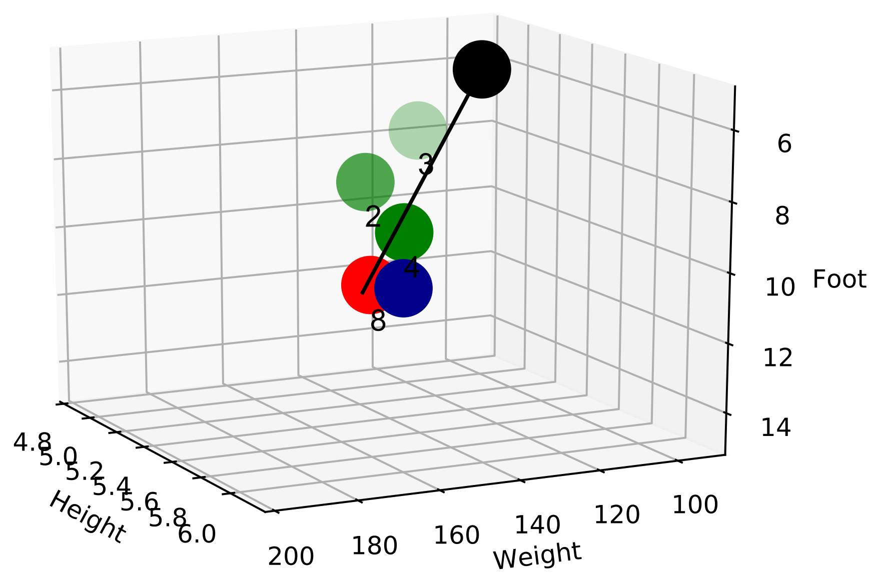
F/W/H Change



id	Height	Weight	Foot	Label
1	$5 \mapsto 6$	$100 \mapsto 170$	$6 \mapsto 10$	green

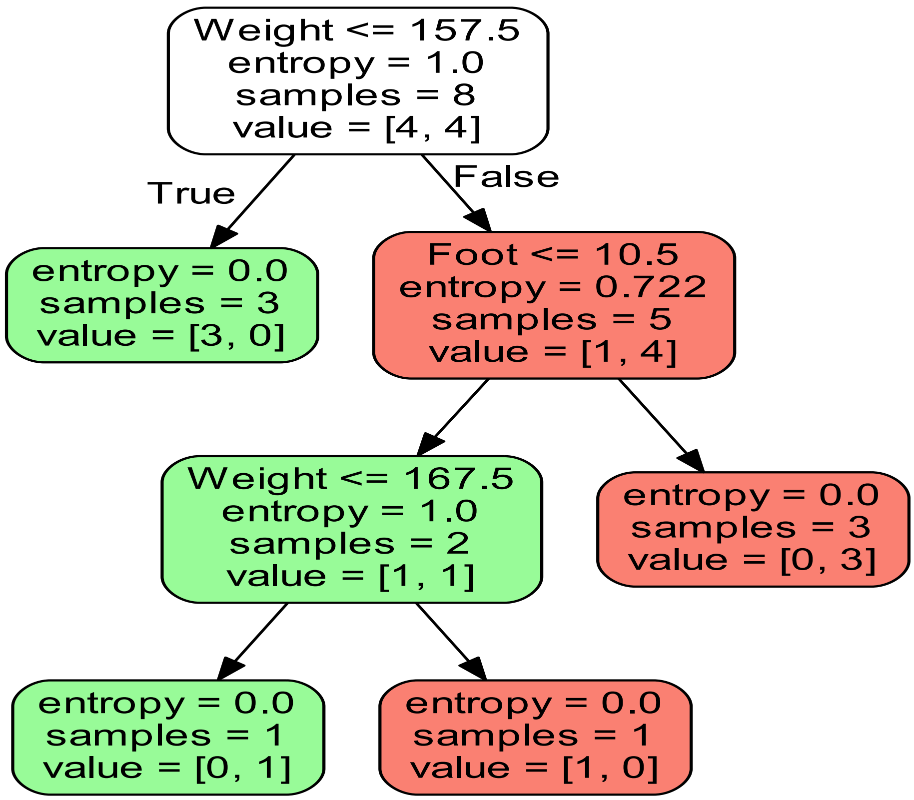
$$(\mathrm{H}{=}6,\ \mathrm{W}{=}160,\ \mathrm{F}{=}10) \mapsto\ ?$$

Intermediate Decision



- (H=6, W=160, F=10) $\mapsto$ green
- decide by weight, foot, weight

Decision Tree for F/W/H Change



$$(\mathrm{H}{=}6,\mathrm{W}{=}160,\mathrm{F}{=}10) \mapsto \text{green}$$

Code for F/W/H Change

```
import numpy as np
import pandas as pd
from sklearn import tree

data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                         'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                            6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                 180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )

data['Foot'].iloc[1] = 10    # change foot from 6 to 10!
data['Weight'].iloc[1] = 170 # weight from 100 to 170
data['Height'].iloc[1] = 6   # height from 5 to 6

X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X,Y)
prediction = clf.predict(np.asmatrix([6, 160, 10]))
```

```
ipdb> prediction[0]
'green'
```

RF in Python

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
data = pd.DataFrame( {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green', 'green', 'green', 'green',
                          'red', 'red', 'red', 'red'],
        'Height': [5, 5.5, 5.33, 5.75,
                             6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                                  180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                              'Foot', 'Label'] )
data['Foot'].iloc[1]   = 10; # foot from 6 to 10
data['Weight'].iloc[1] = 170 # weight from 100 to 170
data['Height'].iloc[1] = 6   # height from 5 to 6
X = data[['Height', 'Weight', 'Foot']].values
Y = data[['Label']].values
class_labels_dict = {'green': 1, 'red': 0}
label_color_dict = {1: 'green', 0: 'red'}
data['class_labels']= data['Label'].map(class_labels_dict)
model = RandomForestClassifier(n_estimators=5,max_depth=3,
                               criterion='entropy')
model.fit(X, Y)
test_instance = np.asmatrix([6, 160, 10])
rf_label = int(model.predict(test_instance)[0])
rf_color = label_color_dict[random_forest_label]
```

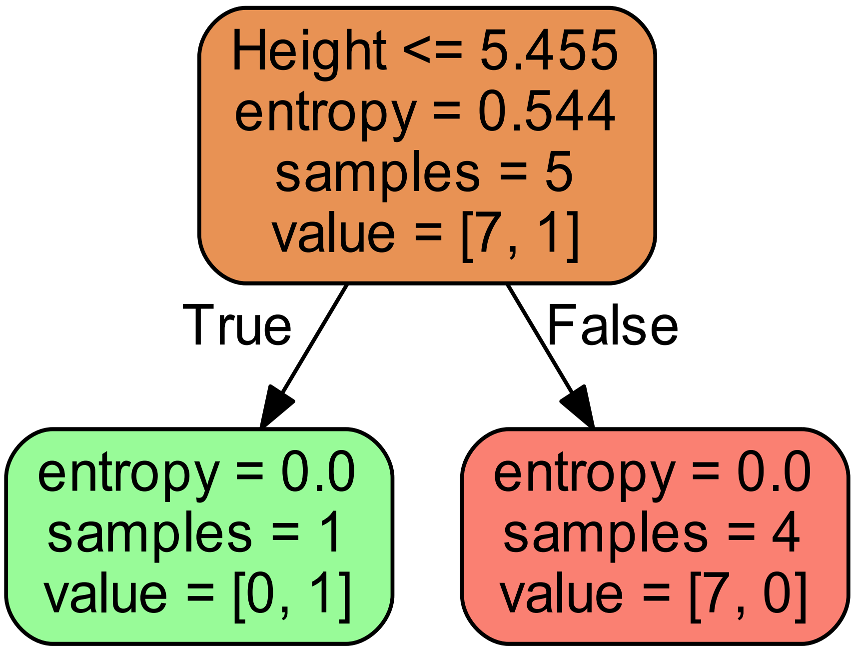
```
ipdb> rf_color
'red'
```

Python Code

```
for i in range(5):
    estimator = model.estimators_[i]
    rf_label = int(estimator.predict(test_instance)[0])
    rf_color = label_color_dict[label_prediction]
    print('estimator:', i, ' label = ', rf_label,' color: ',
        rf_color_color)
```

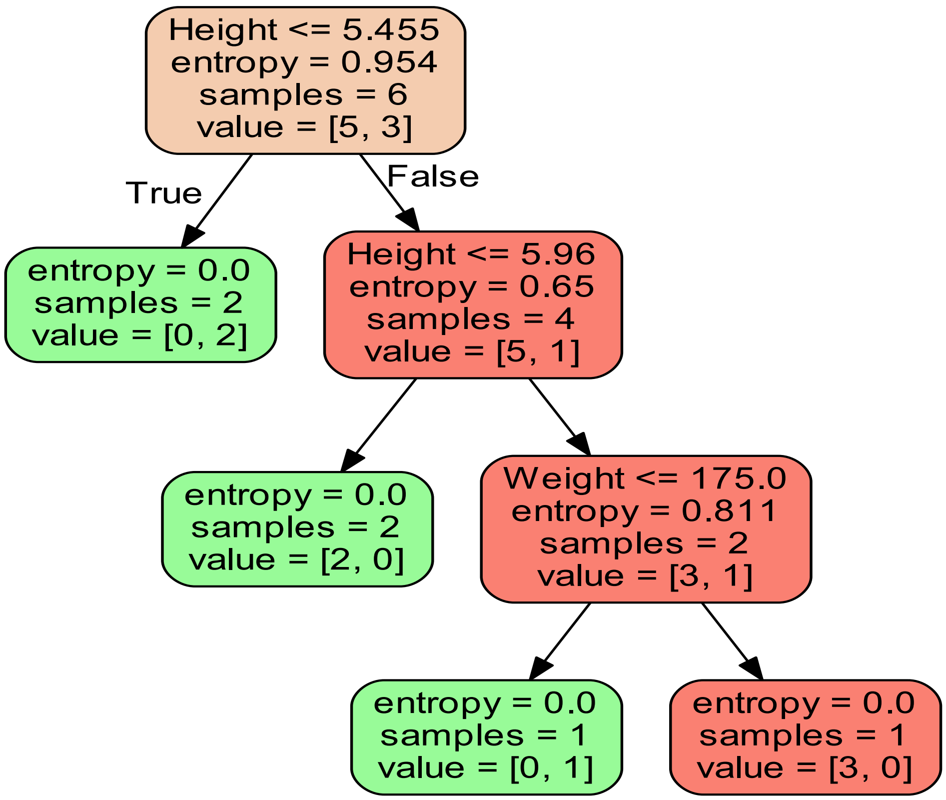
```
ipdb>
estimator: 0  label =  0  color:  red
estimator: 1  label =  0  color:  green
estimator: 2  label =  1  color:  green
estimator: 3  label =  1  color:  red
estimator: 4  label =  0  color:  red
```

RF Estimator 0



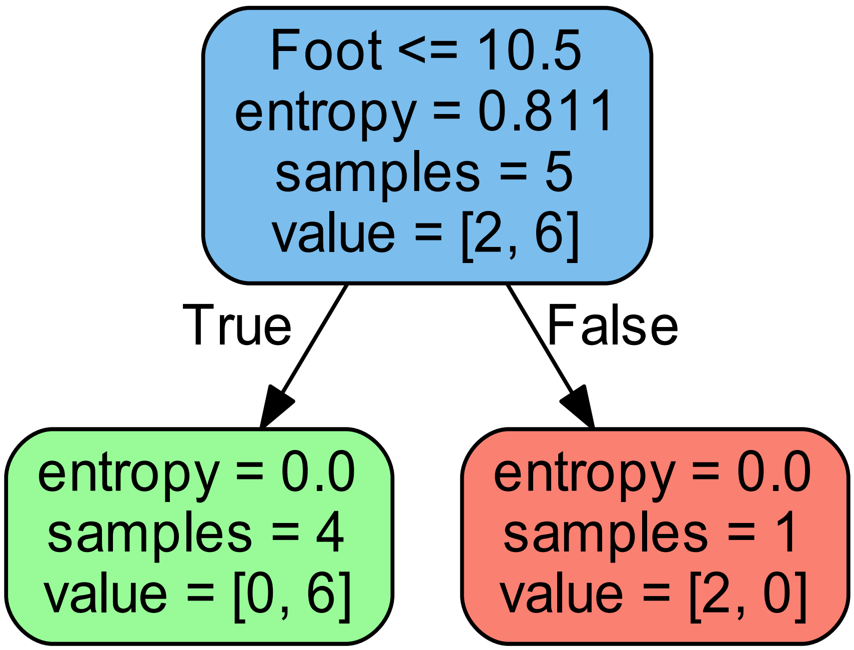
$$(\text{H}=6,\text{W}=160,\text{F}=10) \mapsto \text{red}$$

RF Estimator 1



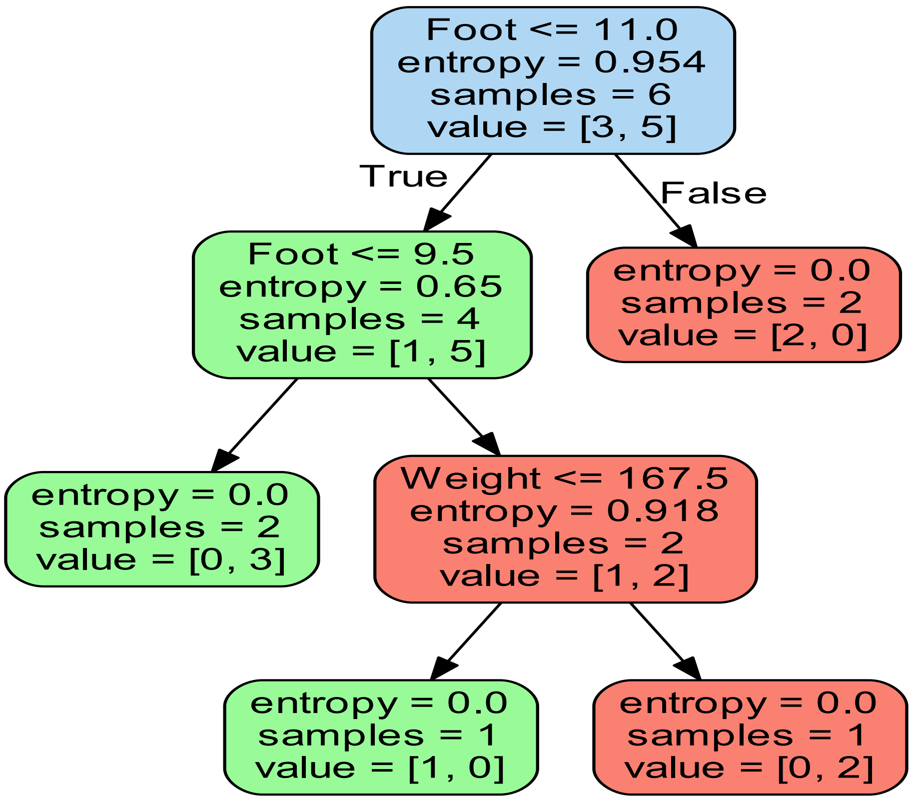
$$(\mathrm{H}=6,\mathrm{W}=160,\mathrm{F}=10) \mapsto \text{green}$$

RF Estimator 2



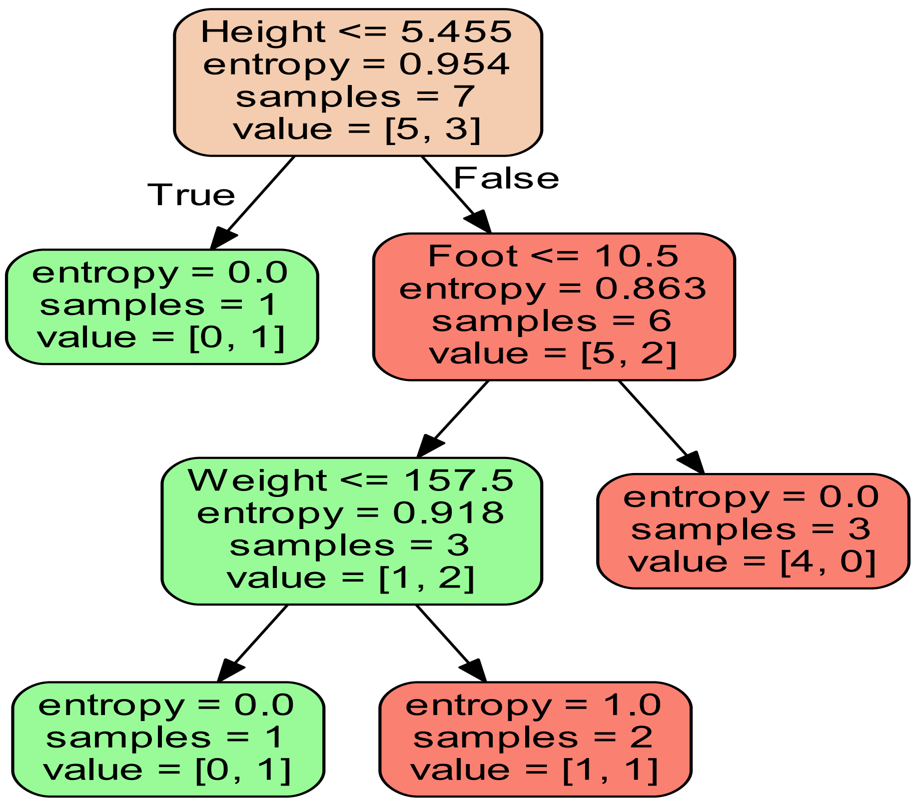
$$(\mathrm{H}=6,\mathrm{W}=160,\mathrm{F}=10) \mapsto \text{green}$$

RF Estimator 3



$$(\mathrm{H}{=}6,\mathrm{W}{=}160,\mathrm{F}{=}10) \mapsto \text{red}$$

RF Estimator: 4



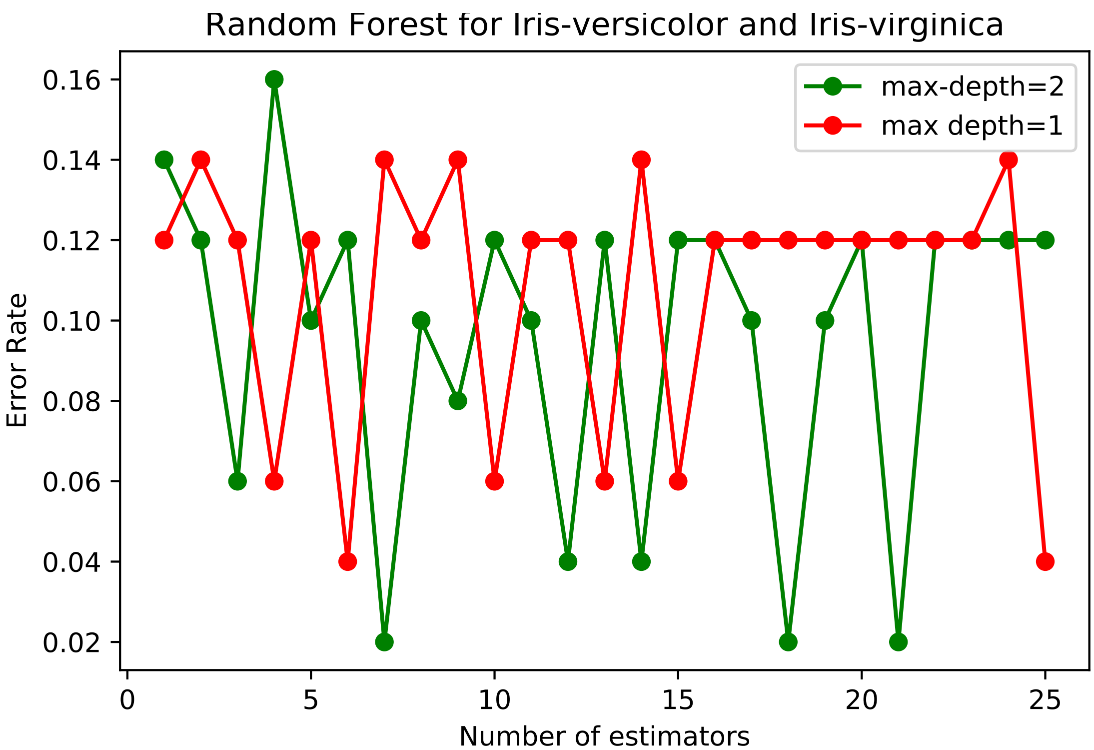
$$(H=6, W=160, F=10) \mapsto \text{red}$$

Random Forest: IRIS

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

url = r'https://archive.ics.uci.edu/ml/'  + \
          r'machine-learning-databases/iris/iris.data'

iris_feature_names = ['sepal-length', 'sepal-width',
                            'petal-length', 'petal-width']
data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                         'petal-length', 'petal-width', 'Class'])

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

X = data[iris_feature_names].values
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,
                                   test_size=0.5,random_state=3)

model = RandomForestClassifier(n_estimators=25,max_depth=5,
                               criterion='entropy')
model.fit(X_train, Y_train)
prediction = model.predict(X_test)
error_rate = np.mean(prediction != Y_test)
```

```
ipdb> error_rate
0.1
```

Impact of Depth and Number of Estimators



AdaBoost Method

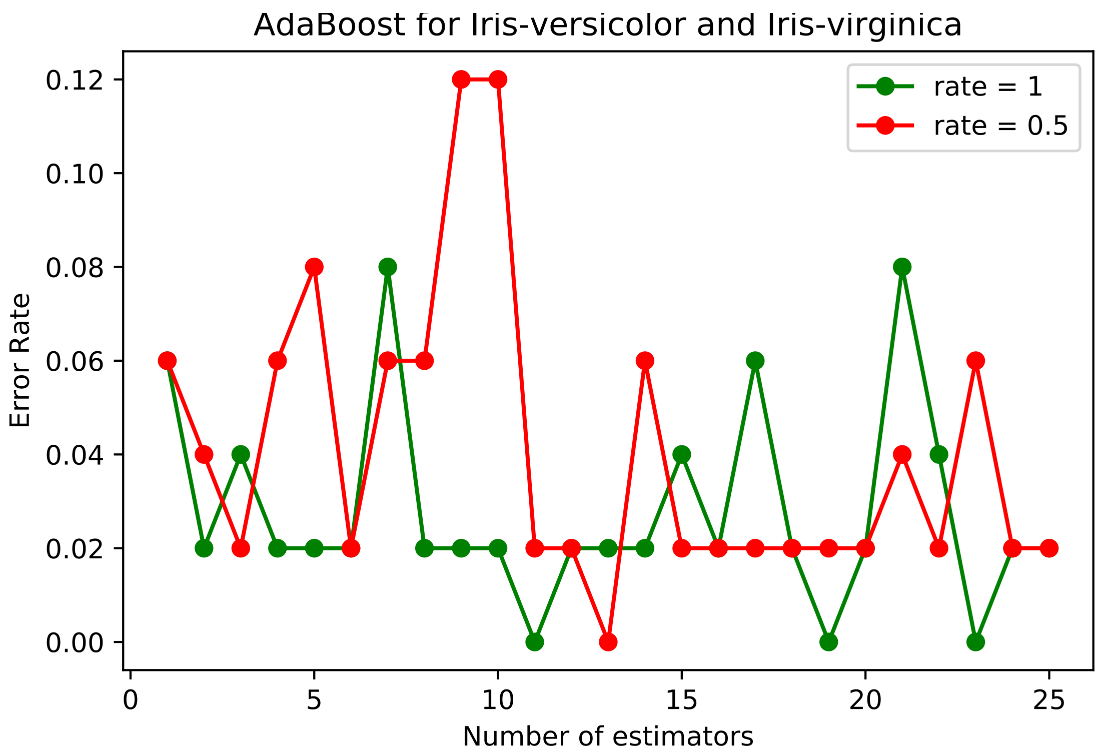
- choose a base classifier
- use k such classifiers $C_1, \ldots, C_k$ (”weak” learners)
- each C_i does a prediction on a subset X_i of data
- assign weight w_i to C_i based on its accuracy
- use weighted average of predictions

AdaBoost: IRIS

```
import pandas as pd
import numpy as np
from sklearn.svm import SVC # use SVM as base
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

url = r'https://archive.ics.uci.edu/ml/' + \
          r'machine-learning-databases/iris/iris.data'

iris_feature_names = ['sepal-length', 'sepal-width',
                            'petal-length', 'petal-width']
data = pd.read_csv(url, names=['sepal-length', 'sepal-width',
                        'petal-length', 'petal-width', 'Class'])

class_labels = ['Iris-versicolor', 'Iris-virginica']
data = data[data['Class'].isin(class_labels)]

X = data[iris_feature_names].values
le = LabelEncoder()
Y = le.fit_transform(data['Class'].values)

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,
                                    test_size=0.5,random_state=3)

svc=SVC(probability=True, kernel='linear')  # use as base learner
model = AdaBoostClassifier(n_estimators=5,base_estimator=svc,
                           learning_rate = 0.5)

model.fit(X_train, Y_train)
prediction = model.predict(X_test)
error_rate = np.mean(prediction != Y_test)
```

```
ipdb> error_rate
0.02
```

Impact of Learning Rate and Learners



Concepts Check:

(a) ensemble learning
(b) bagging
(c) advantages and disadvantages
(d) hyperparameters (estimators, max features, depth)
(e) Random Forest classification
(f) AdaBoost classification