Avisos!

11-13/Nov.: Aula Aprendizado por Reforço com a Profa. Esther Colombini.
18/Nov.: Não teremos aula.
20/Nov.: Feriado.
25/Nov.: Aula.
27/Nov.: **Apresentação dos trabalhos**.

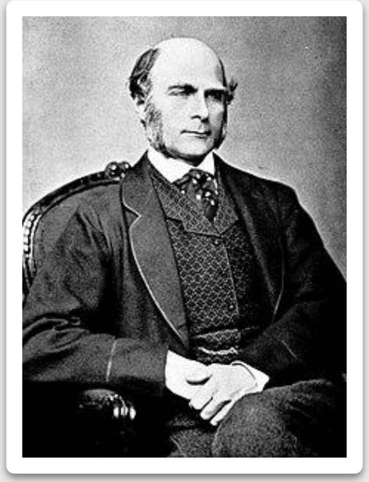
Ensemble Learning

Machine Learning

Prof. Sandra Avila

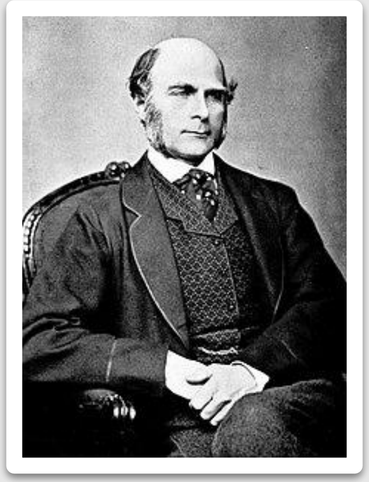
Institute of Computing (IC/Unicamp)

MC886/MO444, November 6, 2019



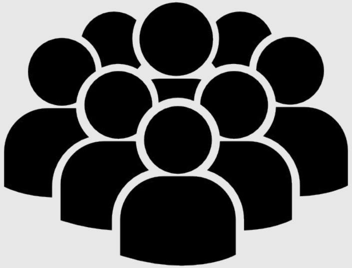
Francis Galton
(1822-1909)

Animal's weight?

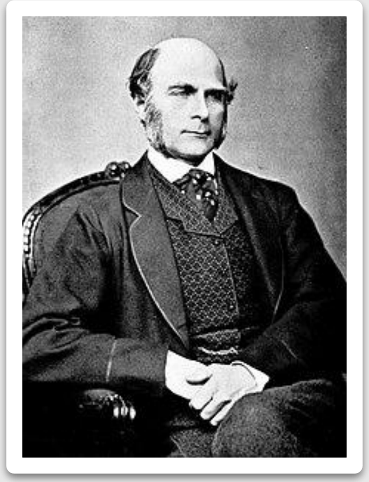


Francis Galton
(1822-1909)

Animal's weight?



~800 people
1,197 kg



Francis Galton
(1822-1909)

Animal's weight?

~800 people
1,197 kg

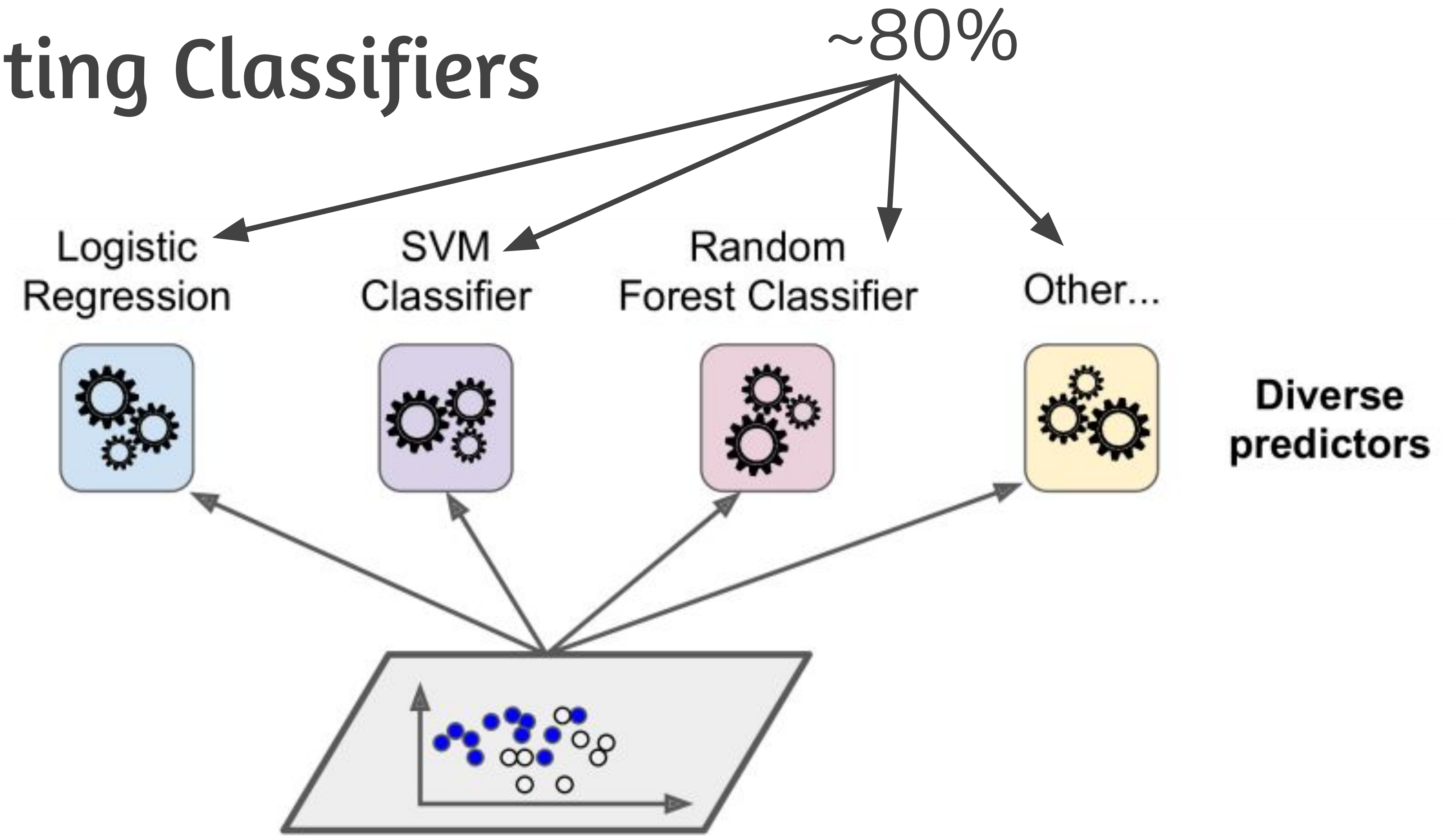
1,207 kg

Wisdom of the Crowd

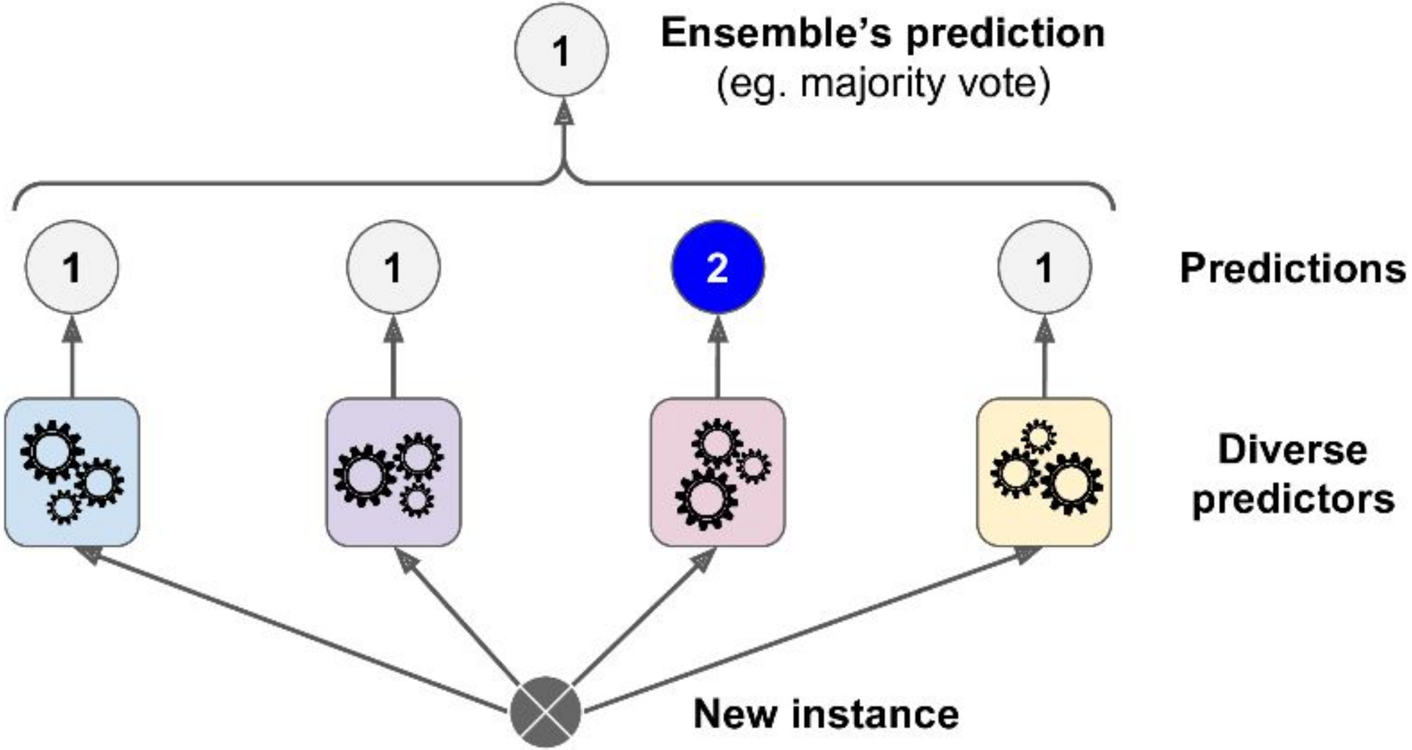
Ensemble Learning

- Multiple learning algorithms **to obtain better predictive performance** than could be obtained from any learning algorithms individually.

Voting Classifiers

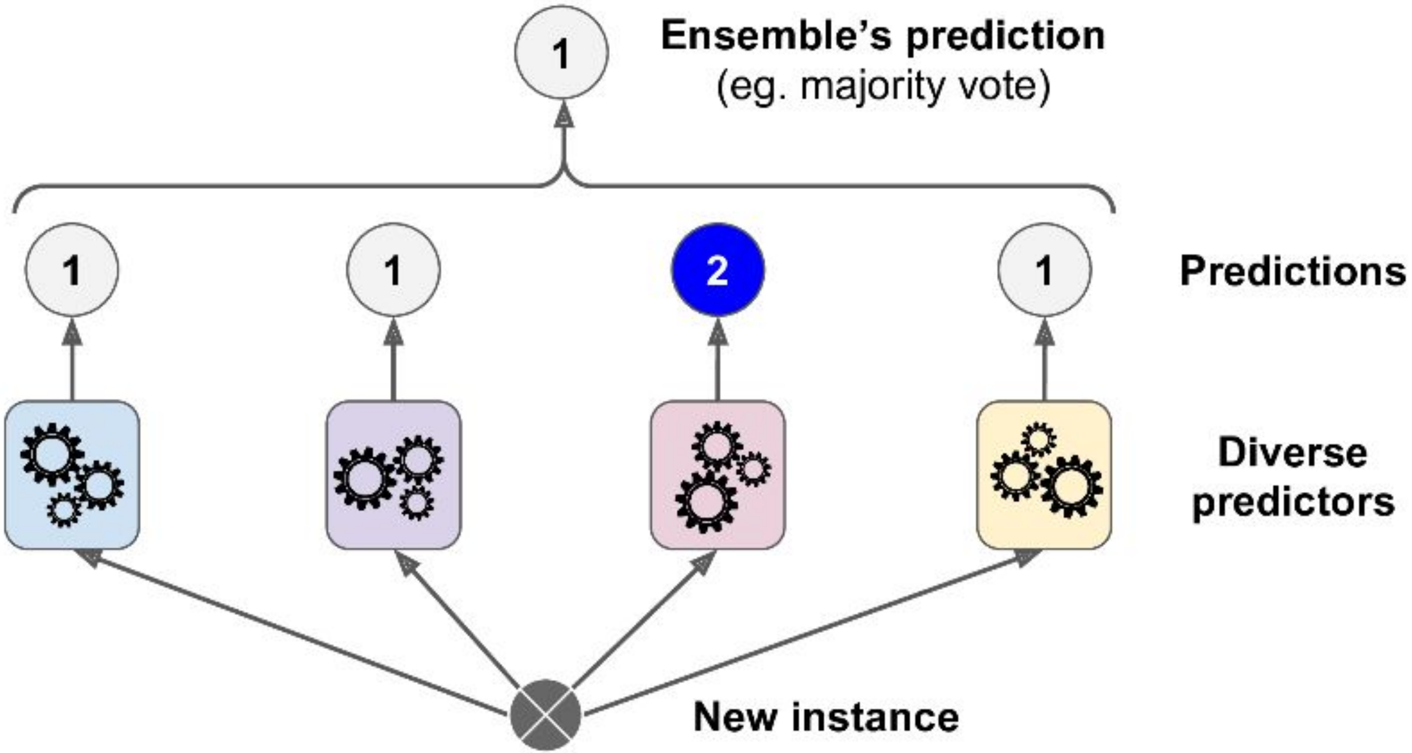


Voting Classifiers



Voting Classifiers

Hard/Soft voting classifier



Voting Classifiers

- Voting classifier **often achieves a higher accuracy** **than the best classifier** in the ensemble.

Voting Classifiers

- Voting classifier **often achieves a higher accuracy than the best classifier** in the ensemble.

- Even if each classifier is a **weak learner**, the ensemble can still be a **strong learner**, provided there are a sufficient number of weak learners and they are sufficiently diverse.

Voting Classifiers

- Ensemble methods work best when the predictors are as **independent** from one another as possible.

Voting Classifiers

- Ensemble methods work best when the predictors are as **independent** from one another as possible.

- One way to get diverse classifiers is to train them using **very different algorithms**: this increases the chance that they will make very different types of errors, improving the ensemble's accuracy.

Voting Classifiers

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()
voting_clf = VotingClassifier(
        estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
                     voting='hard'
    )
voting_clf.fit(X_train, y_train)
```

Voting Classifiers

```
from sklearn.ensemble import RandomForestClassifi
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()
voting_clf = VotingClassifier(
        estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
                     voting='hard'
    )
voting_clf.fit(X_train, y_train)
```

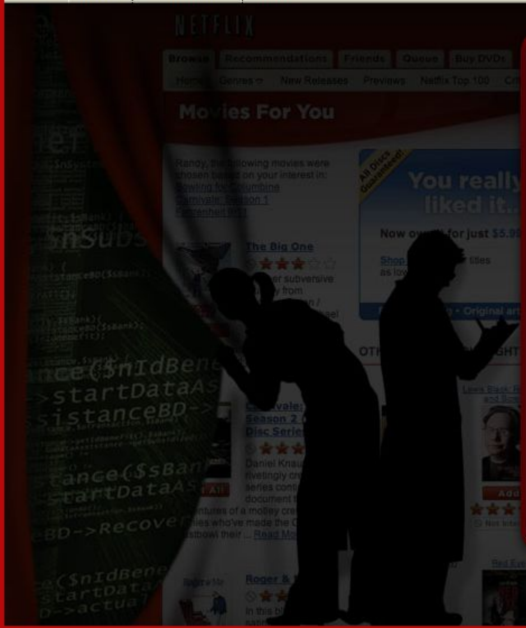
```
LogisticRegression 0.864
RandomForestClassifier 0.896
SVC 0.888
VotingClassifier 0.904
```

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update



Congratulations!

The 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.

On September 21, 2009 we awarded the $1M Grand Prize to team “BellKor’s Pragmatic Chaos”. Read about their algorithm, checkout team scores on the Leaderboard, and join the discussions on the Forum.

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.

FAQ | Forum | Netflix Home

Today's Agenda

- Ensemble Methods
  - Bagging (and Pasting)
  - Boosting
  - Stacking

Bagging & Pasting

Bagging and Pasting

- Use **the same training algorithm** for every predictor, but to train them on **different random subsets** of the training set.

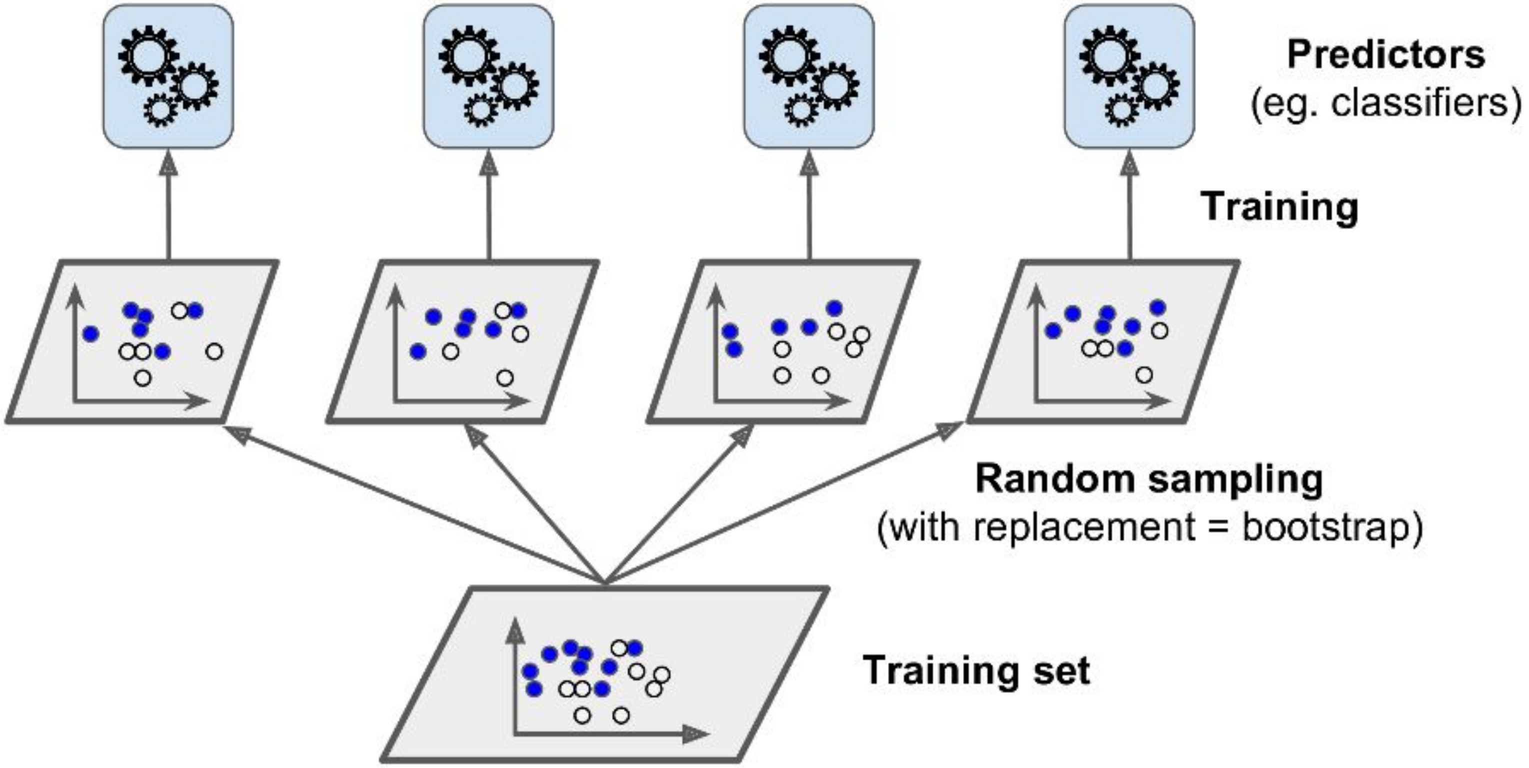
“Bagging Predictors”, L. Breiman (1996): http://goo.gl/o42tml

Bagging and Pasting

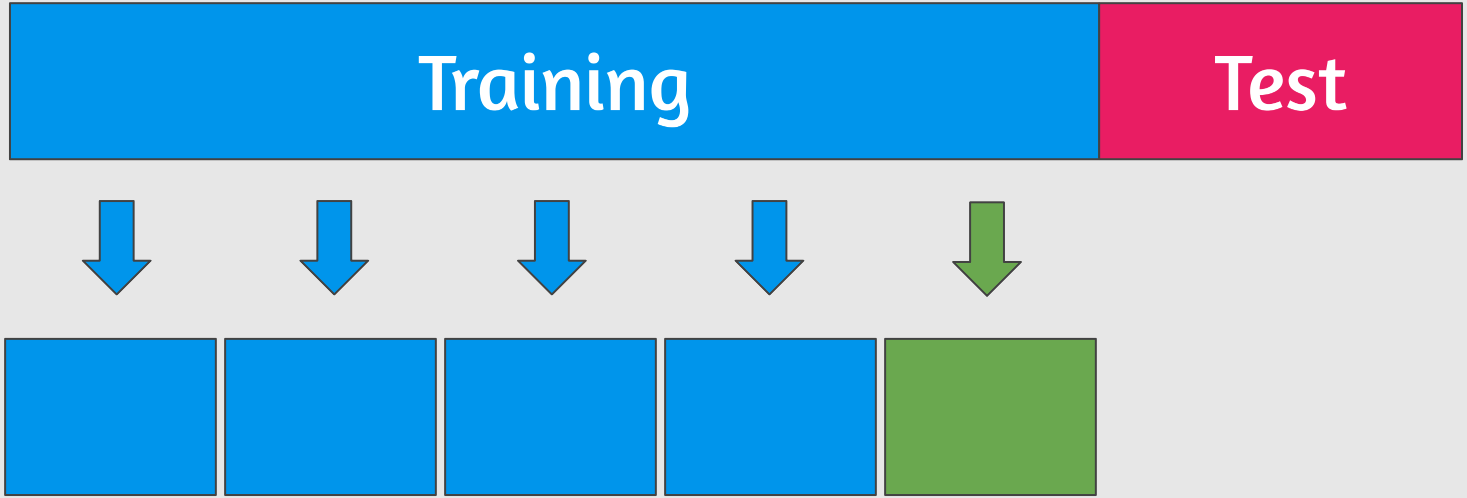
- Use **the same training algorithm** for every predictor, but to train them on **different random subsets** of the training set.
- **Bagging** (short for Bootstrap Aggregating): sampling is performed **with** replacement.
- **Pasting**: sampling is performed **without** replacement.

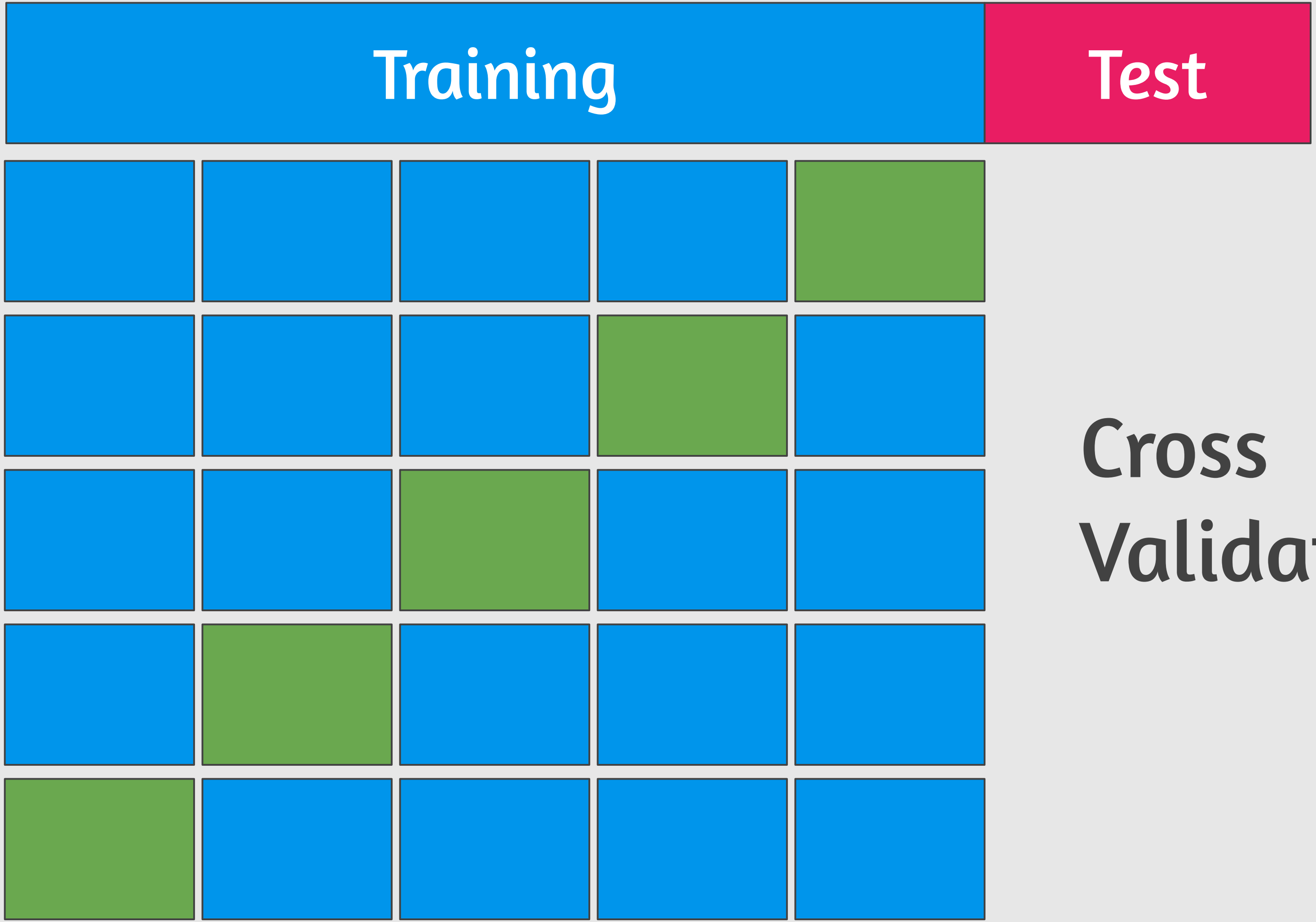
"Bagging Predictors", L. Breiman (1996): http://goo.gl/o42tml

Bagging and Pasting

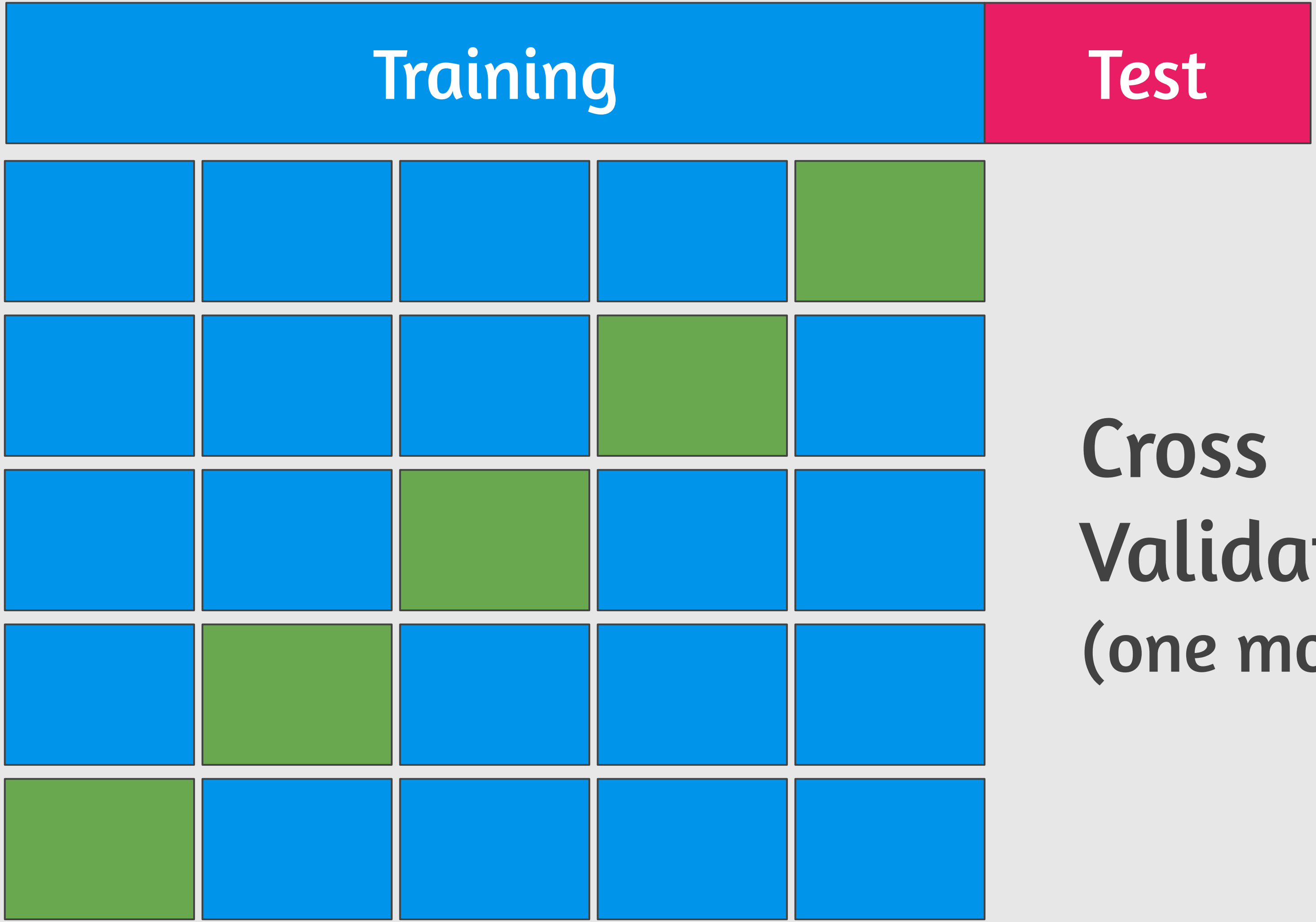


Bagging vs. Cross Validation

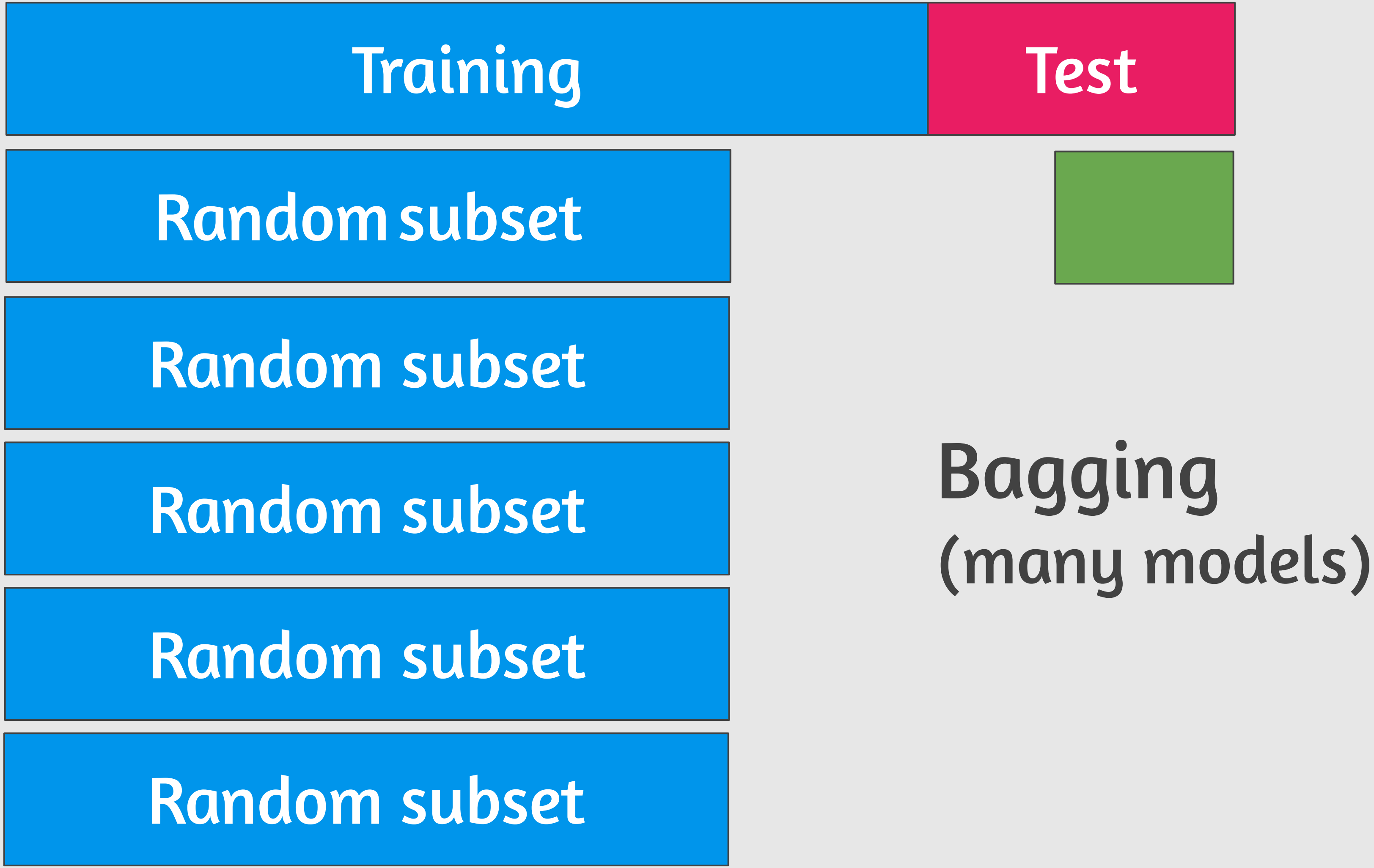




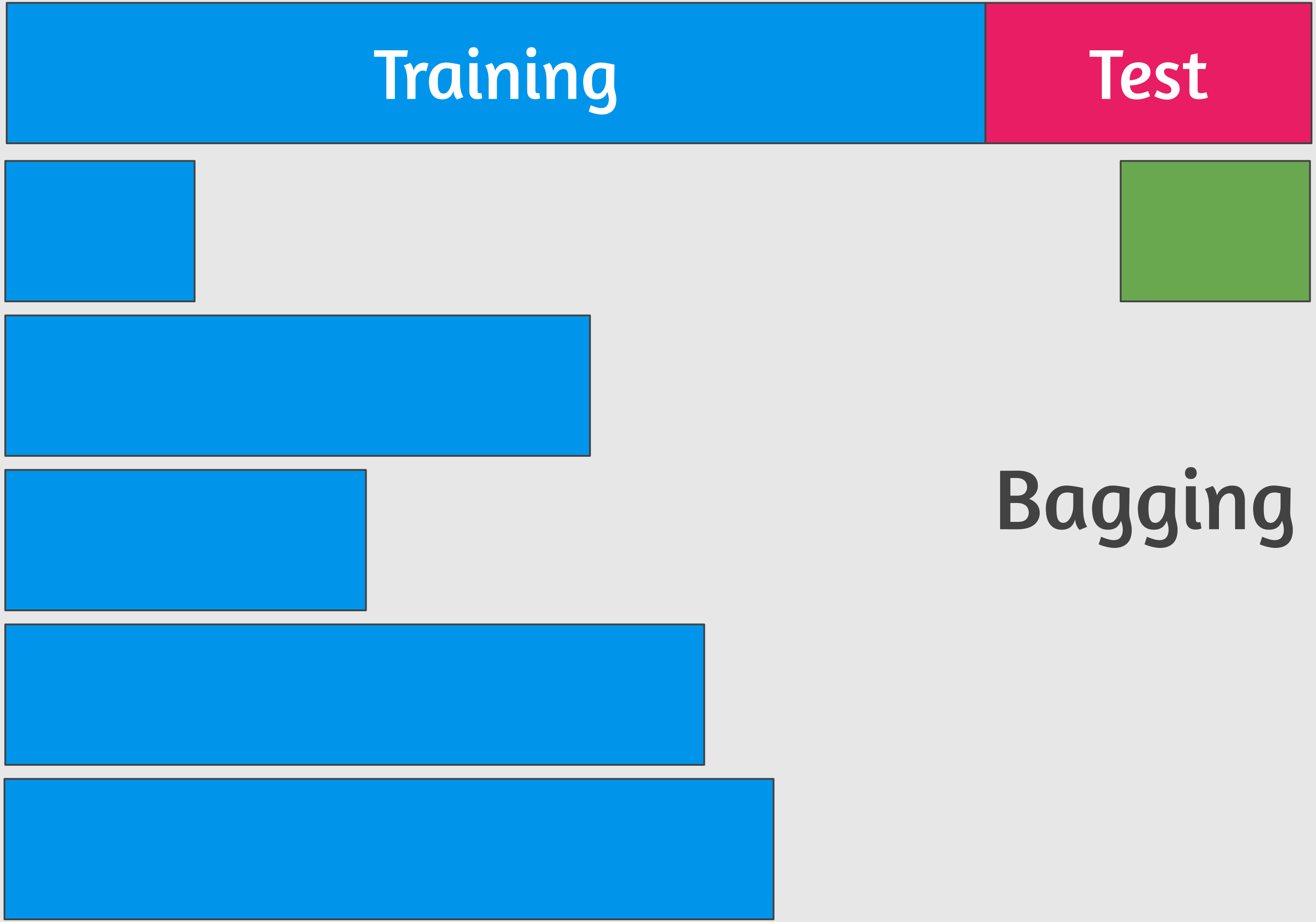
Cross Validation



Cross Validation (one model)


Training
Test
Random subset
Random subset
Random subset
Random subset
Random subset
Bagging
(many models)

Bagging



Bagging and Pasting

- Once all predictors are trained, the ensemble can make a prediction for a new instance by simply **aggregating the predictions of all predictors**.

- **Bagging and Pasting scale very well.**

Bagging and Pasting

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
bag_clf = BaggingClassifier(
        DecisionTreeClassifier(), n_estimators=500,
        max_samples=100, bootstrap=True, n_jobs=-1
    )
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
```

Bagging and Pasting

- Random Patches Ensemble method: sampling **both** training instances and features.
- This is particularly useful when dealing with high-dimensional inputs.

"Ensembles on Random Patches", G. Louppe and P. Geurts (2012): http://goo.gl/B2EcM2

Today's Agenda

- Ensemble Methods
  - Bagging (and Pasting)
  - **Boosting**
  - Stacking

Boosting

Boosting

- The general idea of most boosting methods is **to train predictors sequentially**, each trying to correct its predecessor.

Boosting

- The general idea of most boosting methods is **to train predictors sequentially**, each trying to correct its predecessor.
- Most popular: AdaBoost and Gradient Boost.

AdaBoost [Freund and Schapire, 1997]

- One way for a new predictor to correct its predecessor is to pay a bit **more attention** to the training instances that **the predecessor underfitted**.

“A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”, Y. Freund and R. Schapire (1997): http://goo.gl/OIduRW

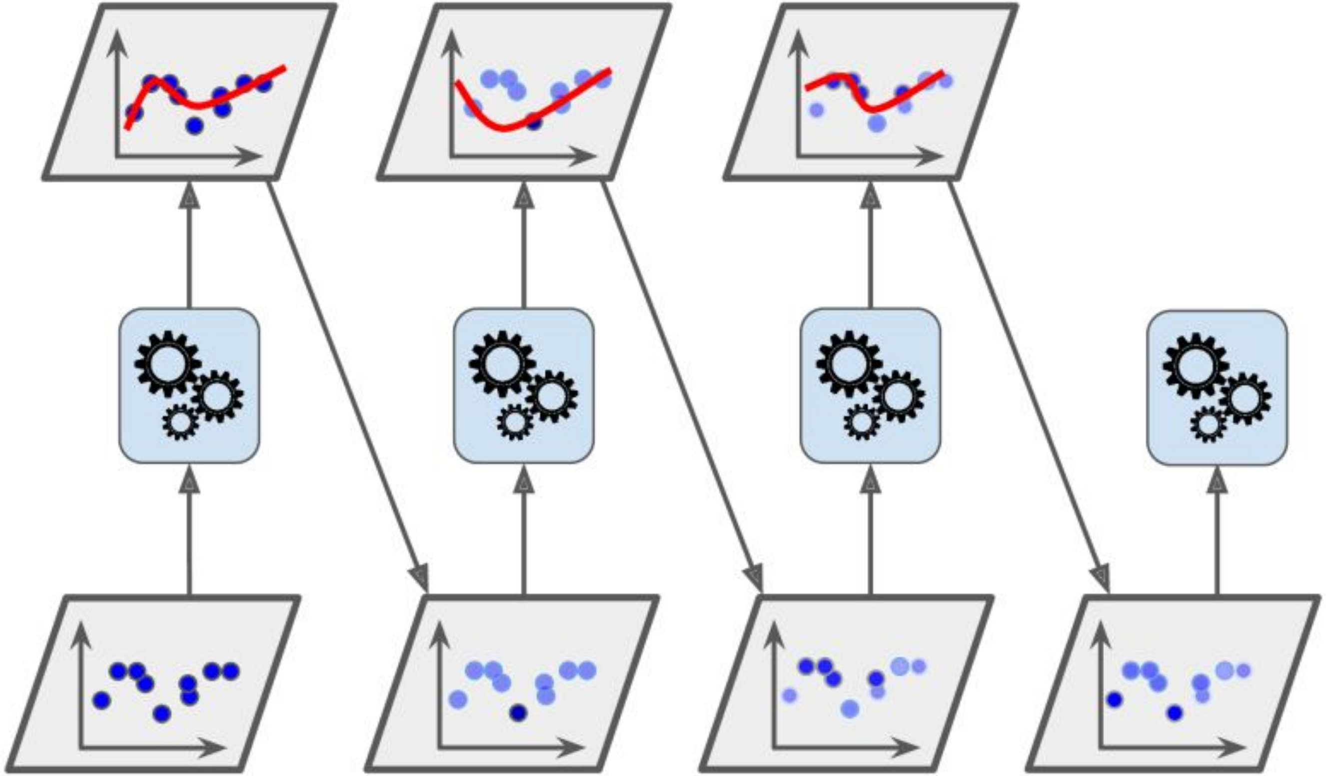
AdaBoost [Freund and Schapire, 1997]

- One way for a new predictor to correct its predecessor is to pay a bit **more attention** to the training instances that **the predecessor underfitted**.

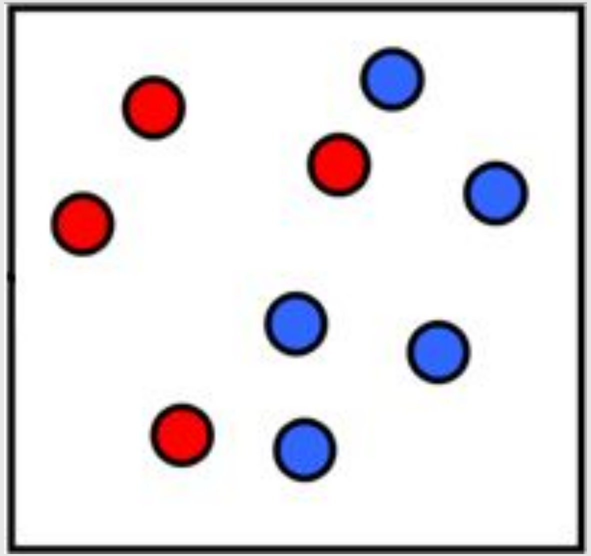
- This results in new predictors focusing more and more on **the hard cases**.

"A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", Y. Freund and R. Schapire (1997): http://goo.gl/OIduRW

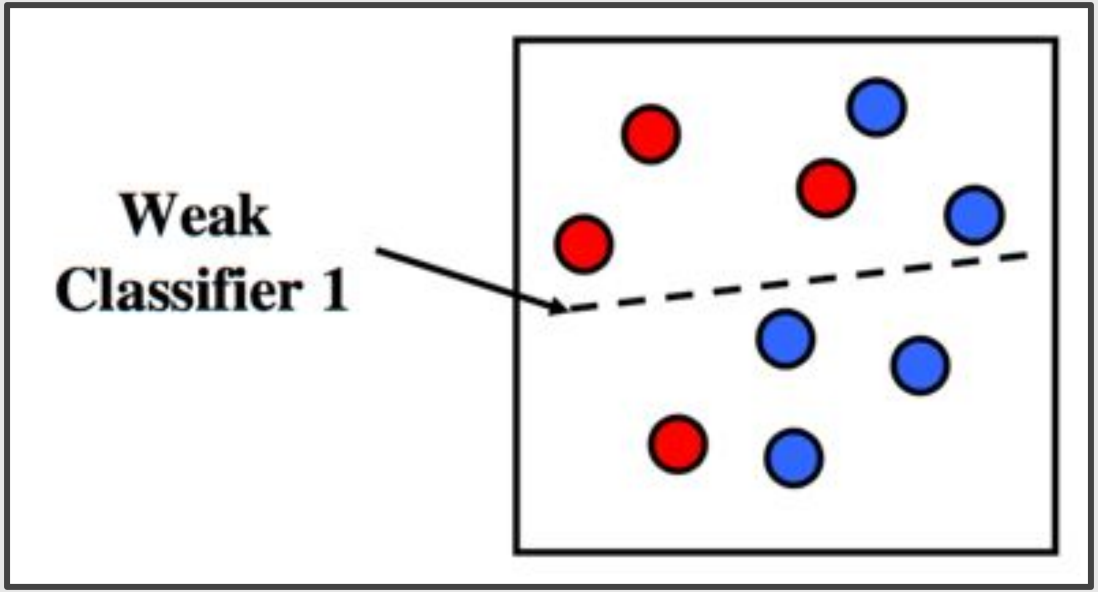
AdaBoost



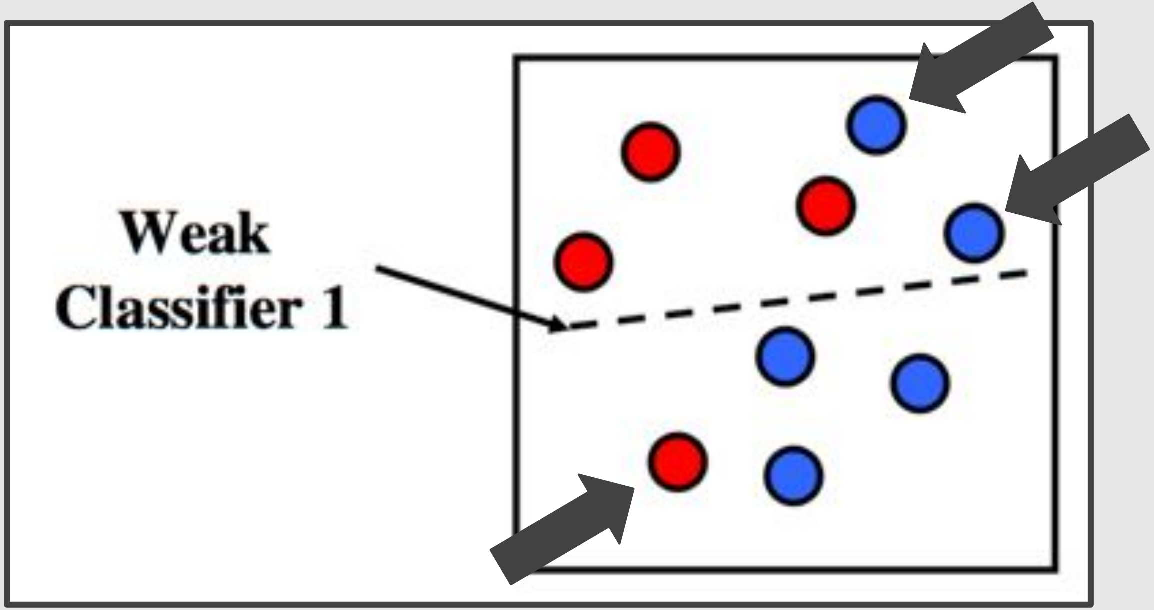
AdaBoost



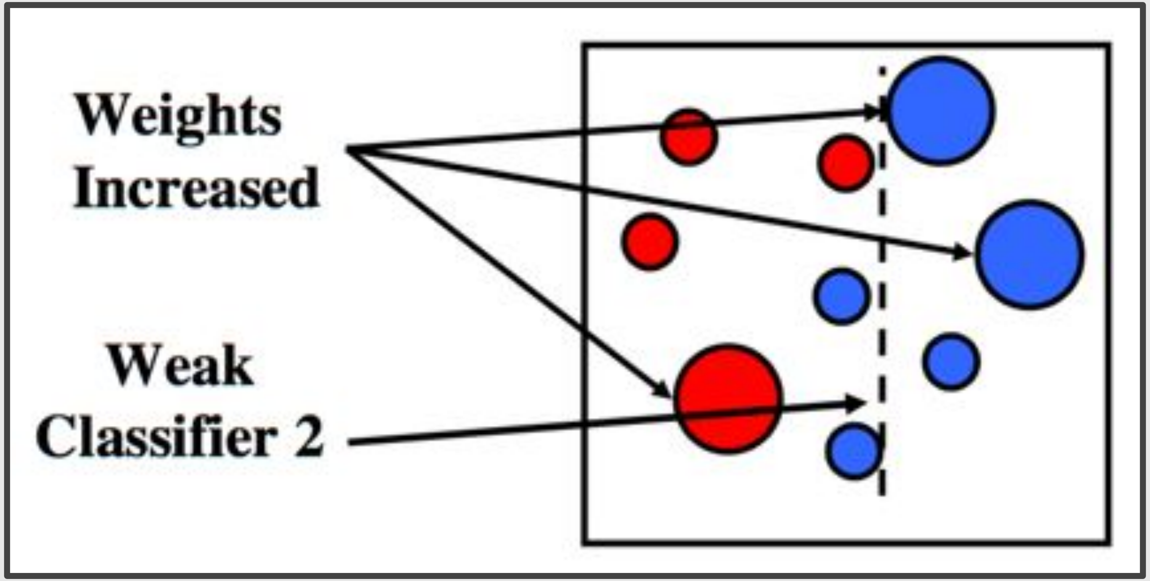
AdaBoost



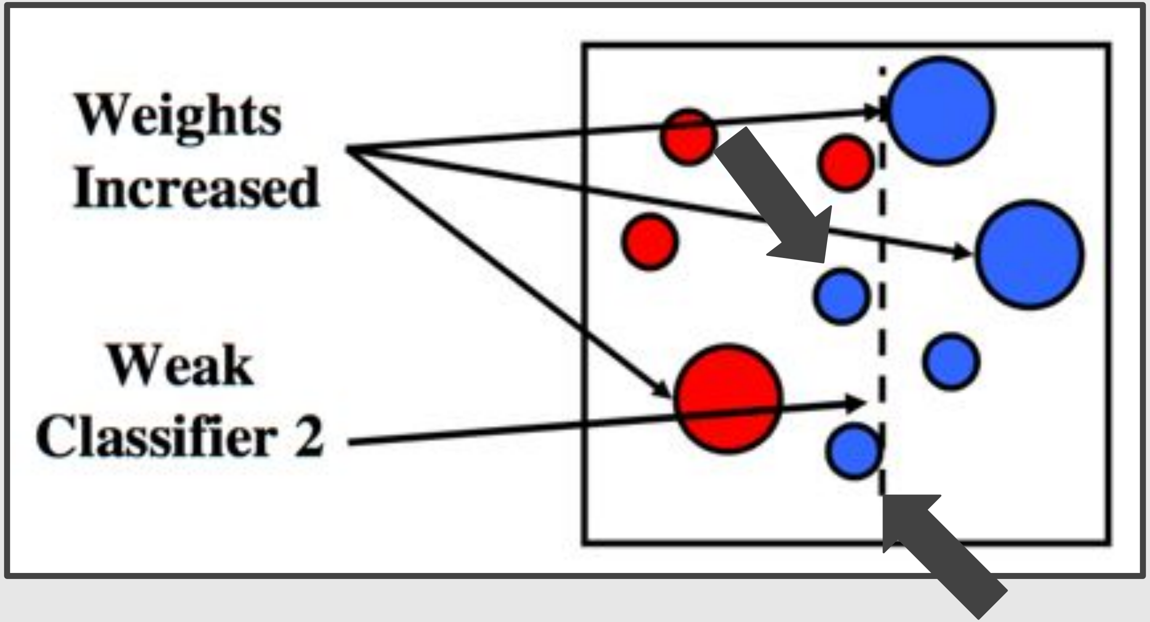
AdaBoost



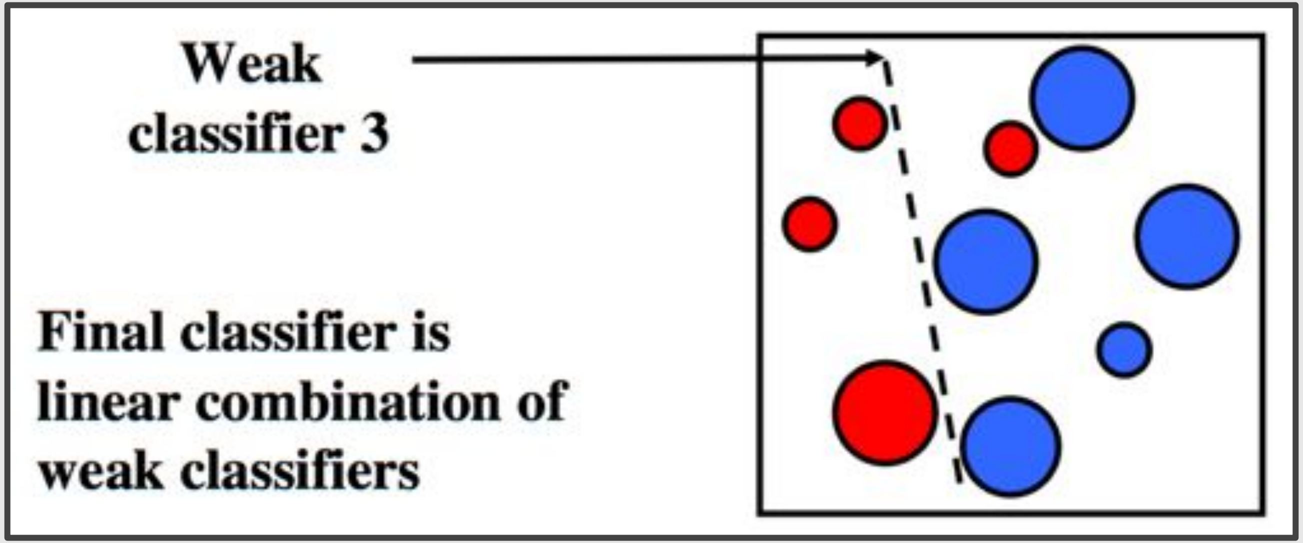
AdaBoost



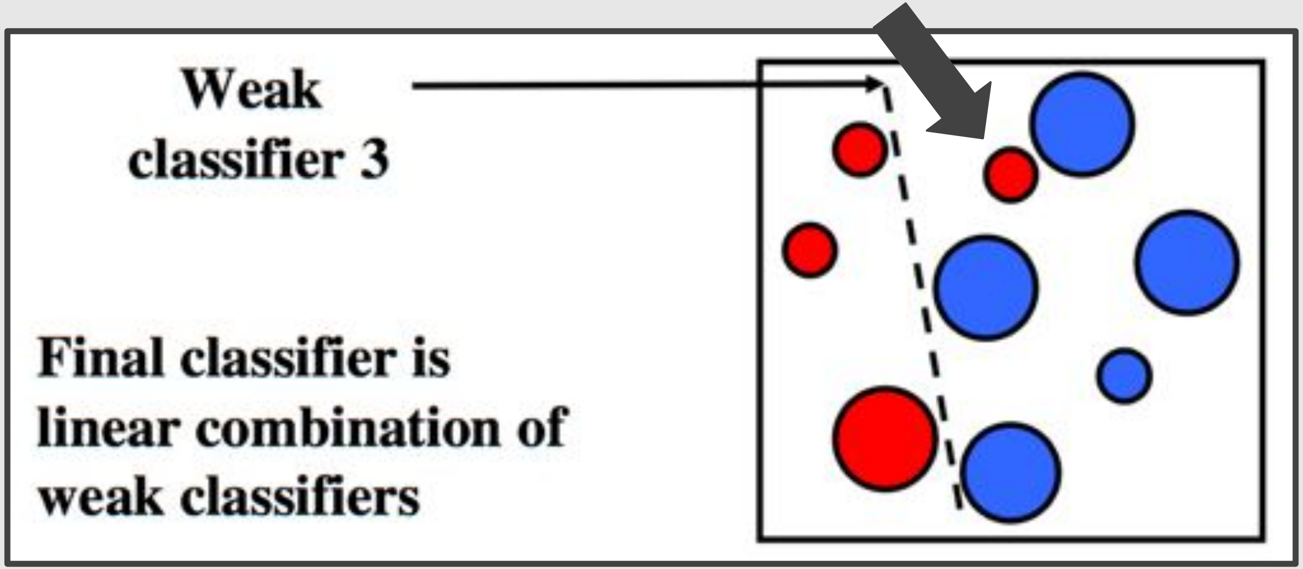
AdaBoost

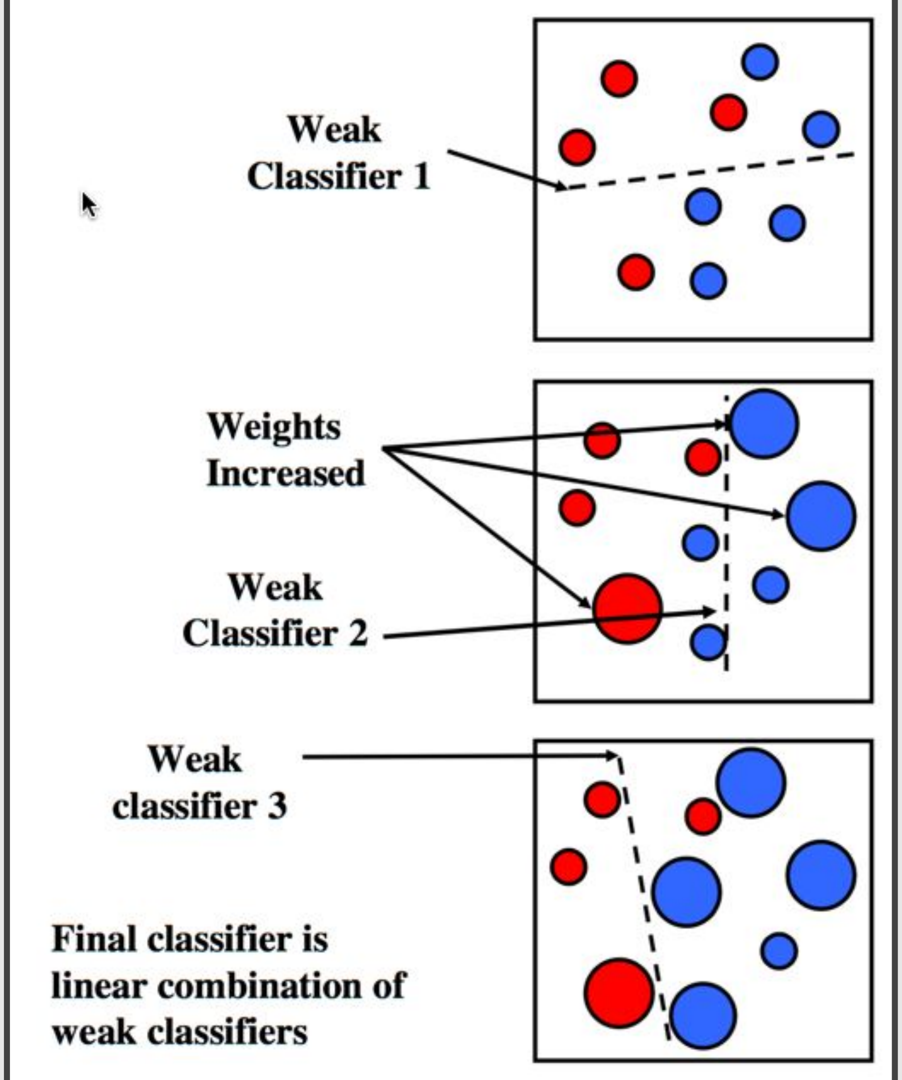


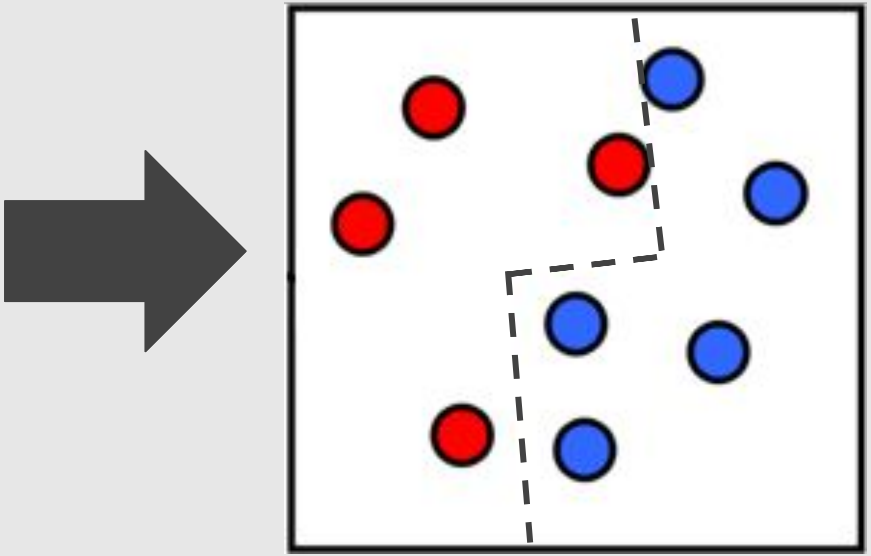
AdaBoost


Weak
classifier 3
Final classifier is
linear combination of
weak classifiers

AdaBoost


Weak
classifier 3
Final classifier is
linear combination of
weak classifiers


Weak
Classifier 1
Weights
Increased
Weak
Classifier 2
Weak
classifier 3
Final classifier is
linear combination of
weak classifiers



AdaBoost

1. Assign every observation, x_i, an initial weight value, $w_i = \frac{1}{n}$, where n is the total number of observations.
2. Train a "weak" model. (most often a decision tree)
3. For each observation:
    3.1. If predicted incorrectly, w_i is increased
    3.2. If predicted correctly, w_i is decreased
4. Train a new weak model where observations with greater weights are given more priority.
5. Repeat steps 3 and 4 until observations perfectly predicted or a preset number of trees are trained.

ChrisAlbon

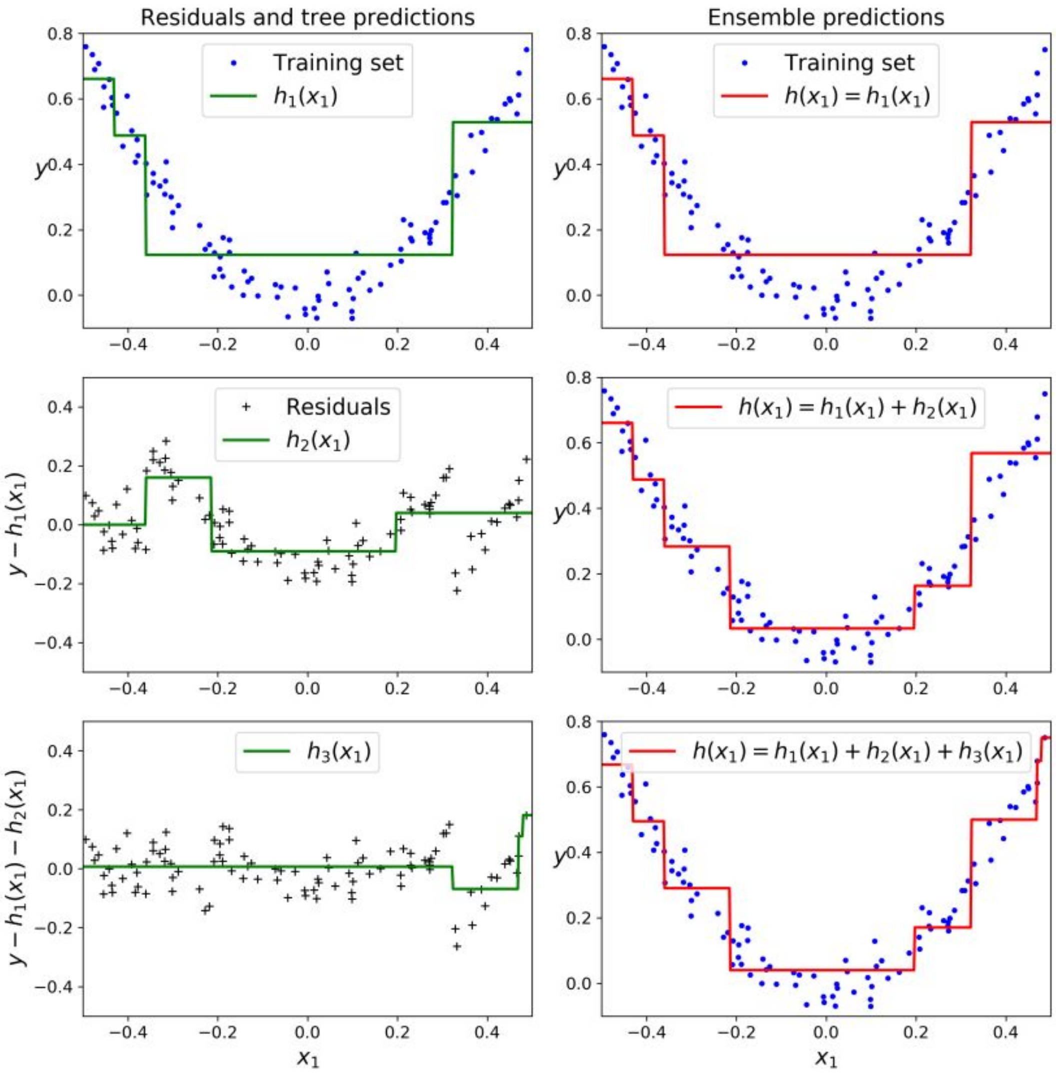
Gradient Boosting [Breiman, 1997]

- Instead of tweaking the instance weights at every iteration like AdaBoost does, this method fit the new predictor to the **residual errors** made by the previous predictor.

Gradient Boosting [Breiman, 1997]

- Instead of tweaking the instance weights at every iteration like AdaBoost does, this method fit the new predictor to the **residual errors** made by the previous predictor.

- Instead of training on a newly sample distribution, the weak learner **trains on the remaining errors**.


Residuals and tree predictions
Ensemble predictions
Training set
$h_1(x_1)$
$h(x_1) = h_1(x_1)$
Residuals
$h_2(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1)$
$h_3(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$
y
$y - h_1(x_1)$
$y - h_1(x_1) - h_2(x_1)$
x_1

Gradient Boosting [Breiman, 1997]

1. Fit a simple linear regressor or decision tree on data
[call x as input and y as output]

2. Calculate error residuals. Actual target value, minus predicted target value
[e1 = y - y_predicted1]

3. Fit a new model on error residuals as target variable with same input variables
[call it e1_predicted]

4. Add the predicted residuals to the previous predictions
[y_predicted2 = y_predicted1 + e1_predicted]

5. Fit another model on residuals that is still left, i.e. **[e2 = y - y_predicted2]** and repeat steps 2 to 5 until it starts overfitting or the sum of residuals become constant.

Gradient Boosting [Breiman, 1997]

- XGboost [Chen and Guestrin, 2016]:

  Extreme Gradient Boosting

  https://github.com/tqchen/xgboost

  It aims at being extremely fast, scalable and portable.

Today's Agenda

- Ensemble Methods
  - Bagging (and Pasting)
  - Boosting
  - **Stacking**

Stacking

Stacking [Wolpert, 1992]

- Stacking (short for Stacked Generalization)

“Stacked Generalization”, D. Wolpert (1992): http://goo.gl/9I2NBw

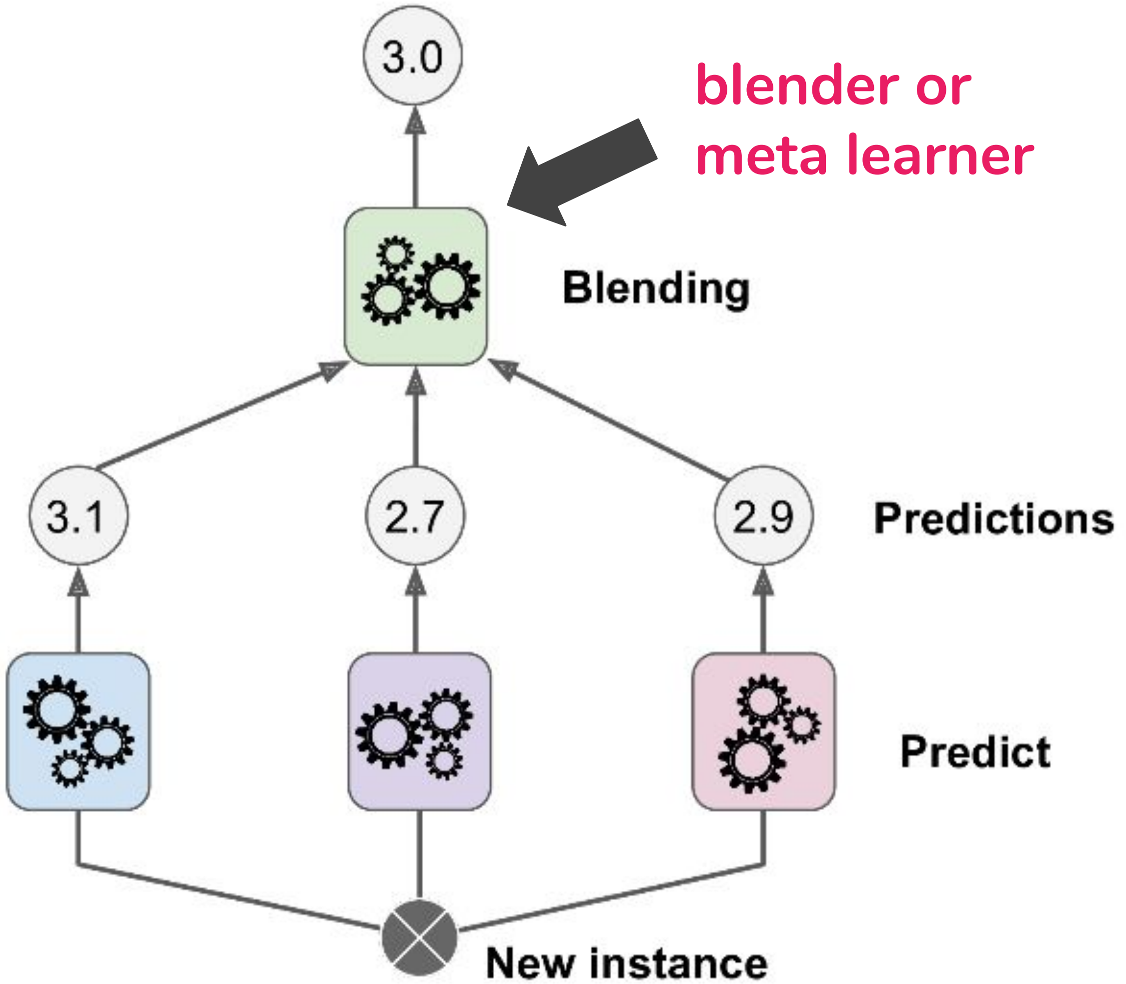
Stacking [Wolpert, 1992]

- Stacking (short for Stacked Generalization)

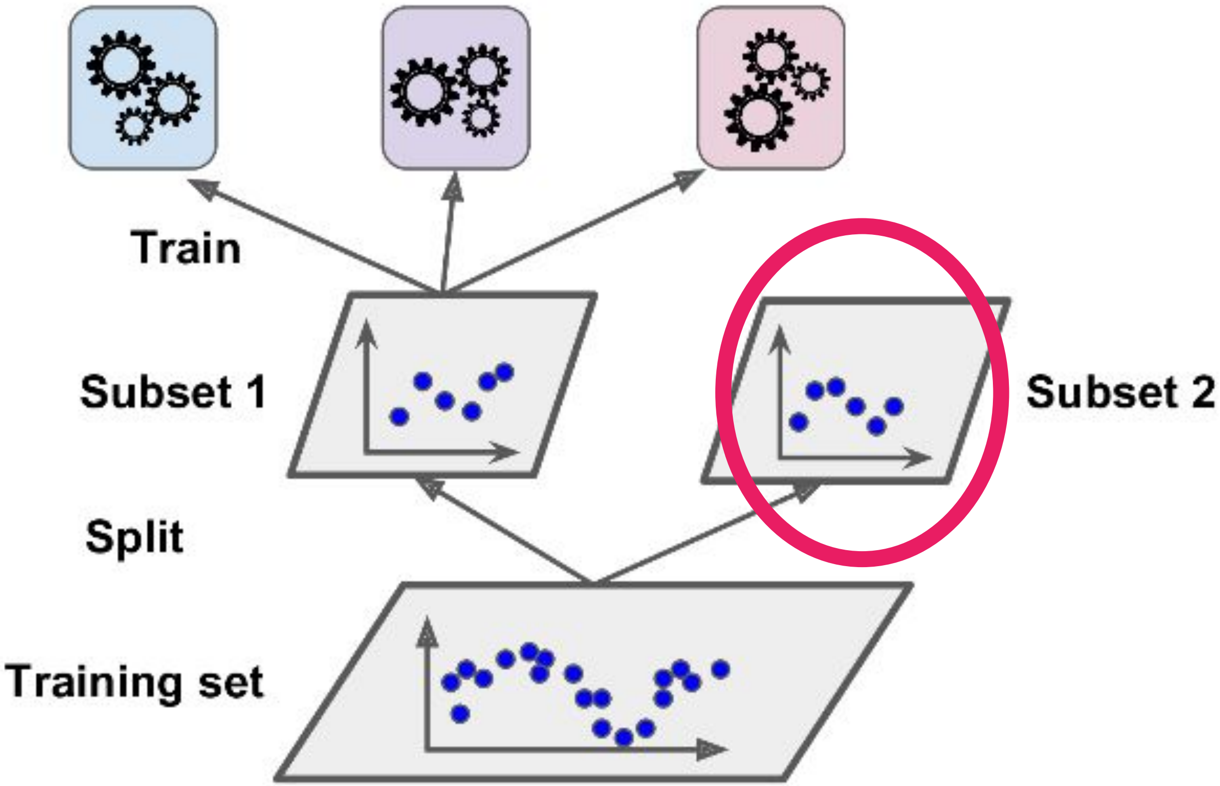
- Instead of using trivial functions (such as hard voting) to aggregate the predictions of all predictors in an ensemble, we **train a model to perform this aggregation**.

"Stacked Generalization", D. Wolpert (1992): http://goo.gl/9I2NBw

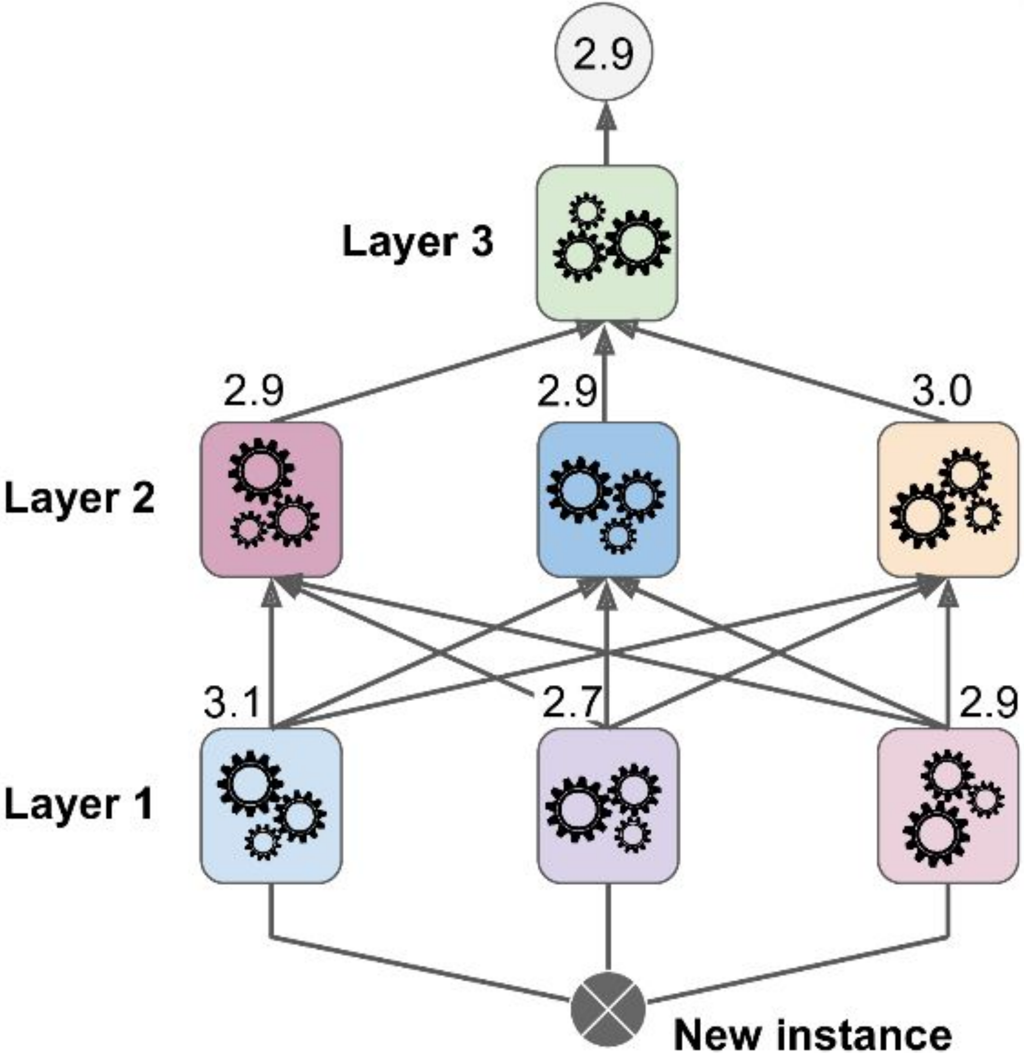
Stacking


3.0
blender or
meta learner
Blending
3.1
2.7
2.9
Predictions
Predict
New instance

To train the blender, a common approach is to use a **hold-out set**.



Multi-layer Stacking Ensemble



Stacking [Wolpert, 1992]

- Scikit-Learn does not support stacking directly. =(

“Stacked Generalization”, D. Wolpert (1992): http://goo.gl/9I2NBw

Deep-Learning Ensembles for Skin-Lesion Segmentation, Analysis, Classification: RECOD Titans at ISIC Challenge 2018

Alceu Bissoto*, Fábio Perez*, Vinícius Ribeiro*, Michel Fornaciali, Sandra Avila, Eduardo Valle†

I. History

Our team has worked on skin lesion analysis since early 2014 [1], and has employed deep learning with transfer learning for that task since 2015 [2]. From 2016 onwards, the community moved from traditional techniques towards deep learning, following the general trend of computer vision [3]. Deep learning poses a challenge for medical applications, as it needs very large training sets. Thus, transfer learning becomes crucial for success in those applications, motivating our paper for ISBI 2017 [4]. Until 2017, the contribution of each factor of a deep learning solution (e.g., model choice, dataset size, data augmentation, image normalization, etc.) to the performance of a skin lesion classifier was not evident. We cleared such question by extensively analyzing several combinations of architectures, dataset sizes, and other eight relevant aspects [5].

We participated in the ISIC Challenge 2017, being ranked in 1st place for melanoma classification and 5th place for skin lesion segmentation [6]. In 2018, for the first time, we participated in all three tasks.Although our team has a long experience with skin-lesion classification (Task 3) and moderate experience with lesion segmentation (Task 1), this Challenge was the very first time we worked on attribute detection (Task 2).

II. Generalities

A. Strategy

We aimed, from the start, at deep learning solutions for all tasks. We know from experience that the success factors

complexity has to bring proportional improvements over the metrics, or we will prefer the simpler model.

Each task allowed up to 3 distinct submissions. We used them to contrast models trained with extra data with models trained with challenge-data only, or to compare different ways to ensemble the final solutions.

B. Data

In previous work, we showed that the training set size responds by almost 50% of the variation on the prediction power of the classifier [5]. The freedom to use external sources enabled us to gather more data to boost our models. First, we restricted ourselves to publicly available (for free, or for a fee) sources with high-quality images:

ISIC 2018 Challenge [7, 8] the official challenge dataset, with 10,015 dermoscopic images.
ISIC Archive[1] with over 13,000 dermoscopic images.
Interactive Atlas of Dermoscopy [9] with 1,000+ clinical cases, each with dermoscopic, and close-up clinical images.
Dermofit Image Library [10] with 1,300 images.
PH2 Dataset [11] with 200 dermoscopic images.

However, due to the extreme imbalance of the dataset, we decided to gather extra images for the severely underrepresented classes (namely Actinic keratosis, Basal cell carcinoma, Dermatofibroma, and Vascular lesion). We found images browsing sources on the web, and asking for contributions from partner researchers in Medical Science (acknowledged in the final section). The web sources were **Der-**

https://arxiv.org/pdf/1808.08480.pdf

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 6 & 7
- Pattern Recognition and Machine Learning, Chap. 14
- Pattern Classification, Chap 8 & 9 (Sec. 9.5)