Lecture 9
Ensemble Learning

GEOL 4397: Data analytics and machine learning for geoscientists

Jiajia Sun, Ph.D.

March. 5th, 2018

UNIVERSITY of HOUSTON
YOU ARE THE PRIDE
EARTH AND ATMOSPHERIC SCIENCES

Agenda

- Basic idea
- Motivation
- Bagging
- Boosting
- Implementation in Scikit-Learn
- Exam

Basic idea

- Train an ensemble of machine learning models (i.e., predictors)
- Aggregate the predictions from all the predictors

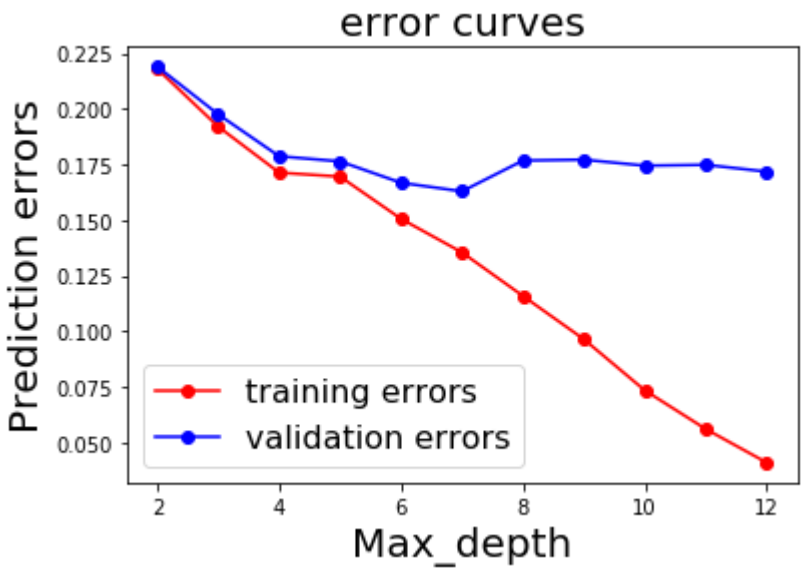
Basic idea

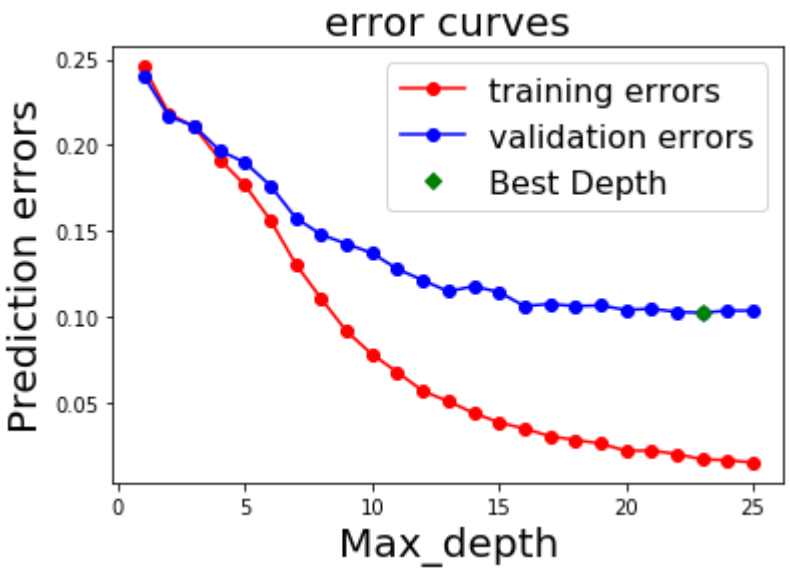
- Train an ensemble of machine learning models (i.e., predictors)
- Aggregate the predictions from all the predictors
- Often get better predictions than with the best individual predictor

Basic idea

- Train an ensemble of machine learning models (i.e., predictors)
- Aggregate the predictions from all the predictors
- Often get better predictions than with the best individual predictor
- Wisdom of crowd

Random Forest



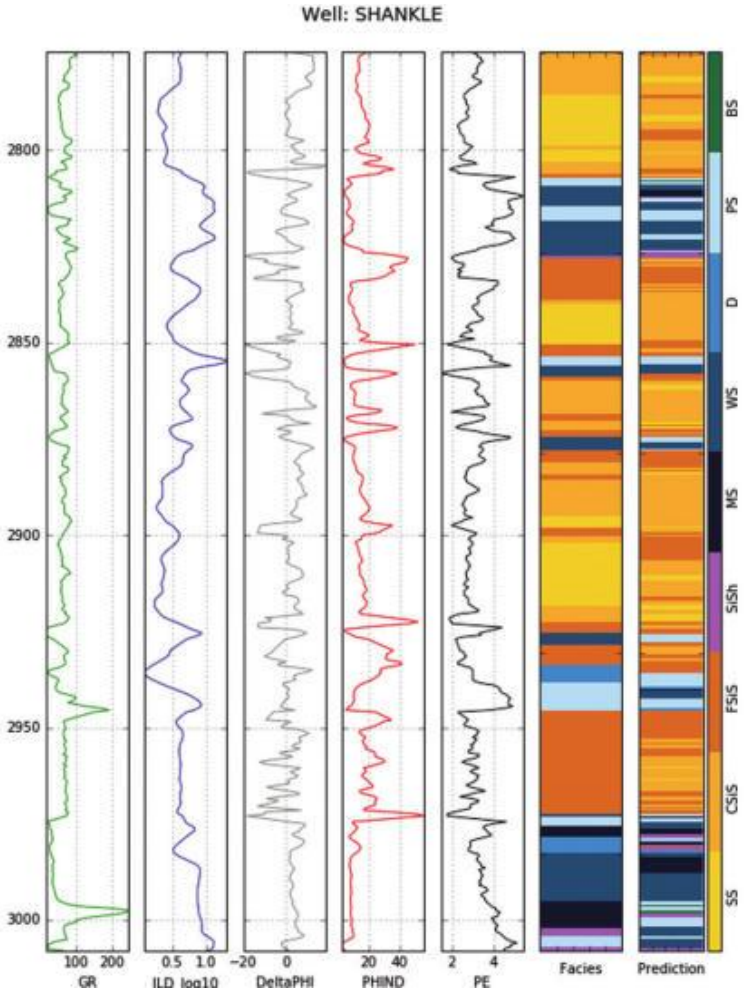


Best-performing decision tree: 83% accuracy

Best-performing random forests: 90% accuracy

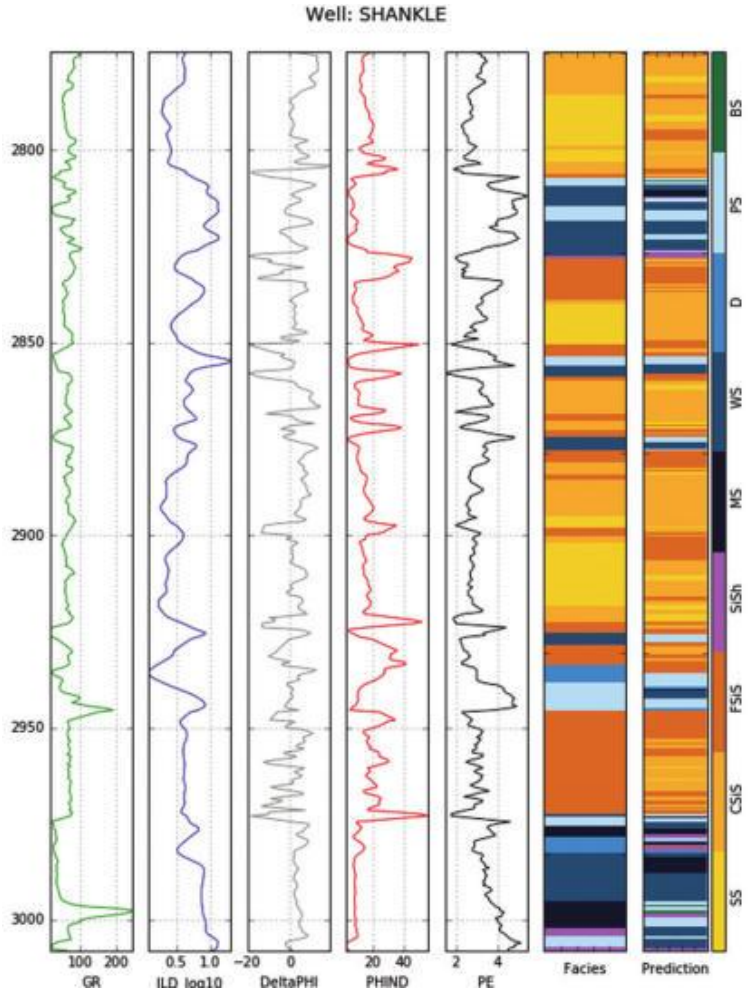
Agenda

- Basic idea
- Motivation
- Bagging
- Boosting
- Implementation in Scikit-Learn



Hall, 2016, TLE

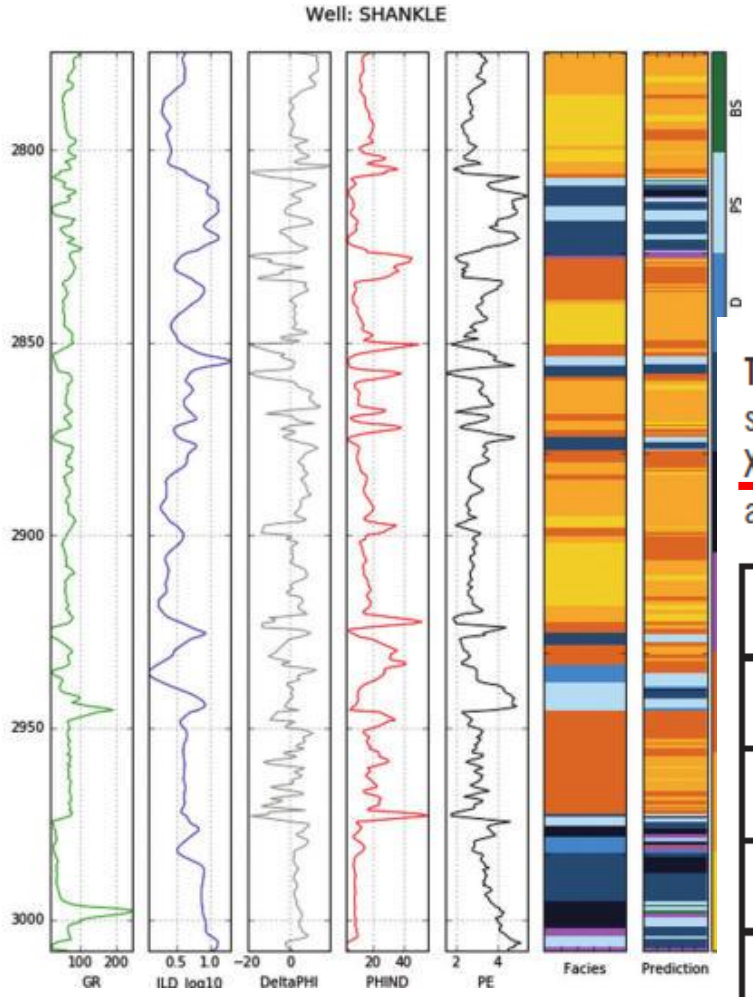
Machine learning contest on predicting facies



Hall, 2016, TLE

- Received 300 solutions from 40 teams
- All of the top 10 solutions used ensemble learning.

Machine learning contest on predicting facies



Hall, 2016, TLE

- Received 300 solutions from 40 teams
- All of the top 10 solutions used ensemble learning.

Table 1. The top five teams at the end of the contest, with median accuracy scores from 100 realizations of their models. All of these entries implemented the XGBoost model in Python. You can see all of the teams, with deterministic scores, at github.com/seg/2016-ml-contest.

1	LA Team (Mosser & de la Fuente)	0.6388
2	PA Team (PetroAnalytix)	0.6250
3	ISPL (Bestagini, Tubaro & Lipari)	0.6231
4	esaTeam (Earth Science Analytics)	0.6225
5	SHandPR (Hall & Raghavan)	0.6200

https://www.kaggle.com/competitions

General | InClass | Sort by Grouped

All Categories | Search competitions

18 Active Competitions

Two Sigma: Using News to Predict Stock Movements — $100,000, 2,927 teams
Use news analytics to predict stock price performance
Featured · Kernels Competition · 4 months to go · news agencies, time series, finance, money

Santander Customer Transaction Prediction — $65,000, 3,977 teams
Can you identify who will make a transaction?
Featured · a month to go · banking, tabular data, binary classification

LANL Earthquake Prediction — $50,000, 1,521 teams
Can you predict upcoming laboratory earthquakes?
Research · 3 months to go · signal processing, earth sciences, physics

Gendered Pronoun Resolution — $25,000, 311 teams
Pair pronouns to their correct entities
Research · 2 months to go · nlp, text data

Google Cloud & NCAA® ML Competition 2019-Women's — $25,000, 157 teams
Apply Machine Learning to NCAA® March Madness®
Featured · a month to go · basketball, sports

Google Cloud & NCAA® ML Competition 2019-Men's — $25,000, 254 teams
Apply Machine Learning to NCAA® March Madness®
Featured · a month to go · basketball, sports

PetFinder.my Adoption Prediction — $25,000, 1,512 teams
How cute is that doggy in the shelter?
Featured · Kernels Competition · 23 days to go · image data, text data

kaggle Search kaggle Competitions Datasets Kernels Discussion Jobs ••• Sign In

Overview Data Kernels Discussion Leaderboard Rules Join Competition

Overview

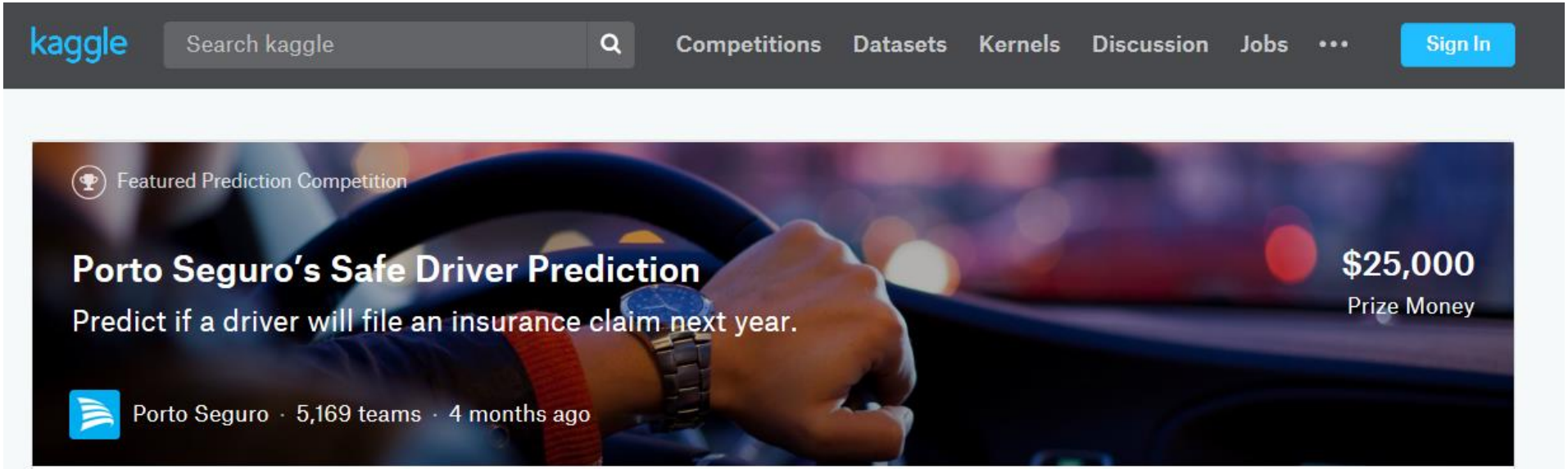
- Description
- Evaluation
- Prizes
- Timeline

Nothing ruins the thrill of buying a brand new car more quickly than seeing your new insurance bill. The sting's even more painful when you know you're a good driver. It doesn't seem fair that you have to pay so much if you've been cautious on the road for years.

Porto Seguro, one of Brazil's largest auto and homeowner insurance companies, completely agrees. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones.

In this competition, you're challenged to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year. While Porto Seguro has used machine learning for the past 20 years, they're looking to Kaggle's machine learning community to explore new, more powerful methods. A more accurate prediction will allow them to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

https://goo.gl/MfnZEz



Overview

Description

Evaluation

Prizes

Timeline

See the Rules Section labeled "Prizes" for the terms on receiving prize money.

- 1st place - $12,000
- 2nd place - $8,000
- 3rd place - $5,000

https://goo.gl/MfnZEz

Leaderboard

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	**Michael Jahrer**			0.29698	83	4mo
2	▲ 3	三个臭皮匠还是打不过诸葛亮			0.29413	231	4mo
3	▲ 1071	**utility**			0.29271	12	4mo
4	▲ 1101	**Dmitriy & Alan**			0.29249	92	4mo
5	▲ 4	**Encik Besi**			0.29228	47	4mo
6	▲ 1091	**Guanshuo Xu**			0.29221	58	4mo
7	▼ 3	**Evgeny Patekha**			0.29190	65	4mo
8	▼ 5	**MSChuan-ironbar**			0.29189	252	4mo

https://goo.gl/6tcuXc

1st place solution

- An ensemble of 6 models
  - 1 LightGBM (one type of gradient boosting)
  - 5 Neural Networks

- To learn more about his solution:
  - https://goo.gl/uaxgyP

Netflix Prize

- Sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

https://netflixprize.com/community/topic_1537.html

Netflix Prize

- Sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

- Winning team 'BellKor's Pragmatic Chaos' was awarded $1M Grand Prize.

https://netflixprize.com/community/topic_1537.html

The BellKor Solution to the Netflix Grand Prize

Yehuda Koren

August 2009

I. Introduction

This article describes part of our contribution to the "BellKor's Pragmatic Chaos" final solution, which won the Netflix Grand Prize. The other portion of the contribution was created while working at AT&T with Robert Bell and Chris Volinsky, as reported in our 2008 Progress Prize report [3]. The final solution includes all the predictors described there. In this article we describe only the newer predictors.

So what is new over last year's solution? First we further improved the baseline predictors (Sec. III). This in turn improves our other models, which incorporate those predictors, like the matrix factorization model (Sec. IV). In addition, an extension of the neighborhood model that addresses temporal dynamics was introduced (Sec. V). On the Restricted Boltzmann Machines (RBM) front, we use a new RBM model with superior accuracy by conditioning the visible units (Sec. VI). The final addition is the introduction of a new blending algorithm, which is based on gradient boosted decision trees (GBDT) (Sec. VII).

II. Preliminaries

The Netflix dataset contains more than 100 million date-stamped movie ratings performed by anonymous Netflix customers between Dec 31, 1999 and Dec 31, 2005 [4]. This dataset gives ratings about $m = 480,189$ users and $n = 17,770$ movies (aka, items).

therefore harder to predict. In a way, this represents real requirements for a collaborative filtering (CF) system, which needs to predict new ratings from older ones, and to equally address all users, not just the heavy raters.

We reserve special indexing letters to distinguish users from movies: for users u, v, and for movies i, j. A rating r_{ui} indicates the preference by user u of movie i. Values are ranging from 1 (star) indicating no interest to 5 (stars) indicating a strong interest. We distinguish predicted ratings from known ones, by using the notation $\hat{r}_{ui}$ for the predicted value of r_{ui}.

The scalar t_{ui} denotes the time of rating r_{ui}. Here, time is measured in days, so t_{ui} counts the number of days elapsed since some early time point. About 99% of the possible ratings are missing, because a user typically rates only a small portion of the movies. The (u,i) pairs for which r_{ui} is known are stored in the *training set* $\mathcal{K} = \{(u,i) \mid r_{ui} \text{ is known}\}$. Notice that $\mathcal{K}$ includes also the Probe set. Each user u is associated with a set of items denoted by $R(u)$, which contains all the items for which ratings by u are available. Likewise, $R(i)$ denotes the set of users who rated item i. Sometimes, we also use a set denoted by $N(u)$, which contains all items for which u provided a rating, even if the rating value is unknown. Thus, $N(u)$ extends $R(u)$ by also considering the ratings in the Qualifying set.

Models for the rating data are learned by fitting the previously observed ratings (training set). However, our goal is

Motivation

- Ensemble learning is currently among the most powerful ML methods
  - Note that ensemble learning is not one specific algorithm. It refers to any learning algorithm that combines several relatively weak predictors to a strong one.

Motivation

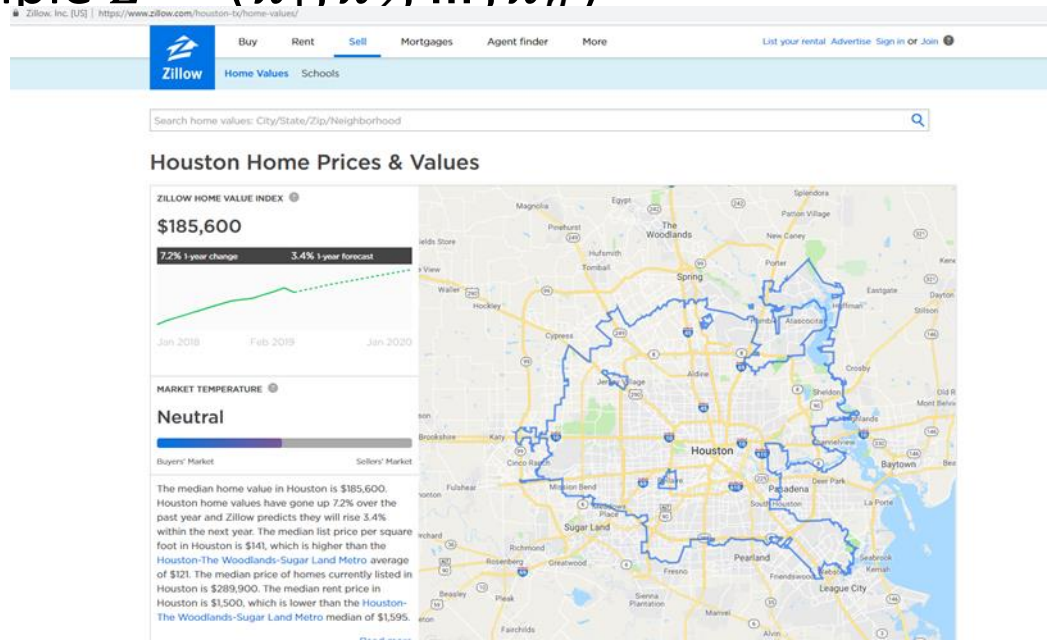
- Ensemble learning is currently among the most powerful ML methods
  - Note that ensemble learning is not one specific algorithm. It refers to any learning algorithm that combines several relatively weak predictors to a strong one.
- It is lucrative
  - Many of the winning solutions in ML competitions often involve ensemble learning methods.

Motivation

- Ensemble learning is currently among the most powerful ML methods
  - Note that ensemble learning is not one specific algorithm. It refers to any learning algorithm that combines several relatively weak predictors to a strong one.
- It is lucrative
  - Many of the winning solutions in ML competitions often involve ensemble learning methods.
  - You could become a millionaire!

More on bootstrapping

- Problem: What is the average housing price in Houston?
  - From Zillow, get a sample $L = (x_1, x_2, \ldots, x_n)$



pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

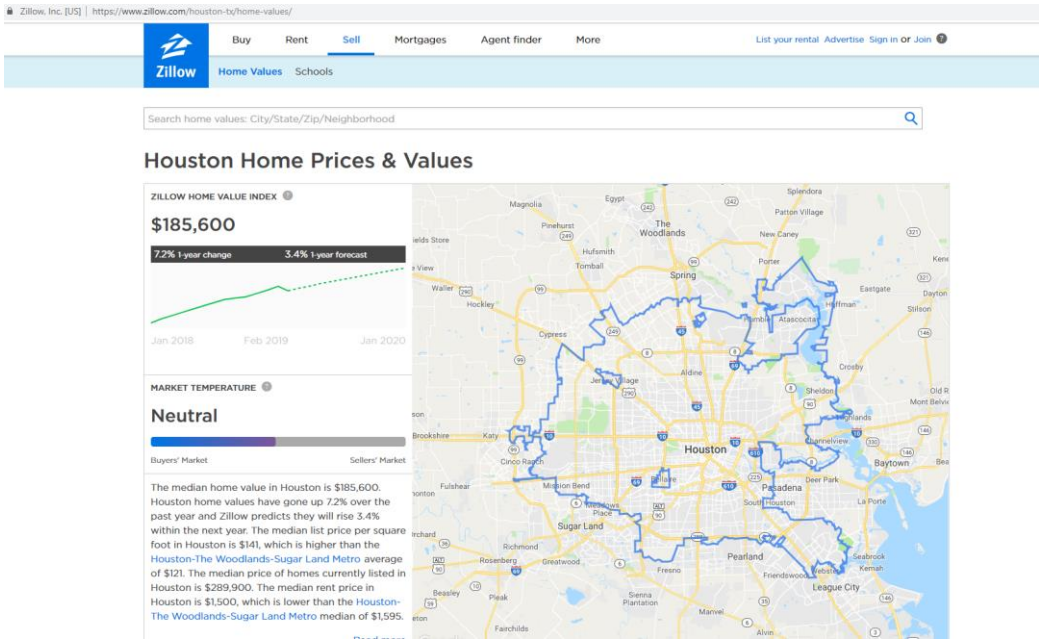
More on bootstrapping

- Problem: What is the average housing price in Houston?

pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

More on bootstrapping

- Problem: What is the average housing price in Houston?
  - From Zillow, get a sample $L = (x_1, x_2, \ldots, x_n)$



pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

More on bootstrapping

- Problem: What is the average housing price in Houston?
  - From Zillow, get a sample $L = (x_1, x_2, \dots, x_n)$
  - Calculate the average $u = \frac{1}{n}\sum_{i=1}^{n} x_i$

pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

More on bootstrapping

- Problem: What is the average housing price in Houston?
  - From Zillow, get a sample $L = (x_1, x_2, \dots, x_n)$
  - Calculate the average $u = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Question: How reliable is u? What is the standard deviation of u? What is the confidence interval?
- A harder question: What is the IQR of the house prices with a confidence interval?

pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

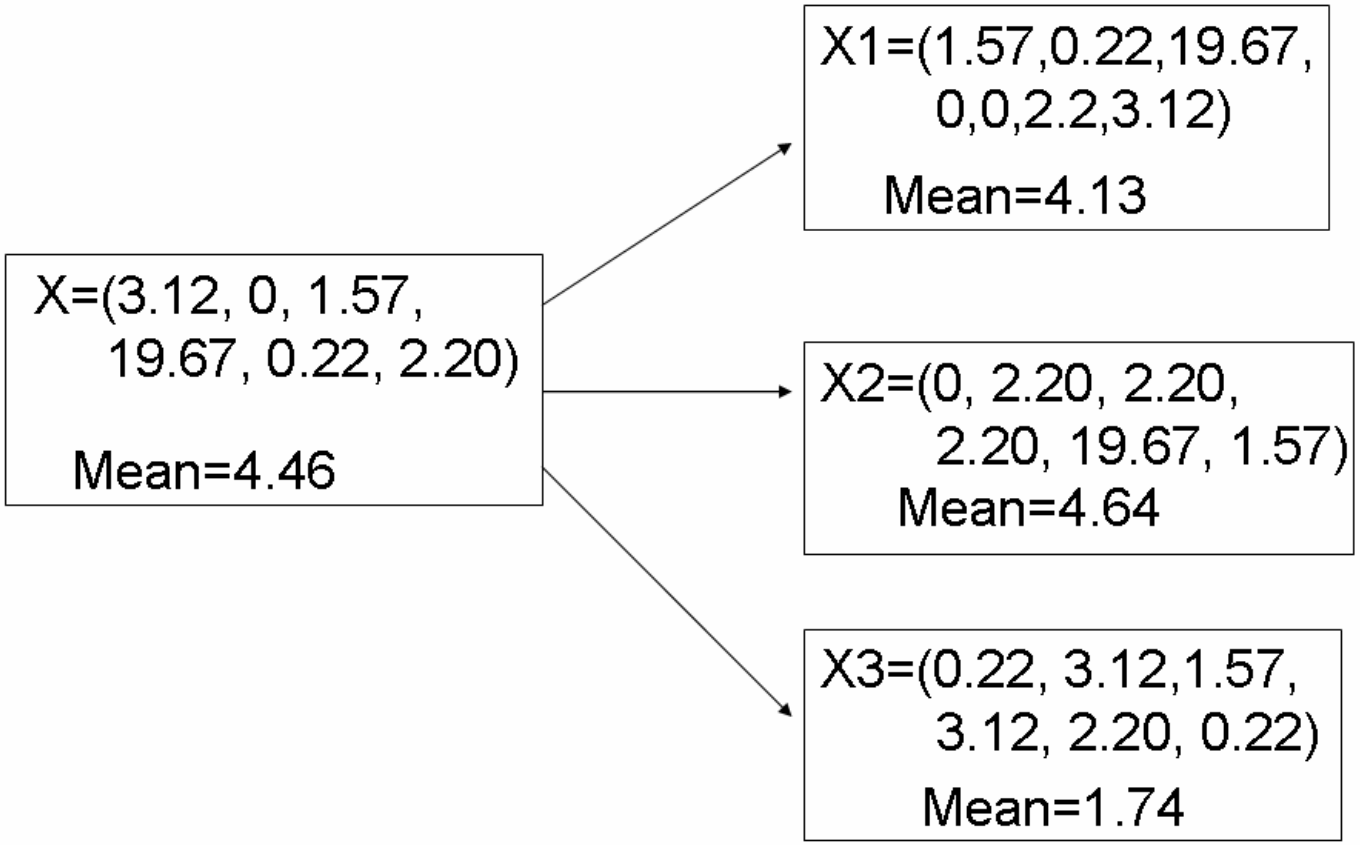
More on bootstrapping

- Problem: What is the average housing price in Houston?
  - From Zillow, get a sample $L = (x_1, x_2, \dots, x_n)$
  - Calculate the average $u = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Question: How reliable is u? What is the standard deviation of u? What is the confidence interval?
- A harder question: What is the IQR of the house prices with a confidence interval?

Bootstrapping!

pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

More on bootstrapping



pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

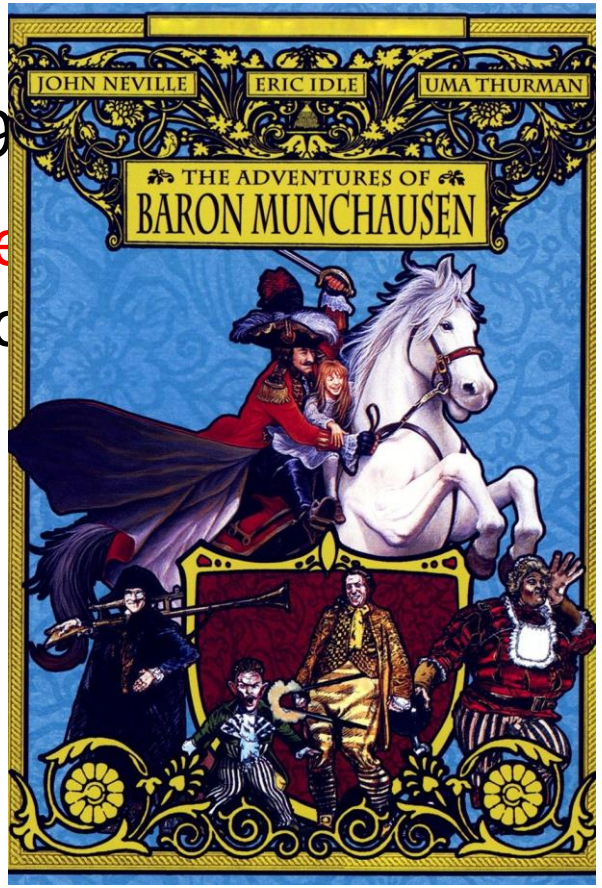
Bootstrapping

- Introduced by Bradley Efron in 1979.
- Named from the phrase "to pull oneself up by one's bootstraps" (from 'the Adventures of Baron Munchausen's')

pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

Bootstrapping

- Introduced by Bradley Efron in 1979
- Named from the phrase "to pull one bootstraps" (from 'the Adventures o Munchausen's')



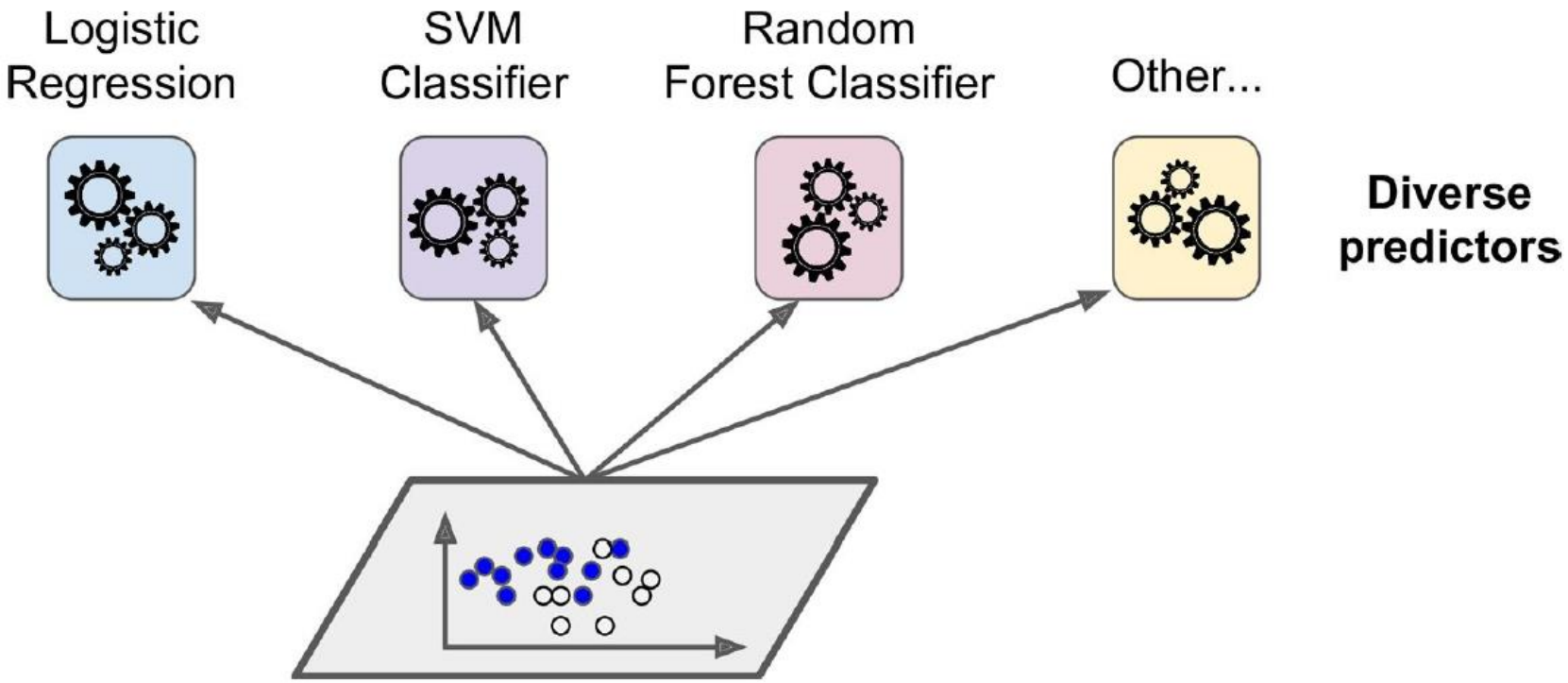
pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

Bootstrapping

- Introduced by Bradley Efron in 1979.
- Named from the phrase "to pull oneself up by one's bootstraps" (from 'the Adventures of Baron Munchausen's')
- Popularized in 1980s due to the introduction of computers in statistical practice.
- Well known as a method for estimating standard errors, bias and constructing confidence intervals for parameters.

pegasus.cc.ucf.edu/~xsu/CLASS/STA5703/notes11.pdf

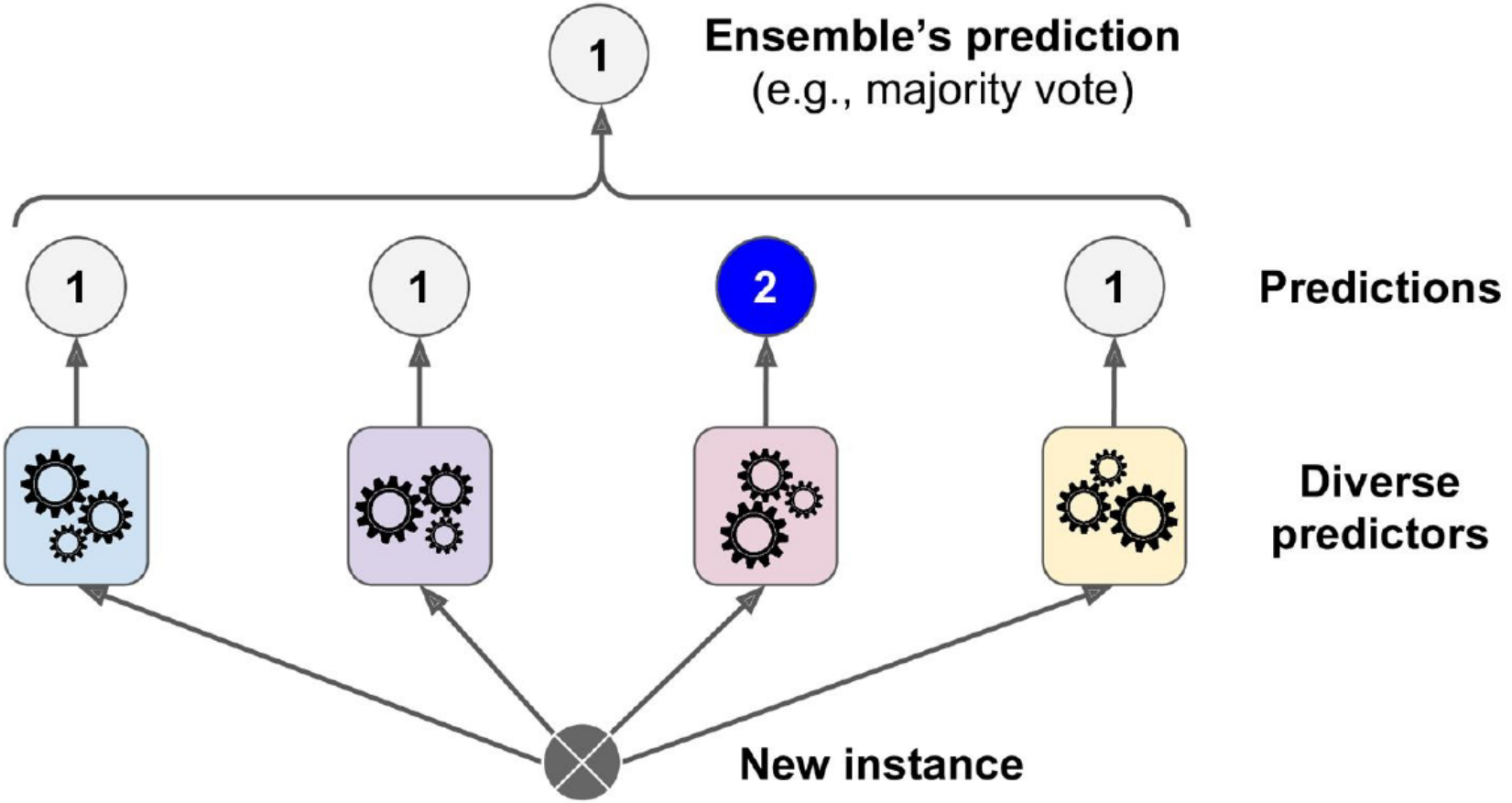
Illustration of ensemble learning



Training diverse classifiers

Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 182

Illustration of ensemble learning



Hard voting classifier predictions

Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 182

Hard voting

- Majority vote
- A.k.a, majority voting
- The predicted class for a particular sample is the class label that receives the most votes from all individual classifier.

Hard voting: example

If the prediction for a given sample is

- classifier 1 -> class 1
- classifier 2 -> class 1
- classifier 3 -> class 2

the hard voting classifier would classify the sample as “class 1” based on the majority class label.

scikit-learn.org/stable/modules/ensemble.html#weighted-average-probabilities-soft-voting

Soft voting

- Suppose each individual classifier predicts the probability of a sample belonging to a class

classifier	class 1	class 2	class 3
classifier 1	0.2	0.5	0.3
classifier 2	0.6	0.3	0.1
classifier 3	0.3	0.4	0.3
weighted average	0.37	0.4	0.23

the predicted class label would be 2, since it has the highest average probability.

scikit-learn.org/stable/modules/ensemble.html#weighted-average-probabilities-soft-voting

In-class quiz

- Suppose we want to classify our training instances into two categories (Class 1/Class 2), and we have trained 3 classifiers.
- Assume we were given a new instance, and the predictions from the 3 classifiers are:

classifier	class 1	class 2	Hard vote
classifier 1	0.45	0.55	2
classifier 2	0.45	0.55	2
classifier 3	0.90	0.10	1

- How would hard voting and soft voting classify this new instance?

- In general, soft voting achieves better performance than hard voting, because it gives more weight to highly confident votes.
- RandomForestClassifier in Scikit-learn uses soft voting. That is, the predicted class is the one with highest mean probability estimate across the trees.

https://goo.gl/CNFbZE

When does ensemble learning work best?

- When the individual predictors (or, models) are as independent from each other as possible

When does ensemble learning work best?

- When the individual predictors (or, models) are as independent from each other as possible
- Thus, we want them to be as diverse as possible.

When does ensemble learning work best?

- When the individual predictors (or, models) are as independent from each other as possible
- Thus, we want them to be as diverse as possible.
- In this case, the individual predictors tend to make very different types of uncorrelated errors

When does ensemble learning work best?

- When the individual predictors (or, models) are as independent from each other as possible
- Thus, we want them to be as diverse as possible.
- In this case, the individual predictors tend to make very different types of uncorrelated errors
- By averaging or aggregating, the prediction accuracy from ensemble learning can be improved.

Strategies to get diverse classifiers

- Use very different training algorithms

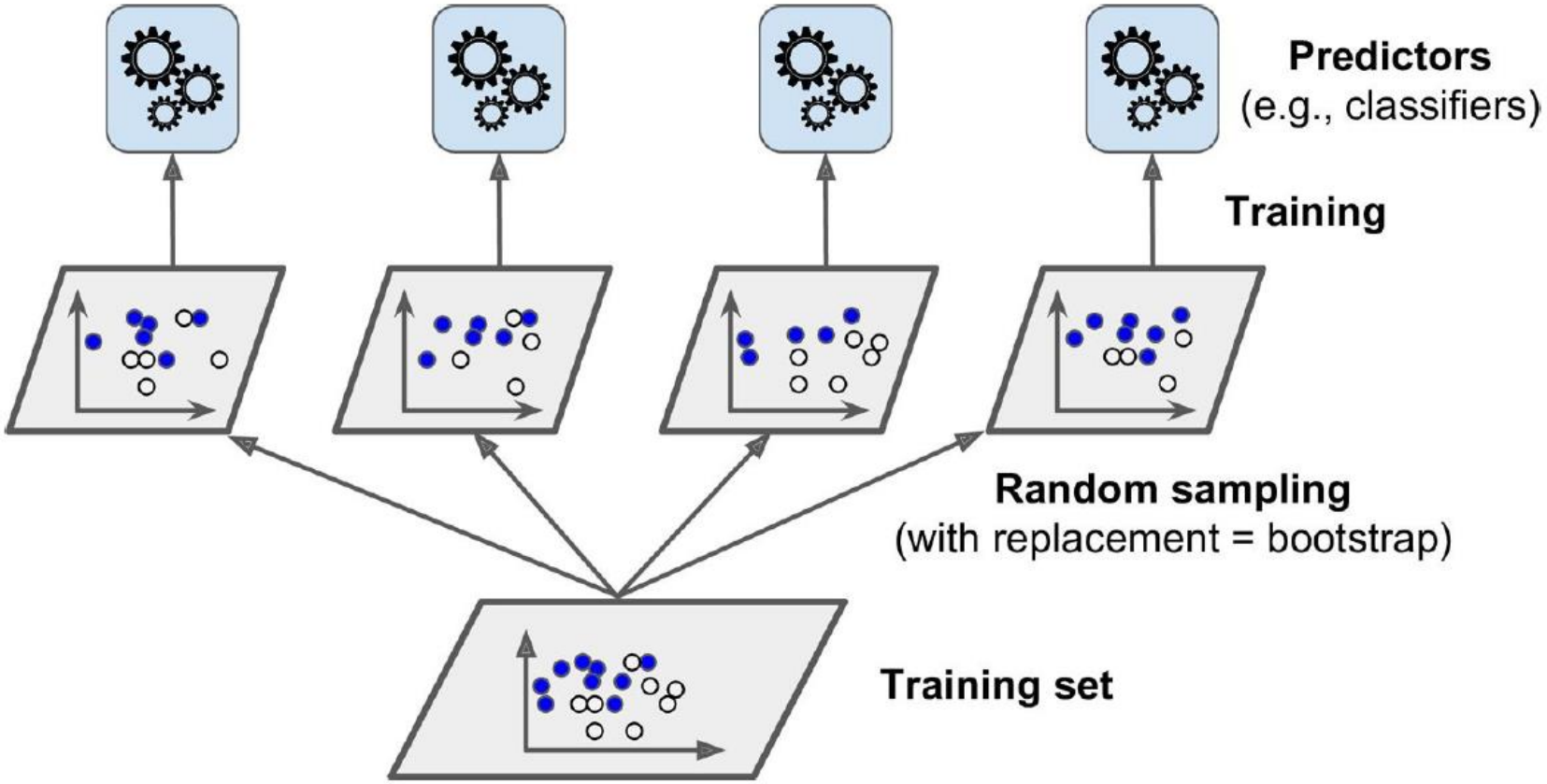
Strategies to get diverse classifiers

- Use very different training algorithms
- Use the same training algorithm, but train them on different random subsets of the training set

Strategies to get diverse classifiers

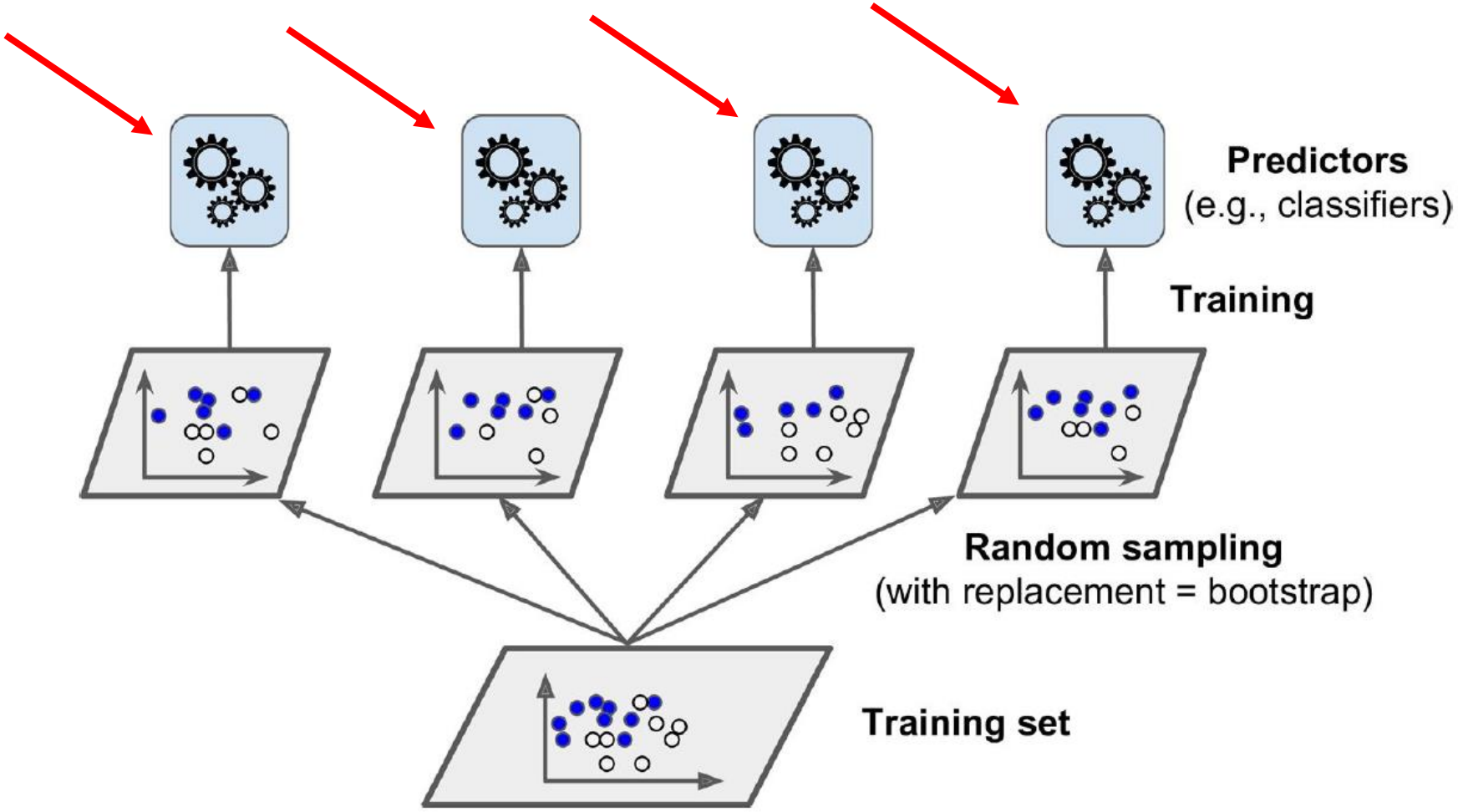
- Use very different training algorithms
- Use the same training algorithm, but train them on different random subsets of the training set
  - When random sampling is done with replacement, the ensemble learning is called bootstrap aggregating (or, bagging).
  - Pasting, when sampling is performed without replacement.

Ensemble learning via pasting/bagging



Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 185

Ensemble learning via pasting/bagging



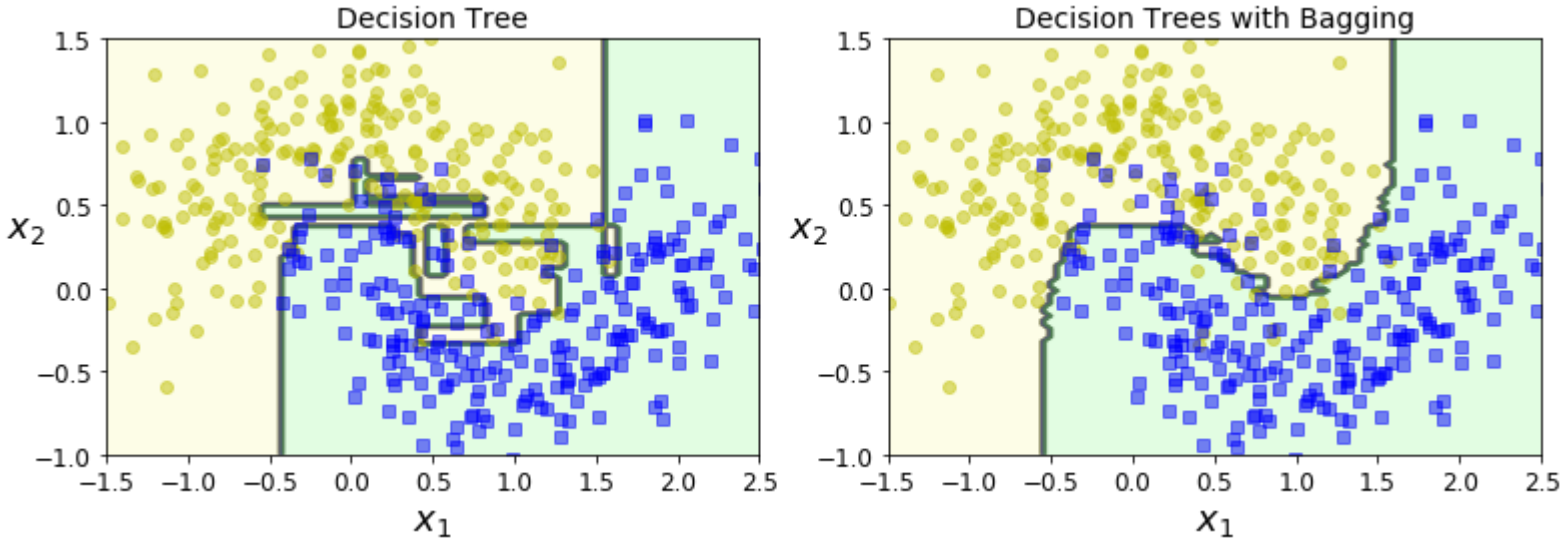
Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 185

How does RF work?

- Select the number of decision trees, M
- For each decision tree:
- create bootstrap samples of training instances
- grow a decision tree

  when splitting a node, instead of searching for the best feature among all features, just search for the best feature among a random subset of the features
- Average predictions from all M decision trees

http://scikit-learn.org/stable/modules/ensemble.html#forest



A single decision tree (left) vs. a bagging ensemble of 500 trees (right).

Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 187

Boosting

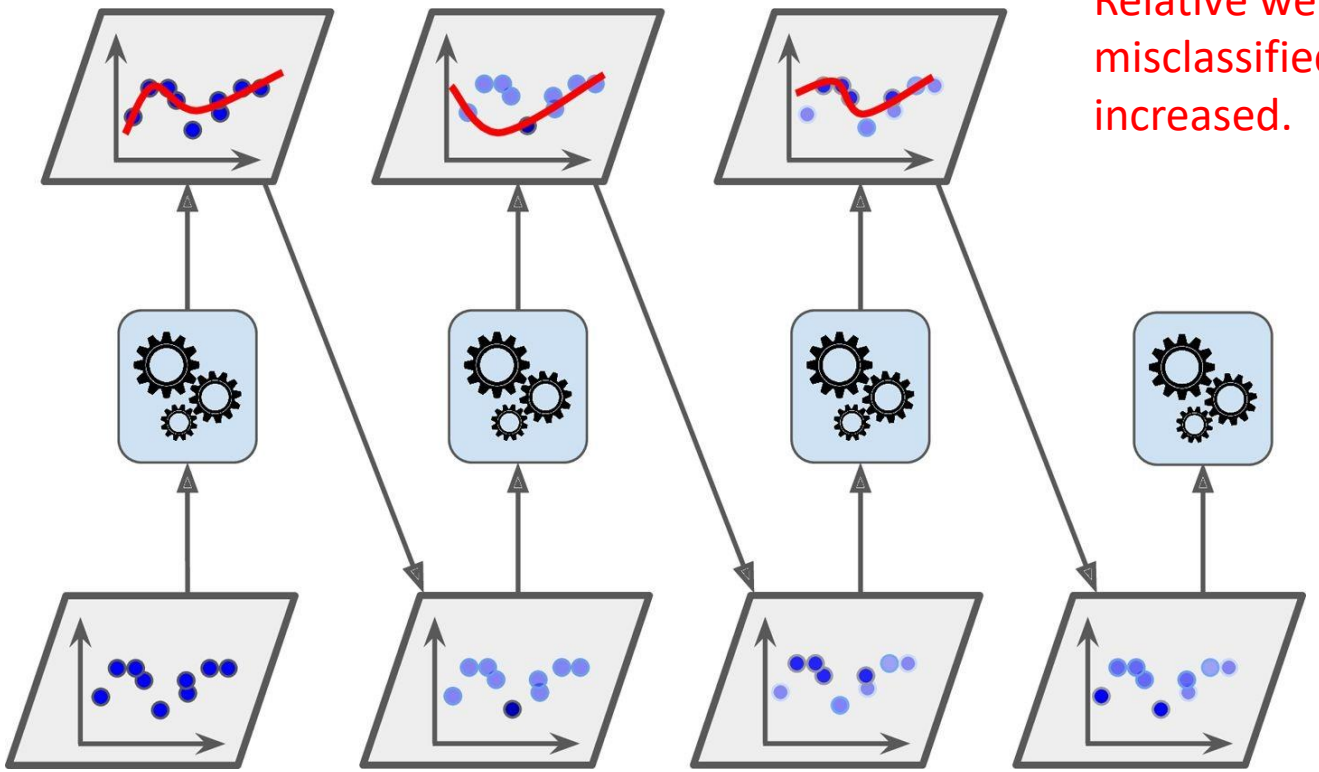
- Idea: train predictors sequentially, each trying to correct its predecessors
- Boosting encourages newer models to become better at instances handled incorrectly by earlier ones.

Boosting

- Idea: train predictors sequentially, each trying to correct its predecessors
- Boosting encourages newer models to become better at instances handled incorrectly by earlier ones.
- Two popular boosting methods:
  - AdaBoost (i.e., adaptive boosting)
  - Gradient Boosting

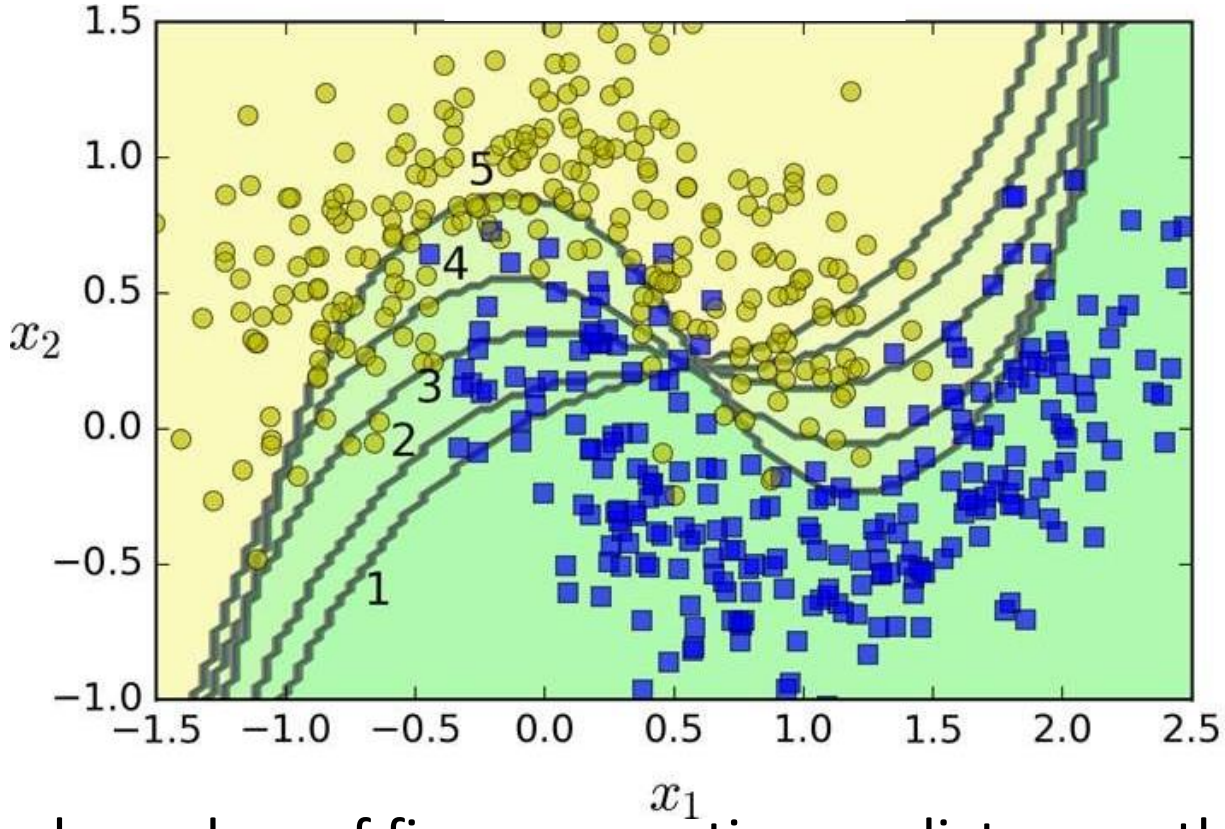
AdaBoost

- Basic idea: pay more attention to the training instances that the predecessor does not do well on.



Relative weight of misclassified instances is increased.

Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 192

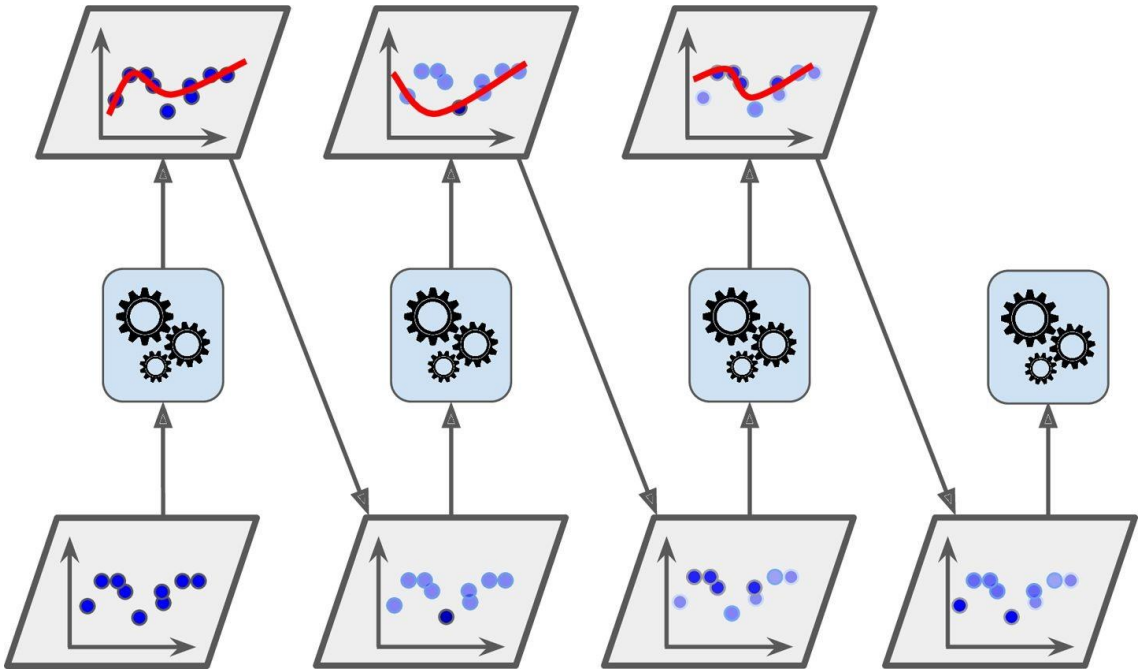


- Decision boundary of five consecutive predictors on the moons dataset (each predictor is a highly regularized SVM with RBF kernel)
- The first classifier gets many instances wrong, so their weights get boosted.
- The second classifier, therefore, does a better job on these instances

Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 193

AdaBoost

- Making predictions: Once all predictors are trained, the ensemble makes predictions much like bagging or pasting, except that predictors have different weights depending on their overall accuracy on the weighted training set.



Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 192

Gradient Boosting

- Like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each on correcting its predecessor.

Gradient Boosting

- Like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each on correcting its predecessor.
- Instead of tweaking the instance weights at every iteration like AdaBoost does, it fits the new predictor to the residual errors made by the previous predictor.

Gradient Boosting

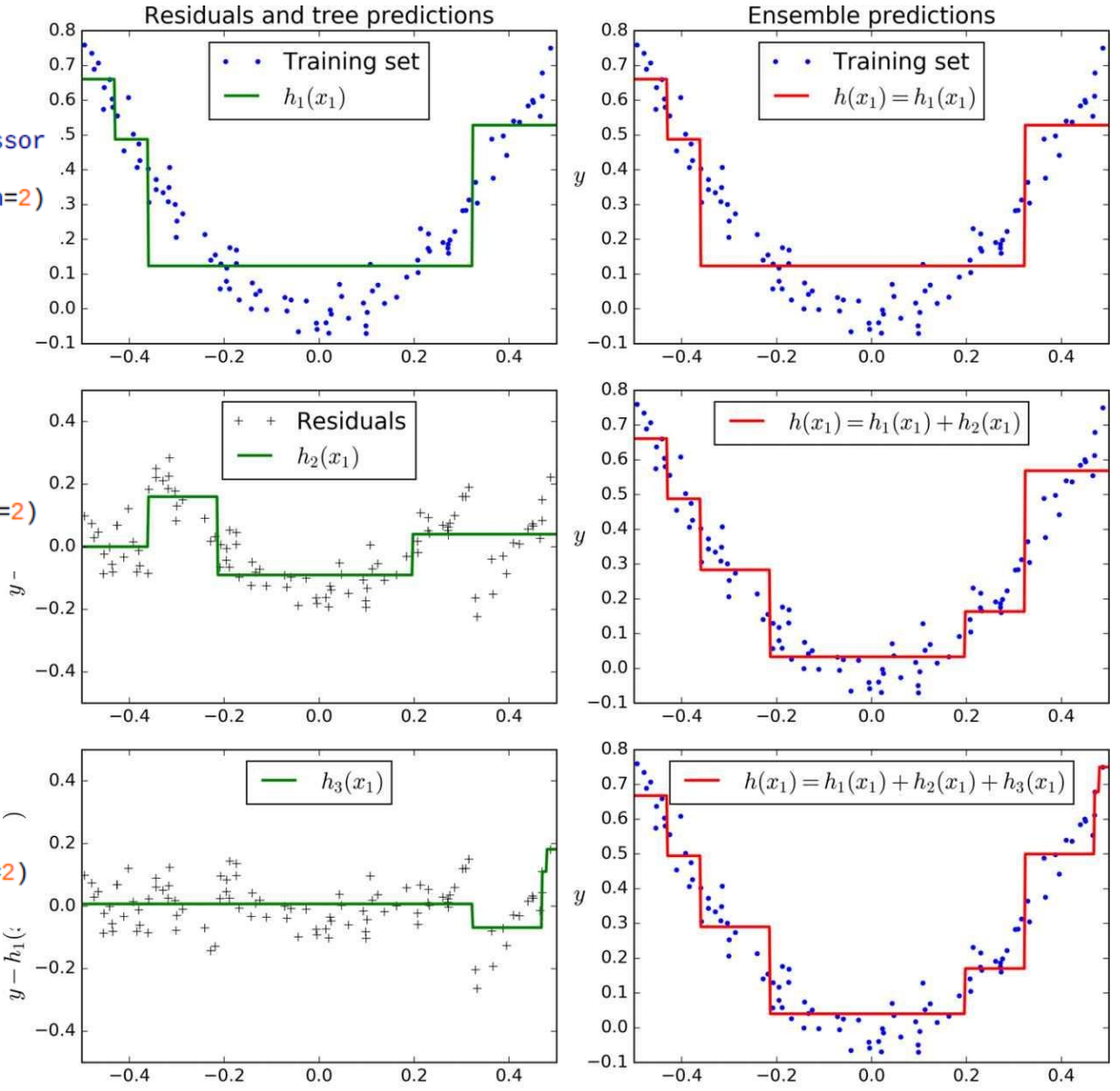
- Like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each on correcting its predecessor.
- Instead of tweaking the instance weights at every iteration like AdaBoost does, it fits the new predictor to the residual errors made by the previous predictor.
- If the base predictor is Decision Trees, it is called Gradient Tree Boosting.

```
from sklearn.tree import DecisionTreeRegressor

tree_reg1 = DecisionTreeRegressor(max_depth=2)
tree_reg1.fit(X, y)
```

```
y2 = y - tree_reg1.predict(X)
tree_reg2 = DecisionTreeRegressor(max_depth=2)
tree_reg2.fit(X, y2)
```

```
y3 = y2 - tree_reg2.predict(X)
tree_reg3 = DecisionTreeRegressor(max_depth=2)
tree_reg3.fit(X, y3)
```



Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 197

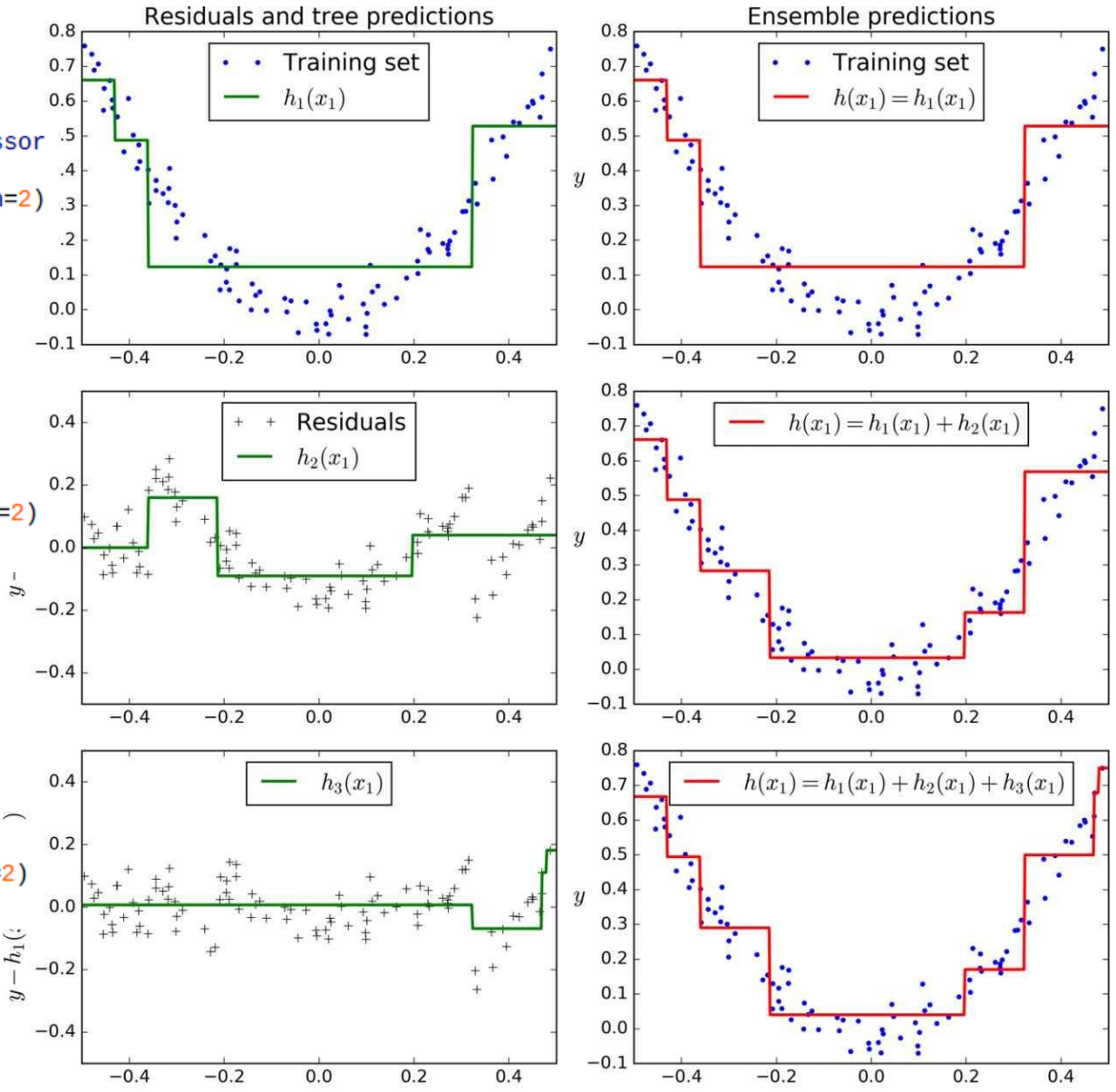
```
from sklearn.tree import DecisionTreeRegressor

tree_reg1 = DecisionTreeRegressor(max_depth=2)
tree_reg1.fit(X, y)
```

```
y2 = y - tree_reg1.predict(X)
tree_reg2 = DecisionTreeRegressor(max_depth=2)
tree_reg2.fit(X, y2)
```

To make predictions on a new instance, we simply add up the predictions from all trees.

```
y3 = y2 - tree_reg2.predict(X)
tree_reg3 = DecisionTreeRegressor(max_depth=2)
tree_reg3.fit(X, y3)
```



Aurelien Geron, 2017, Hands-on Machine Learning with Scikit-Learn & TensorFlow, pp 197

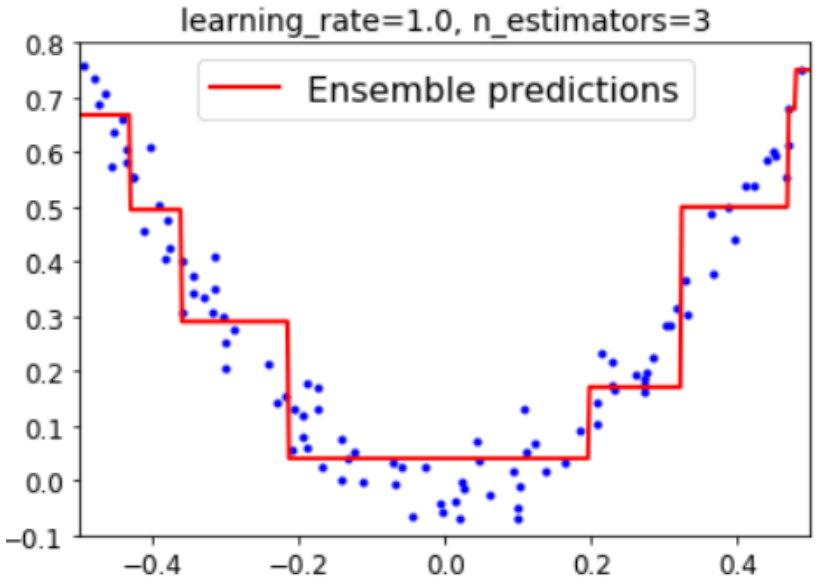
Implementation in Scikit-Learn

```
import numpy as np
np.random.seed(42)
X = np.random.rand(100, 1) - 0.5
y = 3*X[:, 0]**2 + 0.05 * np.random.randn(100)
```

GradientBoostingClassifier

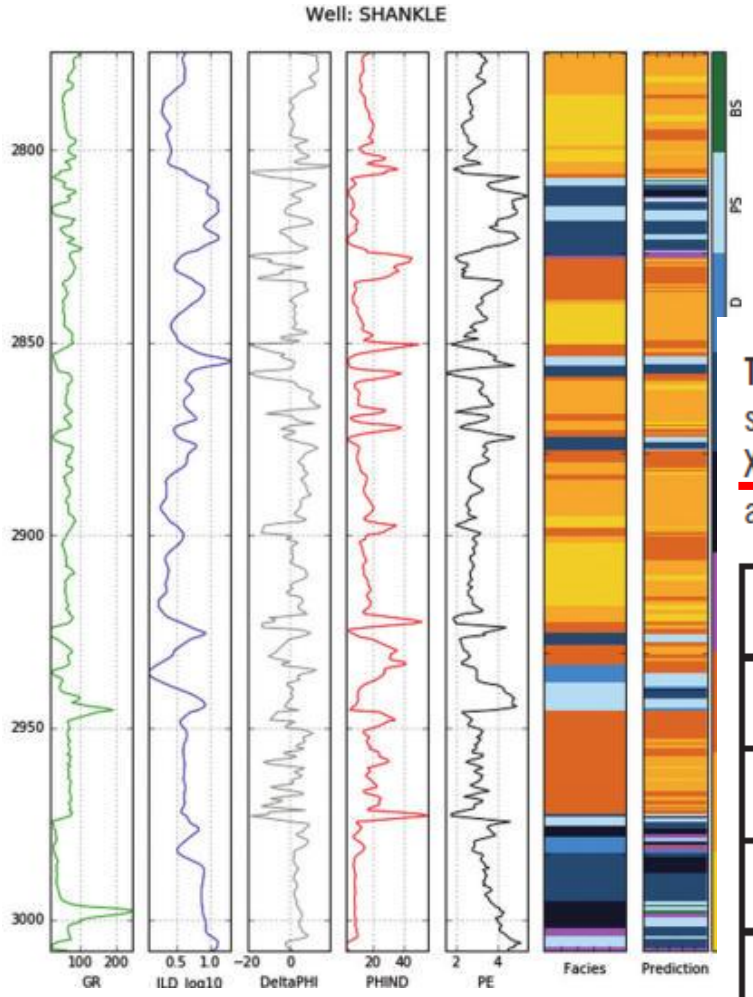
```
from sklearn.ensemble import GradientBoostingRegressor

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3, learning_rate=1.0, random_state=42)
gbrt.fit(X, y)
```



Variants of Gradient Boosting

Machine learning contest on predicting facies



Hall, 2016, TLE

- Received 300 solutions from 40 teams
- All of the top 10 solutions used ensemble learning.

Table 1. The top five teams at the end of the contest, with median accuracy scores from 100 realizations of their models. All of these entries implemented the XGBoost model in Python. You can see all of the teams, with deterministic scores, at github.com/seg/2016-ml-contest.

1	LA Team (Mosser & de la Fuente)	0.6388
2	PA Team (PetroAnalytix)	0.6250
3	ISPL (Bestagini, Tubaro & Lipari)	0.6231
4	esaTeam (Earth Science Analytics)	0.6225
5	SHandPR (Hall & Raghavan)	0.6200

dmlc

XGBoost eXtreme Gradient Boosting

Documentation | Resources | Installation | Release Notes | RoadMap

XGBoost is an optimized distributed gradient boosting library designed to be highly ***efficient***, ***flexible*** and ***portable***. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

https://github.com/dmlc/xgboost

Kaggle 1st place solution

- An ensemble of 6 models
  - 1 LightGBM (one type of gradient boosting)
  - 5 Neural Networks

- To learn more about his solution:
  - https://goo.gl/uaxgyP

GitHub, Inc. [US] https://github.com/Microsoft/LightGBM

README.md

LightGBM, Light Gradient Boosting Machine

build passing | build passing | docs passing | issues 44 open | license MIT | python 2.7, 3.4, 3.5, 3.6 | pypi package 2.1.0 | chat on gitter

Slack

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency
- Lower memory usage
- Better accuracy
- Parallel and GPU learning supported
- Capable of handling large-scale data

For more details, please refer to Features. Benefit from these advantages, LightGBM is being widely-used in many winning solutions of machine learning competitions.

Comparison experiments on public datasets show that LightGBM can outperform existing boosting frameworks on both efficiency and accuracy, with significantly lower memory consumption. What's more, the parallel experiments show that LightGBM can achieve a linear speed-up by using multiple machines for training in specific settings.

https://github.com/Microsoft/LightGBM

Week	Date	Topics	Comments
		Overview of syllabus	
1	01/15 Tues	Lecture: Introduction to Machine learning: applications	
	01/17 Thur	Lecture: Review of linear algebra	
2	01/22 Tues	Lab: Linear algebra in Python	Not graded
	01/24 Thur	Lecture: Introduction to optimization	
3	01/29 Tues	Lab: Gradient descent + Linear regression	Report due on 02/05 at 5:30 pm
	01/31 Thur	Lecture: Introduction to machine learning: concepts	
4	02/05 Tues	Lecture: Logistic regression	
	02/07 Thur	Lab: Logistic regression	Report due on 02/14 at 5:30 pm
5	02/12 Tues	Lecture: Support vector machine	
	02/14 Thur	Lab: Support vector machine	Report due on 02/21 at 5:30 pm
6	02/19 Tues	Lecture: Decision trees	
	02/21 Thur	Lab: Decision trees	Report due on 02/28 at 5:30 pm
7	02/26 Tues	Lecture: Random Forest	
	02/28 Thur	Lab: Random forest	Report due on 03/07 at 5:30 pm
8	03/05 Tues	Lecture: Ensemble learning	
	03/07 Thur	Lab: Ensemble learning	Reprot due on 03/19 at 5:30 pm
9	03/12 Tues	No class due to spring break	
	03/14 Thur	No class due to spring break	
10	03/19 Tues	Review & Recap	
	03/21 Thur	Exam	
11	03/26 Tues	Lecture: Clustering	
	03/28 Thur	Lab: Clustering	Report due on 04/04 at 5:30 pm
12	04/02 Tues	Lecture: Introduction to TensorFlow	
	04/04 Thur	Lab: TensorFlow	Not graded
13	04/09 Tues	Lecture: Introduction to neural networks 1	
	04/11 Thur	Lecture: Introduction to neural networks 2	
14	04/16 Tues	Lab: Deep learning	Report due on 04/23 at 5:30pm
	04/18 Thur	Lecture: Convolutional neural networks 1	
15	04/23 Tues	Guest lecture: Convolutional neural networks 2	
	04/25 Thur	Lab: CNN (optional)	Report due on 05/02 at 5:30 pm
16	04/30 Tues	final presentation??	
	05/02 Thur	final presentation??	
Note	28 class meetings		04/29 last day of class

GEOL 4397 Machine learning competition

Task

The prediction errors on validation dataset that I obtained are: 0.10319599537928381 for standard **AdaBoostClassifier**, 0.11166730843280703 for **AdaBoostClassifier** with **balanced class_weight**, and 0.1058914131690411 6 for **GradientBoostingClassifier**. So, the lowest error that I achieved is 0.10319599537928381, which corresponds to a prediction accuracy of 1-0.10319599537928381 = 0.8968.

Your task: do whatever you can to come up with a predictor (or synonymously, classifier) with better predication accuracy. By 'better', I mean better than 0.8968. **(100 points)**

HINT: Feel free to try anything you can think of. For example, it is up to you to choose between **AdaBoostClassifier** and **GradientBoostingClassifier**. Or, if you like, you can even create your own ensemble by combining, say, support vector machine, logistic regression and decision trees. Also, feel free to try different values for the hyperparameters such as **max_depth**, **n_estimators**, **learning_rate**, and **class_weight** (not available for Gradient Boosting).

Bonus

If you achieve a prediction accuracy of **91% or above**, you get **10 bonus points**.

If you achieve a prediction accuracy of **92% or above**, you will get a **$20 gift card** from a merchant of your choice. (Limited to the first five submissions)

If you achieve a prediction accuracy of **93% or above**, I will buy you **a dinner at Eric's**.