Random Forests & Boosting Webinar

Adam Baybutt, PhD student at UCLA in economics (& computer science)

Today's Moderator: Jacob Koehler

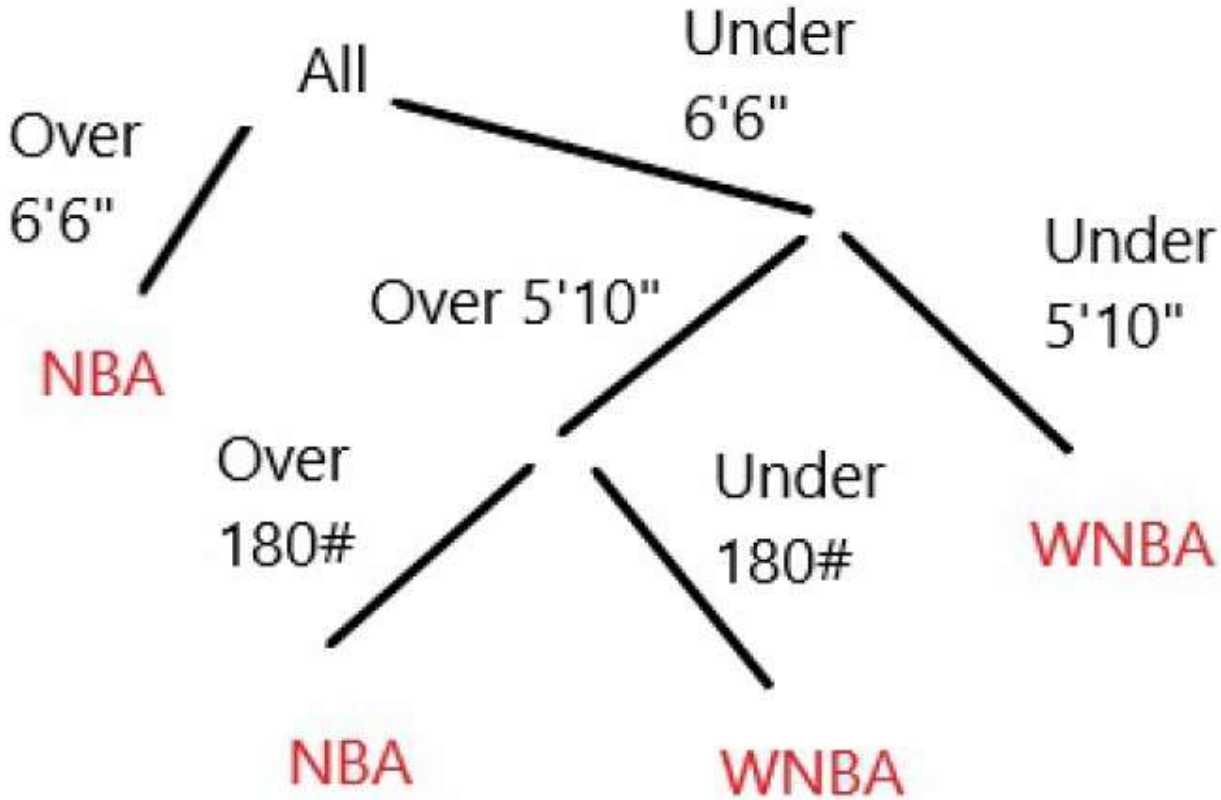
Overview:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - ### Decision Trees
  - ### Bootstrapping
  - ### Bagging
  - ### Random Forests
  - ### Boosting
- ### Examples:
  - ### Random forests
  - ### Boosting algorithm

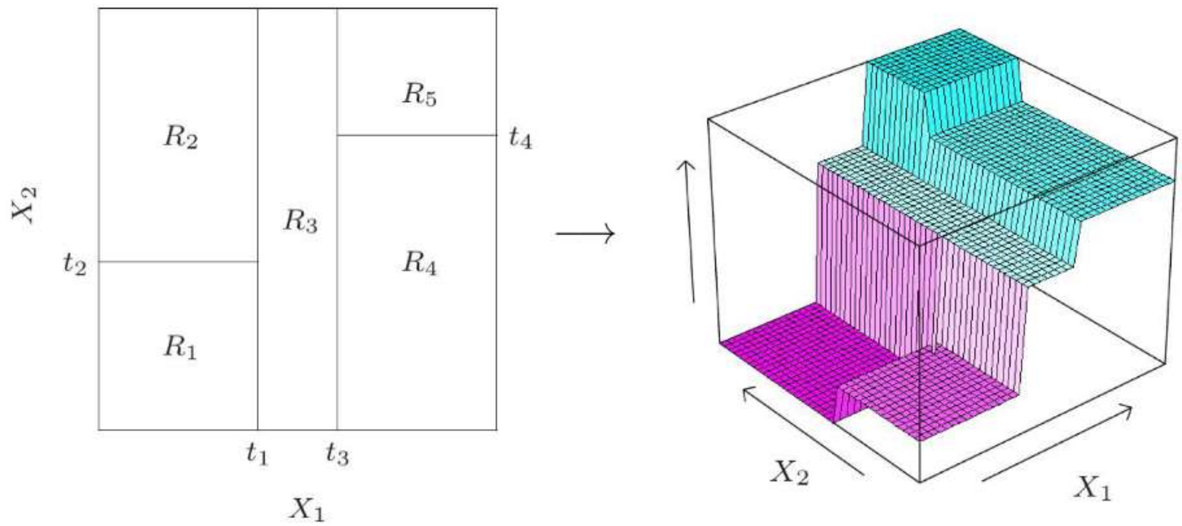
Why do we use decision trees, random forests, & boosting?

- Decision trees make no assumptions about the distribution of the data (i.e. nonparametric)
- Decision trees map regression and classification problems very well
- The resulting model is clear for interpretability (e.g. important variables, model can be easily visualize for non-technical audience, etc.)
- Random forests are a simple, clear extension of decisions trees to improve performance
  - Using the bootstrap is one way it improves over a simple decision tree as the bootstrap often reduces bias in finite samples (i.e. asymptotic refinements)
- Empirical applications have shown random forests and boosted decision trees to be effective out-of-the-box algorithms, especially with a non-linear function
  - XGBoost is a very popular gradient boosted decision tree algorithm that has won many online data science competitions

Review of concepts: Decision Trees


All
Over
6'6"
NBA
Under
6'6"
Over 5'10"
Under
5'10"
WNBA
Over
180#
Under
180#
NBA
WNBA

REGRESSION TREES



Adding an output dimension to the figure (right), we can see how regression trees can learn a step function approximation to the data.

How to build a tree?

Look at all the cut points, of all the variables, and decide which ones improves the algorithm the most.

Well, what is "improves the algorithm the most"?

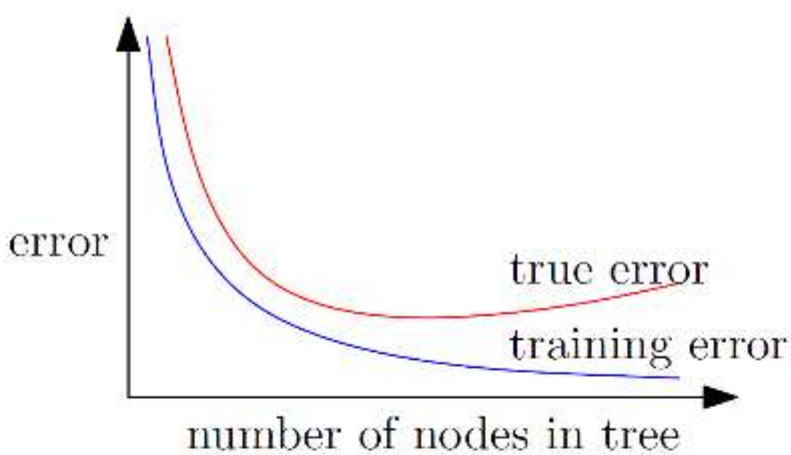
Decision rules

- ### Classification error
- ### Entropy
- ### Information gain
- ### Gini impurity

Why not grow a huge tree for minimal training error?

What's the answer in 95% of machine learning questions?

AVOID OVER-FITTING!



There are loads of "regularization" methods to find minimum of test error by not over-fitting on training error.

- Tree depth
- Number of leaves
- Number of nodes
- Leaf size
- Limit splits to above a certain classification error reduction
- Pruning (i.e. total cost formula to find optimal tree complexity and training error)
- Among many others...

Agenda:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - Decision Trees
  - ### Bootstrapping
  - Bagging
  - Random Forests
  - Boosting
- ### Examples:
  - Random forests
  - Boosting algorithm

Review of concepts: Bootstrapping

The bootstrap is resampling from our data to estimate the distribution of an estimator.

Bootstrap: Basic algorithm

Input

- A sample of data $x_1, \ldots, x_n$.
- An estimation rule $\hat{S}$ of a statistic S. For example, $\hat{S} = \text{med}(x_{1:n})$ estimates the true median S of the unknown distribution on x.

Bootstrap algorithm

1. Generate *bootstrap samples* $\mathcal{B}_1, \ldots, \mathcal{B}_B$.
   - Create $\mathcal{B}_b$ by picking points from $\{x_1, \ldots, x_n\}$ randomly n times.
   - A particular x_i can appear in $\mathcal{B}_b$ many times (it's simply duplicated).
2. Evaluate the estimator on each $\mathcal{B}_b$ by pretending it's the data set:

$$\hat{S}_b := \hat{S}(\mathcal{B}_b)$$

3. Estimate the mean and variance of $\hat{S}$:

$$\mu_B = \frac{1}{B}\sum_{b=1}^{B} \hat{S}_b, \quad \sigma_B^2 = \frac{1}{B}\sum_{b=1}^{B} (\hat{S}_b - \mu_B)^2$$

- I do not follow the math for why the bootstrap actually gives improved finite sample performance -- termed asymptotic refinements -- (it seems like magic), but intuitively it makes some sense:
  - given our random sample, each observation had an equal probability of arising
  - thus, when we take a random sample of our random sample, our data still have an equal probability of ending up in the bootstrap random sample
  - So, it is as if we are able to conduct many experiments but we are working with a smaller data set than the population; but, hopefully because these data were collected that they should be over weighted

Agenda:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - Decision Trees
  - Bootstrapping
  - ### Bagging
  - Random Forests
  - Boosting
- ### Examples:
  - Random forests
  - Boosting algorithm

Review of concepts: Bagging

Bagging is Bootstrap AGGregation where we create many bootstrap samples, fit models on all, and then combine.

BAGGING

Bagging uses the bootstrap for regression or classification:

Bagging = **B**ootstrap **agg**regation

Algorithm

For $b = 1, \ldots, B$:

1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from training data.
2. Train a classifier or regression model f_b on $\mathcal{B}_b$.

- For a new point x_0, compute:

$$f_{\text{avg}}(x_0) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_0)$$

  - For regression, $f_{\text{avg}}(x_0)$ is the prediction.
  - For classification, view $f_{\text{avg}}(x_0)$ as an average over B votes. Pick the majority.

- A good analogy is it is the wisdom of the crowds: each model doesn't need to be great but their average is highly predictive

Agenda:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - Decision Trees
  - Bootstrapping
  - Bagging
  - ### Random Forests
  - Boosting
- ### Examples:
  - Random forests
  - Boosting algorithm

Review of concepts: Random Forests

Random Forest is an ensemble algorithm which creates many decision trees using bagging and dropping variables.

RANDOM FORESTS: ALGORITHM

Training

Input parameter: m — a positive integer with $m < d$, often $m \approx \sqrt{d}$

For $b = 1, \dots, B$:

1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from the training data.
2. Train a tree classifier on $\mathcal{B}_b$, where each split is computed as follows:
   - Randomly select m dimensions of $x \in \mathbb{R}^d$, newly chosen for each b.
   - Make the best split restricted to that subset of dimensions.

We set m ~= sqrt(d), which is an empirical result as opposed to a mathematical result.

Improvement from reducing the correlation between each tree.

Agenda:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - Decision Trees
  - Bootstrapping
  - Bagging
  - Random Forests
  - ### Boosting
- ### Examples:
  - Random forests
  - Boosting algorithm

Review of concepts: Boosting

Boosted is an intelligent way of improving weakness in the model with each new bootstrap sample+model fit.

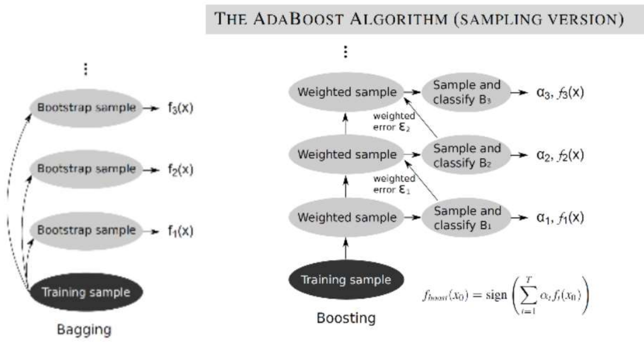
THE ADABOOST ALGORITHM (SAMPLING VERSION)

Algorithm: Boosting a binary classifier

Given $(x_1, y_1), \dots, (x_n, y_n)$, $x \in \mathcal{X}$, $y \in \{-1, +1\}$, set $w_1(i) = \frac{1}{n}$

- For $t = 1, \dots, T$
  1. Sample a bootstrap dataset $\mathcal{B}_t$ of size n according to distribution w_t. Notice we pick (x_i, y_i) with probability $w_t(i)$ and not $\frac{1}{n}$.
  2. Learn a classifier f_t using data in $\mathcal{B}_t$.
  3. Set $\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbb{1}\{y_i \neq f_t(x_i)\}$ and $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
  4. Scale $\hat{w}_{t+1}(i) = w_t(i) e^{-\alpha_t y_i f_t(x_i)}$ and set $w_{t+1}(i) = \frac{\hat{w}_{t+1}(i)}{\sum_j \hat{w}_{t+1}(j)}$.
- Set the classification rule to be

$$f_{boost}(x_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x_0)\right).$$



Agenda:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - Decision Trees
  - Bootstrapping
  - Bagging
  - Random Forests
  - Boosting
- ### Examples:
  - ### Random forests
  - Boosting algorithm

Examples: Random Forests

Admissions data from last week's webinar

Classify [admissions] using predictors: [gre, gpa, prestige]

In [6]:
```
# Imports
from sklearn.externals.six import StringIO
from IPython.display import Image
l
import pydotplus
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
from sklearn.model_selection import train_test_split

plt.style.use('fivethirtyeight')

from ipywidgets import *
from IPython.display import display

%matplotlib inline
%config InlineBackend.figure_format = 'retina'

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

In [7]:
```
# Load admissions data
admit = pd.read_csv('datasets/admissions.csv')
```

In [8]:
```
# Drop the nulls instead of imputing right now, to save time
admit = admit.dropna()
admit.head()
```

Out[8]:

	admit	gre	gpa	prestige
0	0	380.0	3.61	3.0
1	1	660.0	3.67	3.0
2	1	800.0	4.00	1.0
3	1	640.0	3.19	4.0
4	0	520.0	2.93	4.0

In [9]:
```
### Clean up the data set
y_class = [-1 if k == 0 else k for k in admit['admit']]
X_class = admit[['gpa','gre','prestige']]
```

In [10]:
```
# Create tts
X_train, X_test, y_train, y_test = train_test_split(
    X_class, y_class,
    test_size = .25, random_state = 42)
```

In [41]:
```python
len(y_test)
```

Out[41]:
```
100
```

In [42]:
```python
len(y_train)
```

Out[42]:
```
297
```

In [11]:
```python
# Build tree
model_dt = DecisionTreeClassifier(max_depth=2)
model_dt.fit(X_train, y_train)
```

Out[11]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
2,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=N
one,
            splitter='best')
```

In [44]:
```python
model_dt_scores = cross_val_score(model_dt, X_train, y_train, cv=4)
print("Decision Classifier max depth 4: ", model_dt_scores, 'Mean score:'
, np.mean(model_dt_scores))
model_dt.score(X_test,y_test)
```

```
Decision Classifier max depth 4:  [0.69333333 0.7027027  0.7027027  0.7
027027 ] Mean score: 0.7003603603603603
```

Out[44]:
```
0.63
```

In [50]:
```python
model_rf = RandomForestClassifier(n_estimators=20, max_features=2, max_de
pth=4, bootstrap=True)
model_rf.fit(X_train, y_train)
model_rf_scores = cross_val_score(model_rf, X_train, y_train, cv=4)
print("Decision Classifier max depth 4: ", model_rf_scores, 'Mean score:'
, np.mean(model_rf_scores))
model_rf.score(X_test,y_test)
```

```
Decision Classifier max depth 4:  [0.69333333 0.74324324 0.75675676 0.7
027027 ] Mean score: 0.724009009009009
```

Out[50]:
```
0.7
```

Let's use the college dataset to see if we can get a similiar improvement

In [51]:
```python
col = pd.read_csv('datasets/College.csv')
y   = col.Private.map(lambda x: 1 if x == 'Yes' else -1)
X   = col.iloc[:, 2:]
```

In [13]:
```
y.shape
```
Out[13]:
```
(777,)
```

In [14]:
```
X.head()
```
Out[14]:

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate	Room.Boar
0	1660	1232	721	23	52	2885	537	7440	330
1	2186	1924	512	16	29	2683	1227	12280	645
2	1428	1097	336	22	50	1036	99	11250	375
3	417	349	137	60	89	510	63	12960	545
4	193	146	55	16	44	249	869	7560	412

In [15]:
```
X.shape
```
Out[15]:
```
(777, 17)
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size = .25, random_state = 1)
```

In [17]:
```
model_dt = DecisionTreeClassifier(max_depth=5)
model_dt.fit(X_train, y_train)
```
Out[17]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
5,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=N
one,
            splitter='best')
```

In [18]:
```
model_dt_scores = cross_val_score(model_dt, X_train, y_train, cv=4)
print("Decision Classifier max depth 4: ",model_dt_scores, 'Mean score:',
 np.mean(model_dt_scores))
```
```
Decision Classifier max depth 4:  [0.87755102 0.92413793 0.91724138 0.9
2413793] Mean score: 0.9107670654468684
```

In [19]:
```
model_dt.score(X_test,y_test)
```
Out[19]:
```
0.9128205128205128
```

In [20]:
```
max_depth_input = 5
model_rf = RandomForestClassifier(n_estimators=100, max_features=6, max_d
epth=max_depth_input, bootstrap=True)
model_rf.fit(X_train, y_train)
model_rf_scores = cross_val_score(model_rf, X_train, y_train, cv=4)
print("Random Forest Classifier max depth ",max_depth_input, ": ", model_
rf_scores, 'Mean score:', np.mean(model_rf_scores))
model_rf.score(X_test,y_test)
```

```
Random Forest Classifier max depth  5 :  [0.93197279 0.93793103 0.92413
793 0.95172414] Mean score: 0.9364414731409806
```

Out[20]: 0.9384615384615385

Agenda:

- ### Why do we use decision trees, random forests, & boosting
- ### Review of concepts:
  - Decision Trees
  - Bootstrapping
  - Bagging
  - Random Forests
  - Boosting
- ### Examples:
  - Random forests
  - ### Boosting algorithm

Examples: Boosting

In [21]:
```
#Import
from sklearn.ensemble import AdaBoostClassifier
```

In [22]:
```
col = pd.read_csv('datasets/College.csv')
y =col.Private.map(lambda x: 1 if x == 'Yes' else -1)
X = col.iloc[:, 2:]
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size = .25, random_state = 1)
```

In [23]:
```
model_bst = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_
features=6, max_depth=5), n_estimators=200, learning_rate=0.01, random_st
ate=1)
model_bst.fit(X_train, y_train)
model_bst.score(X_test,y_test)
```

Out[23]: 0.958974358974359

EXTRAS!

Let's see how gradient boosting does

In [24]:
```
#Import
from sklearn.ensemble import GradientBoostingClassifier
```

In [25]:
```
col = pd.read_csv('datasets/College.csv')
y   = col.Private.map(lambda x: 1 if x == 'Yes' else -1)
X   = col.iloc[:, 2:]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25
, random_state = 1)
```

In [26]:
```
model_grad_bst = GradientBoostingClassifier(max_features=8, max_depth=5,
n_estimators=400, learning_rate=0.05, random_state=1)
model_grad_bst.fit(X_train, y_train)
model_grad_bst.score(X_test,y_test)
```

Out[26]:
```
0.9435897435897436
```

What about the admissions data?

In [27]:
```
admit = pd.read_csv('datasets/admissions.csv')
admit = admit.dropna()
y = [-1 if k == 0 else k for k in admit['admit']]
X = admit[['gpa','gre','prestige']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25
, random_state = 2)
```

In [28]:
```
model_grad_bst = GradientBoostingClassifier(max_features=2, max_depth=2,
n_estimators=2750, learning_rate=0.001, random_state=1)
model_grad_bst.fit(X_train, y_train)
model_grad_bst_scores = cross_val_score(model_grad_bst, X_train, y_train,
 cv=4)
print("Grad Boost Classifier: ",model_grad_bst_scores, 'Mean score:', np.
mean(model_grad_bst_scores))
model_grad_bst.score(X_test,y_test)
```

```
Grad Boost Classifier:  [0.74666667 0.70666667 0.66216216 0.73972603] M
ean score: 0.7138053807231889
```

Out[28]:
```
0.69
```

Why do we care about balanced classes?

In [52]:
```
# let's get the college private/public data again, but let's make it imba
lanced
col = pd.read_csv('datasets/College.csv')
col_priv = col.loc[col['Private'] == 'Yes']
col_pub = col.loc[col['Private'] == 'No']
```

In [53]:
```
col_priv.shape
```

Out[53]: (565, 19)

In [54]:
```
col_pub.shape
```

Out[54]: (212, 19)

In [55]:
```
col_priv = col_priv.sample(200)
col_pub = col_pub.sample(200)
col = col_priv.append(col_pub)
col.shape
```

Out[55]: (400, 19)

In [56]:
```
y   = col.Private.map(lambda x: 1 if x == 'Yes' else -1)
X   = col.iloc[:, 2:]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25
, random_state = 1)
```

In [57]:
```
max_depth_input = 5
model_rf = RandomForestClassifier(n_estimators=100, max_features=6, max_d
epth=max_depth_input, bootstrap=True)
model_rf.fit(X_train, y_train)
model_rf_scores = cross_val_score(model_rf, X_train, y_train, cv=4)
print("Random Forest Classifier max depth ",max_depth_input, ": ", model_
rf_scores, 'Mean score:', np.mean(model_rf_scores))
model_rf.score(X_test,y_test)
```

```
Random Forest Classifier max depth  5 :  [0.92105263 0.84210526 0.93243
243 0.93243243] Mean score: 0.9070056899004267
```

Out[57]: 0.93

In [58]:
```
model_grad_bst = GradientBoostingClassifier(max_features=2, max_depth=9,
n_estimators=200, learning_rate=0.005, random_state=1)
model_grad_bst.fit(X_train, y_train)
model_grad_bst.score(X_test,y_test)
```

Out[58]: 0.96

References:

- https://en.wikipedia.org/wiki/Bootstrapping_%28statistics%29 (https://en.wikipedia.org/wiki/Bootstrapping_%28statistics%29)
- https://en.wikipedia.org/wiki/Random_forest (https://en.wikipedia.org/wiki/Random_forest)
- https://en.wikipedia.org/wiki/Gradient_boosting#Informal_introduction (https://en.wikipedia.org/wiki/Gradient_boosting#Informal_introduction)
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html)
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html)
- Joel Horowitz. 2001. The Bootstrap. Handbook of Econometrics. https://www.sciencedirect.com/science/article/pii/S157344120105005X?via%3Dihub (https://www.sciencedirect.com/science/article/pii/S157344120105005X?via%3Dihub)
- Elements of Stat. Learning by Hastie https://web.stanford.edu/~hastie/ElemStatLearn// (https://web.stanford.edu/~hastie/ElemStatLearn//)

In [ ]:

```

```