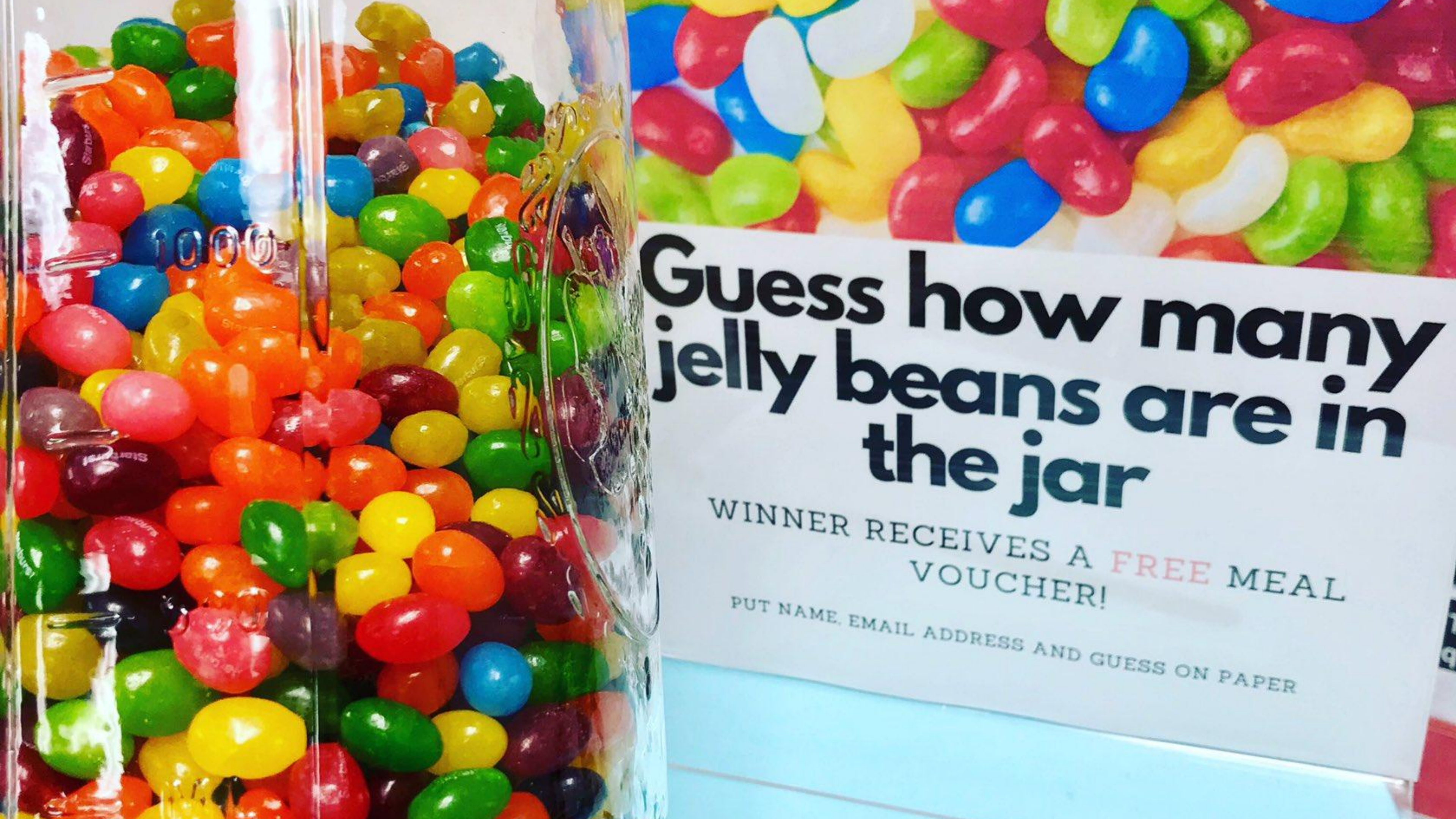
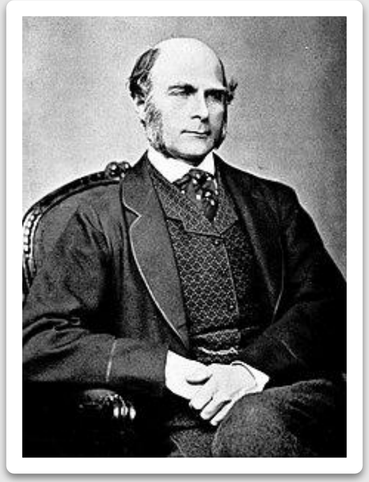
Ensemble Learning

Machine Learning

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

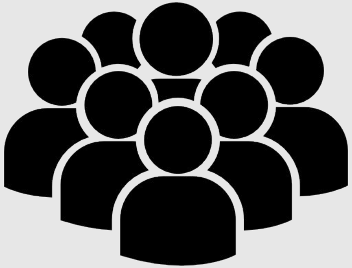
MC886/MO444, November 22, 2022


1000
Guess how many jelly beans are in the jar
WINNER RECEIVES A FREE MEAL VOUCHER!
PUT NAME, EMAIL ADDRESS AND GUESS ON PAPER



Francis Galton
(1822-1909)

Animal's weight?



~800 people
542 kg

543 kg

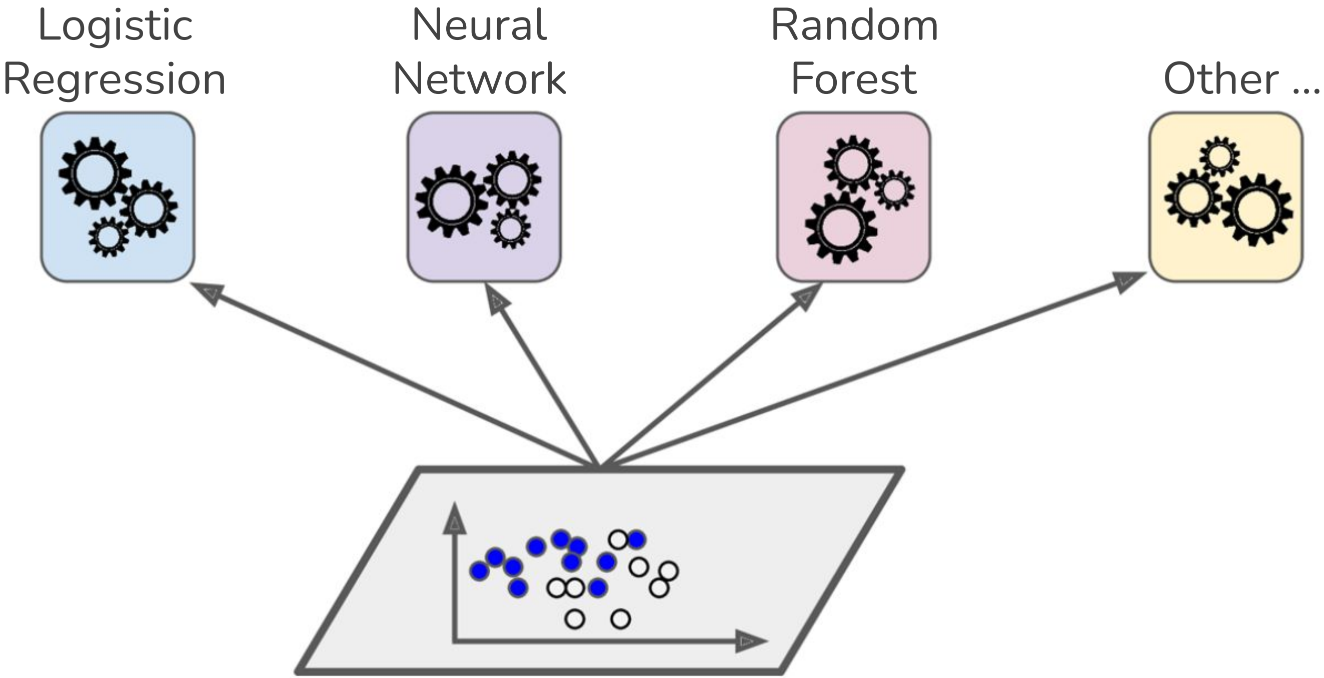
Wisdom of the Crowd

Ensemble Learning

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

Voting Classifiers

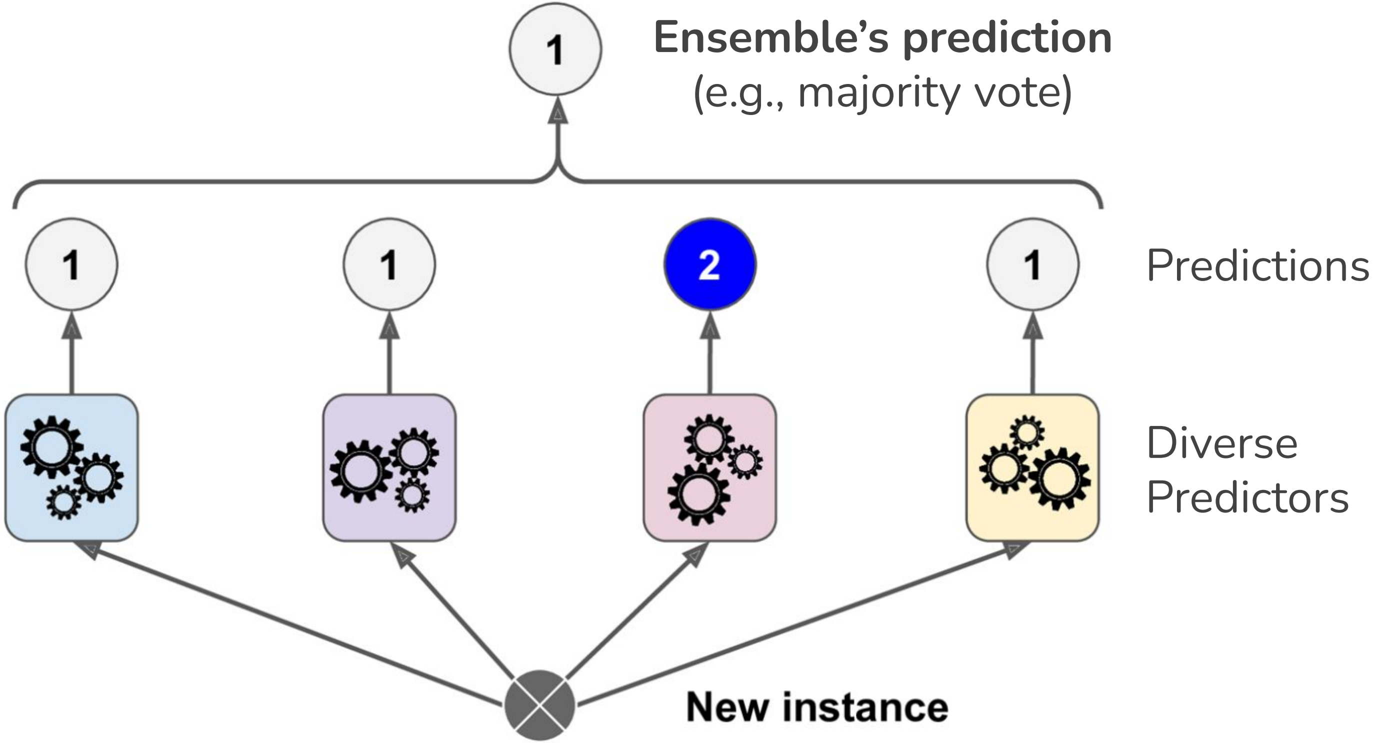
~80%



Diverse Predictors

Voting Classifiers

Hard/Soft voting classifier



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.
- 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 RandomForestClassif
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
```

Today's Agenda

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

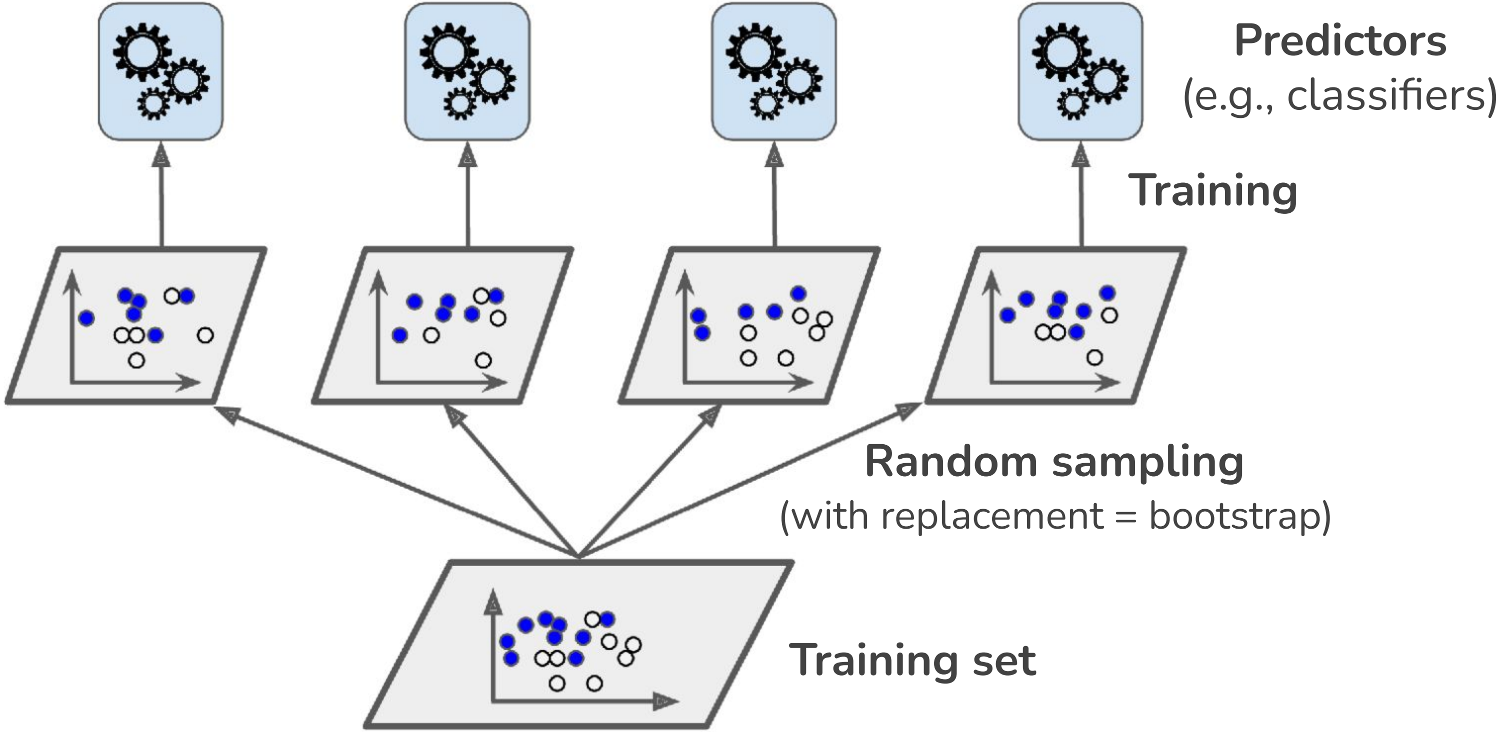
Bagging & Pasting

Bagging and Pasting

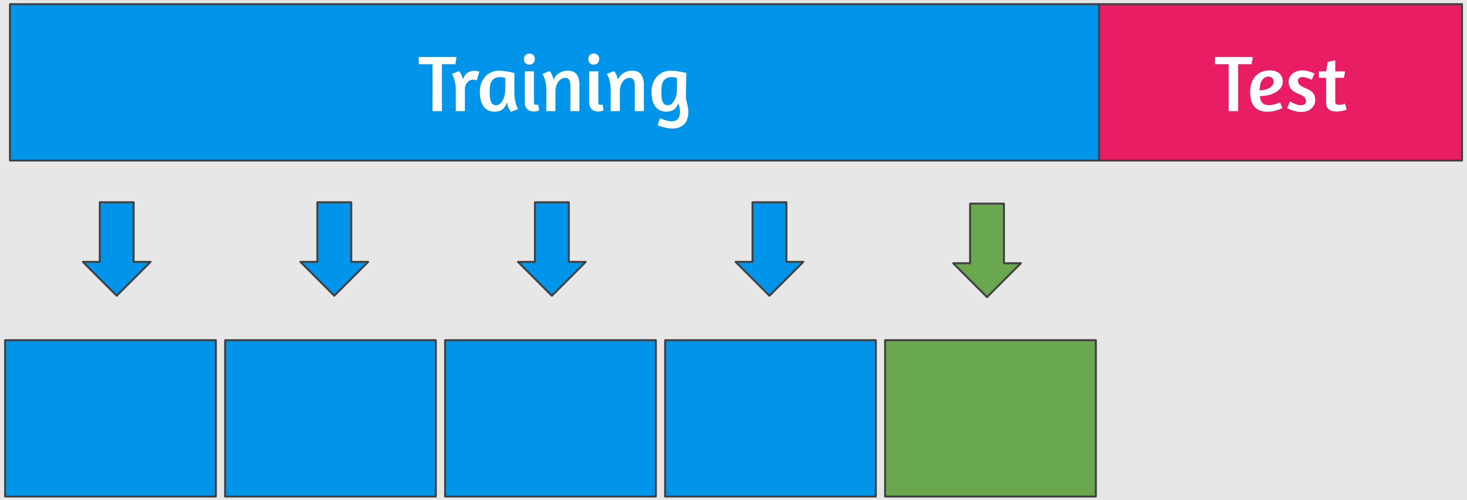
- Use **the same training algorithm** for every predictor, but to train them on **different random subsets** of the training set.
- **Bagging** (short for Bootstrap Aggregating): sampling is performed **with** replacement.
- **Pasting**: sampling is performed **without** replacement.

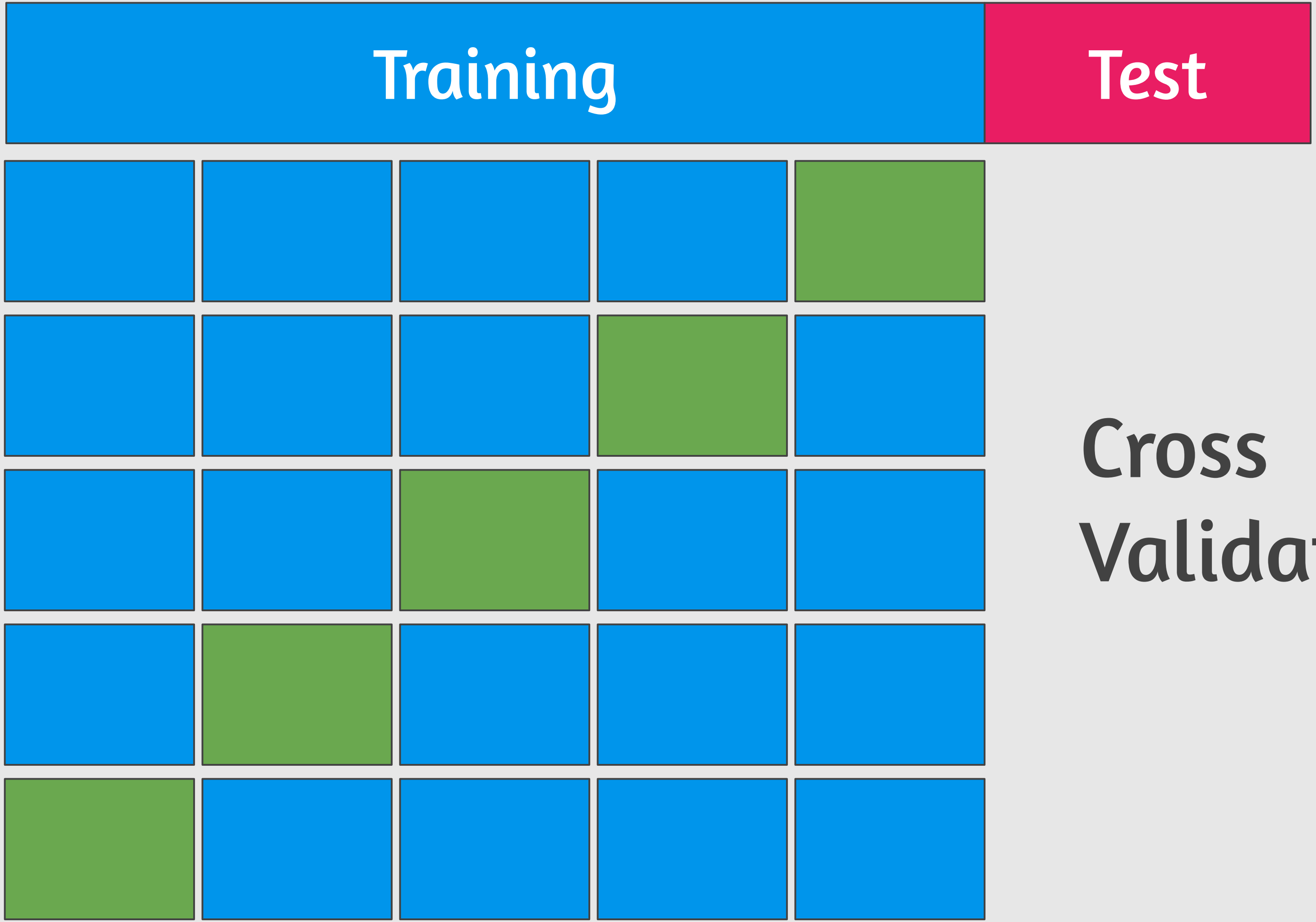
"Bagging Predictors", L. Breiman (1996): http://goo.gl/o42tml

Bagging and Pasting

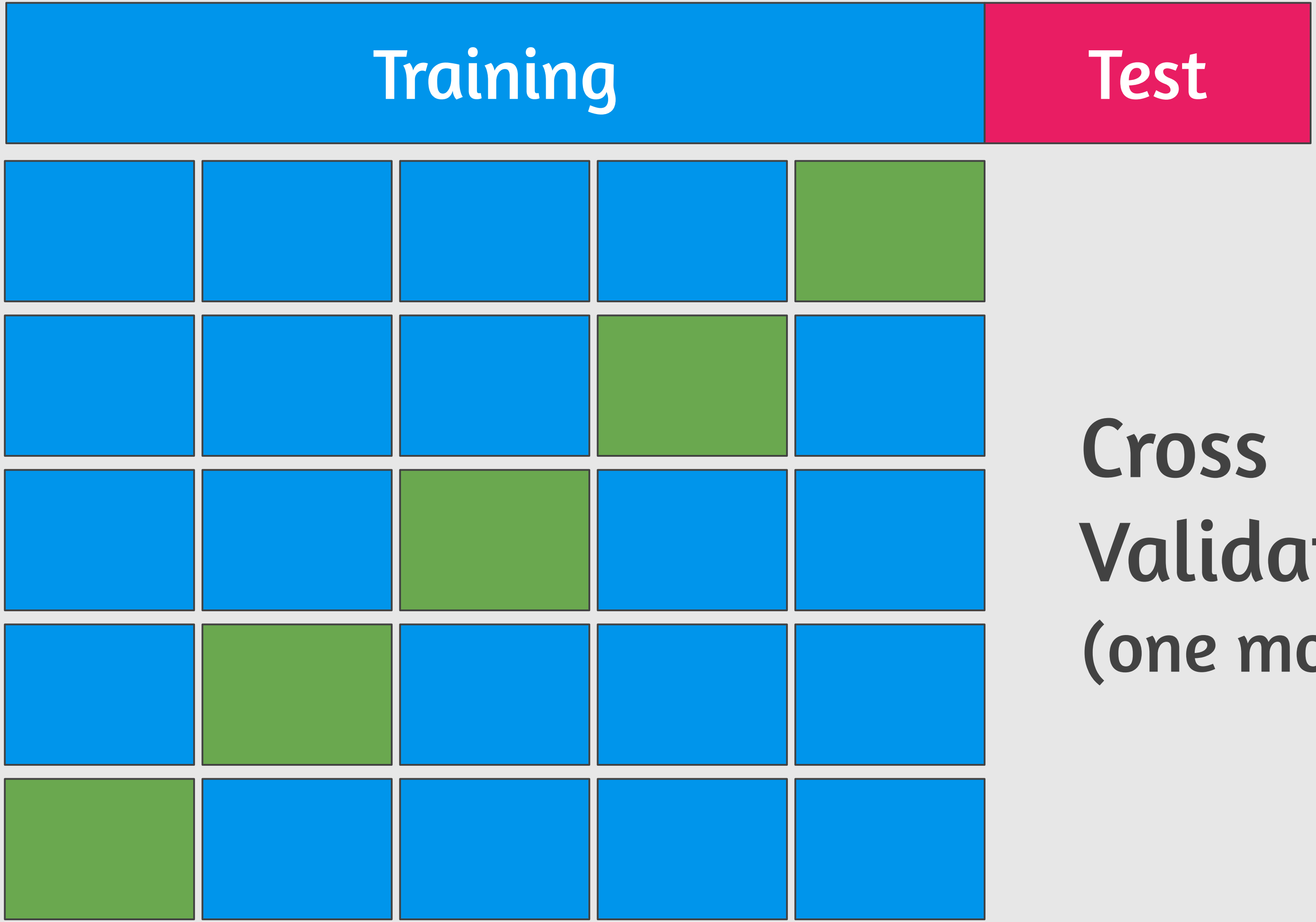


Bagging vs. Cross Validation

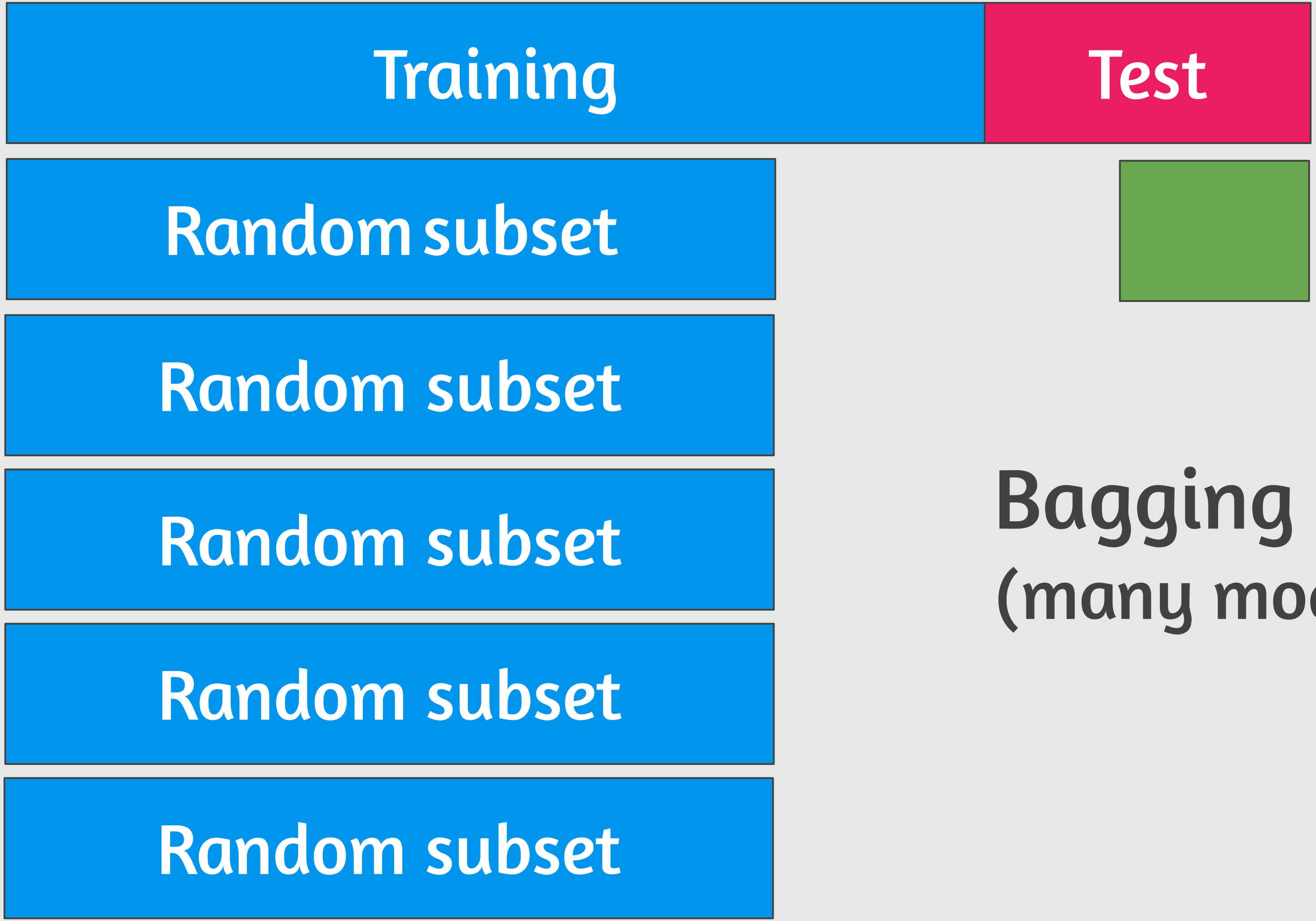




Cross Validation

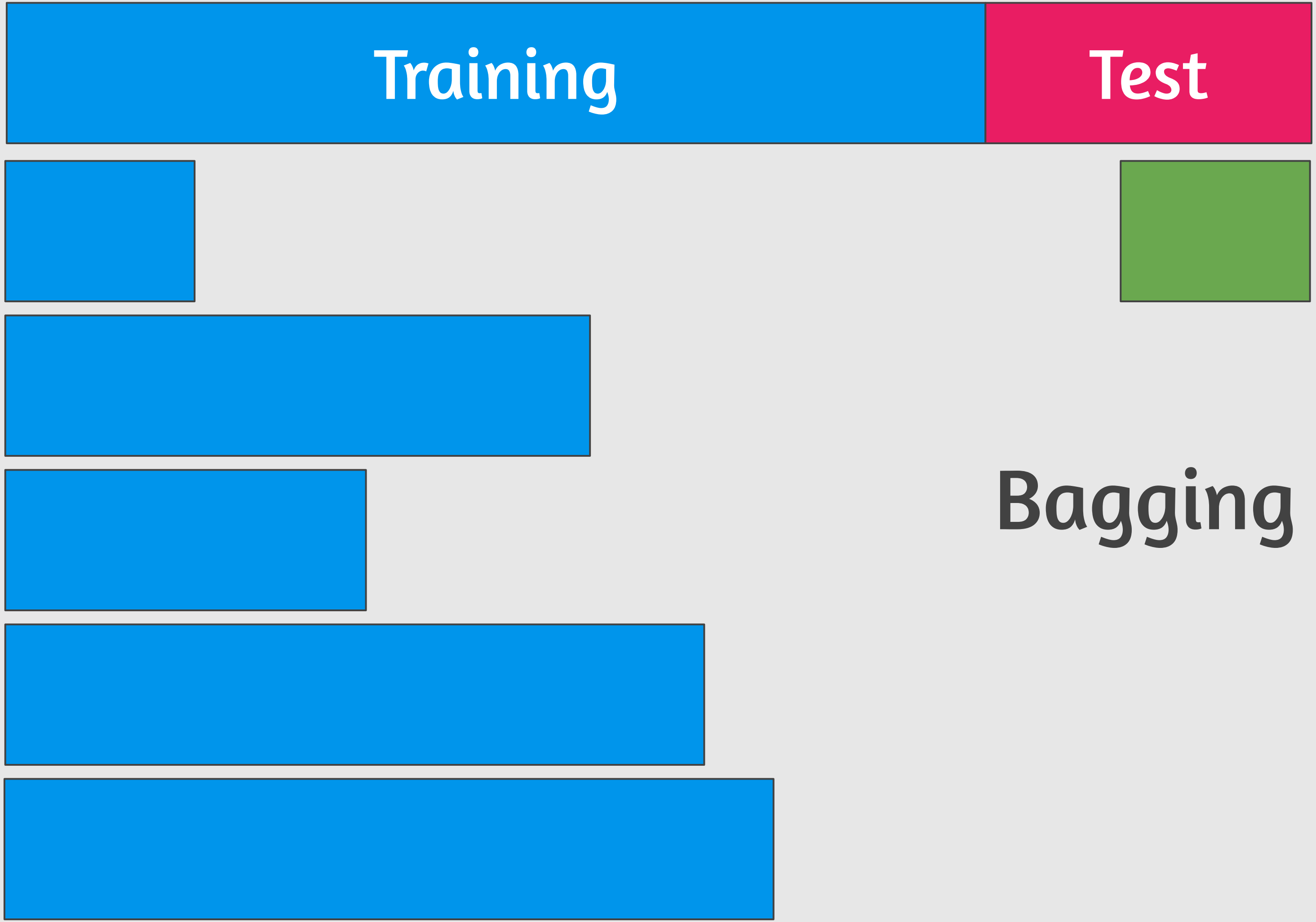


Cross Validation (one model)



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)
```

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.
- Most popular: AdaBoost and Gradient Boost.

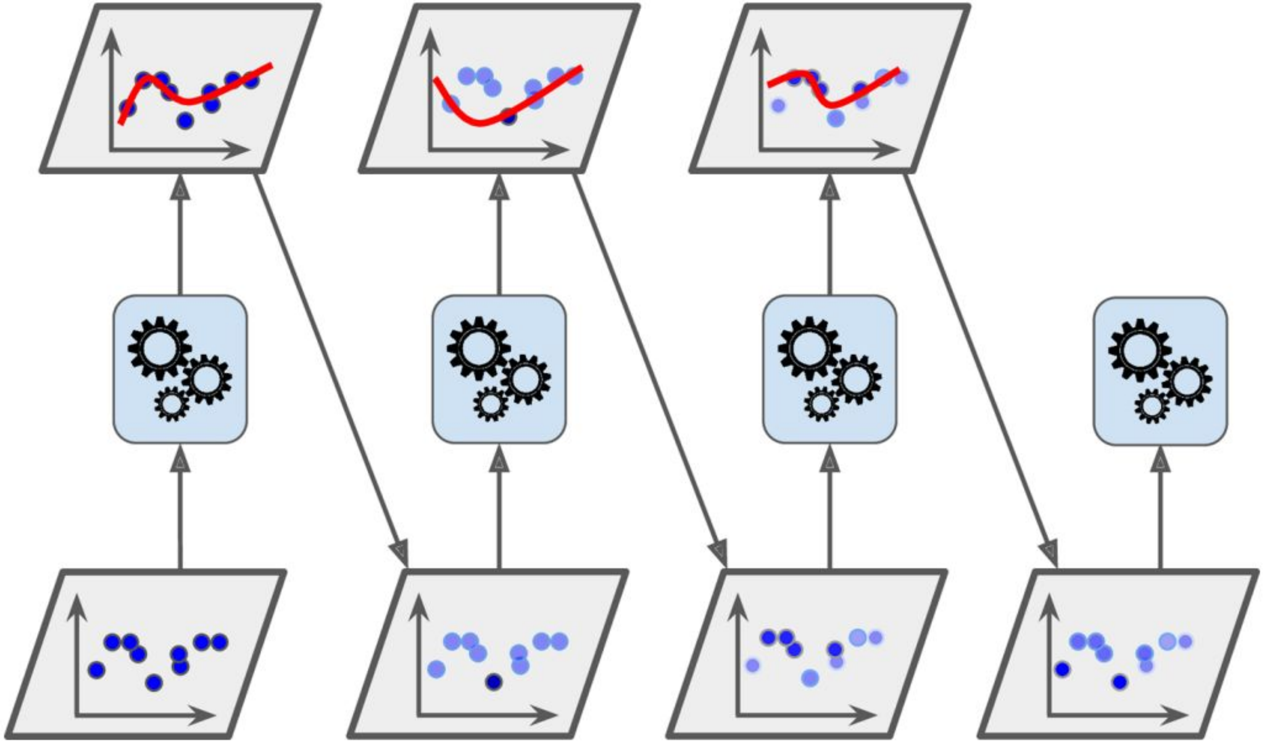
AdaBoost [Freund and Schapire, 1997]

- 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**.

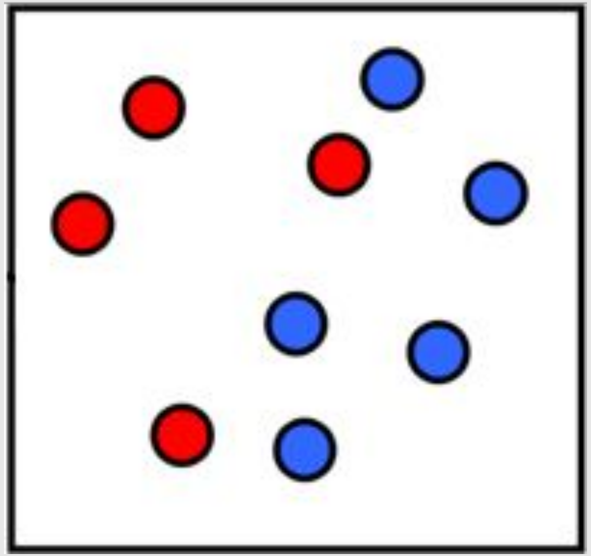
- This results in new predictors focusing more and more on **the hard cases**.

“A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”, http://goo.gl/OIduRW

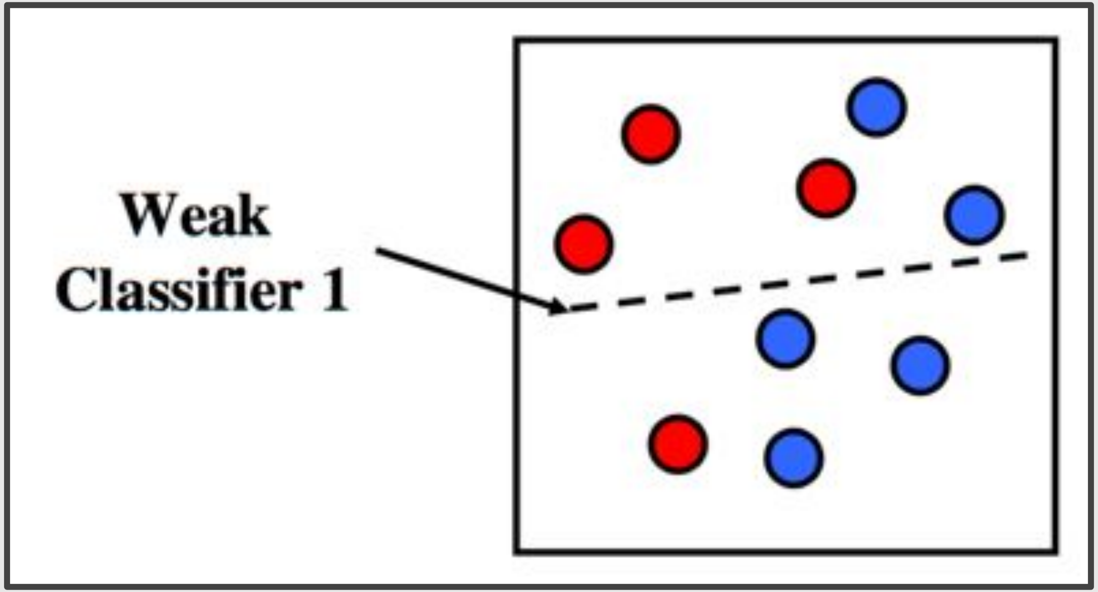
AdaBoost



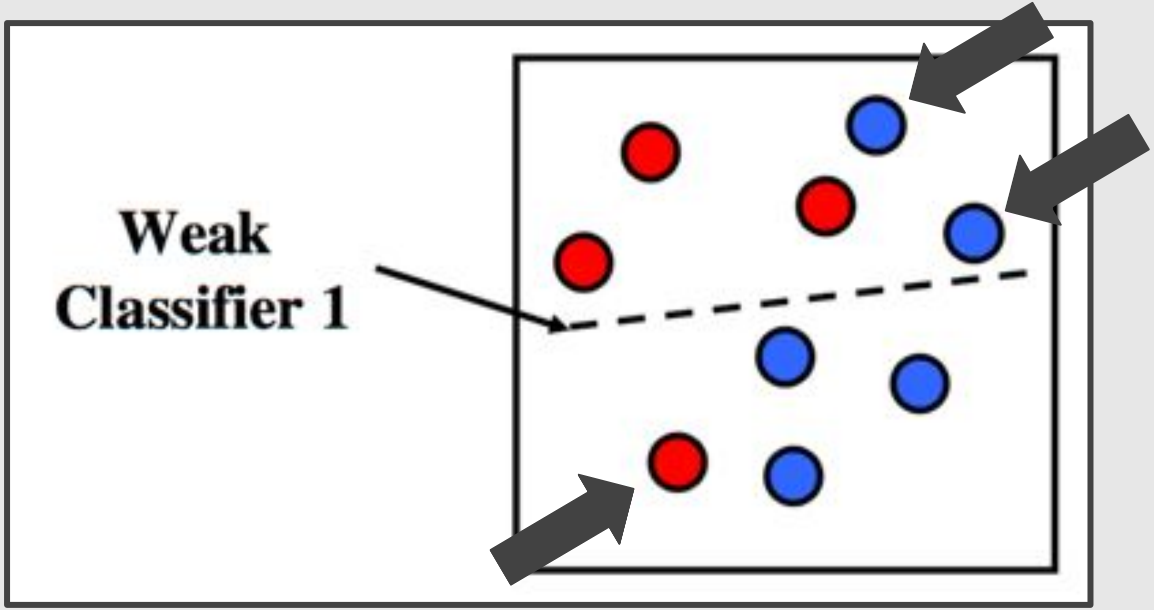
AdaBoost



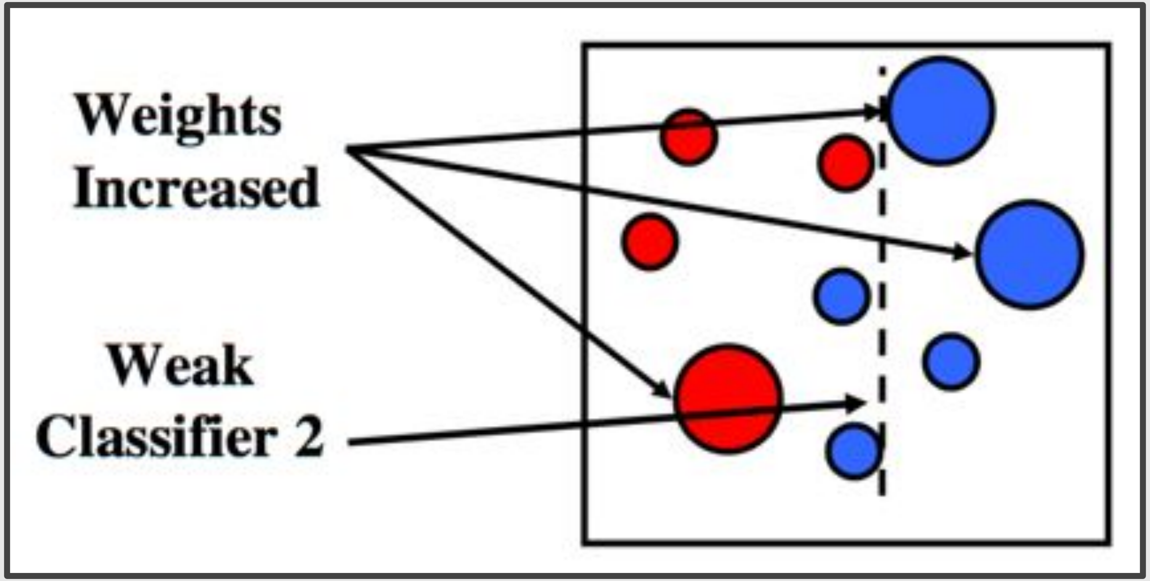
AdaBoost



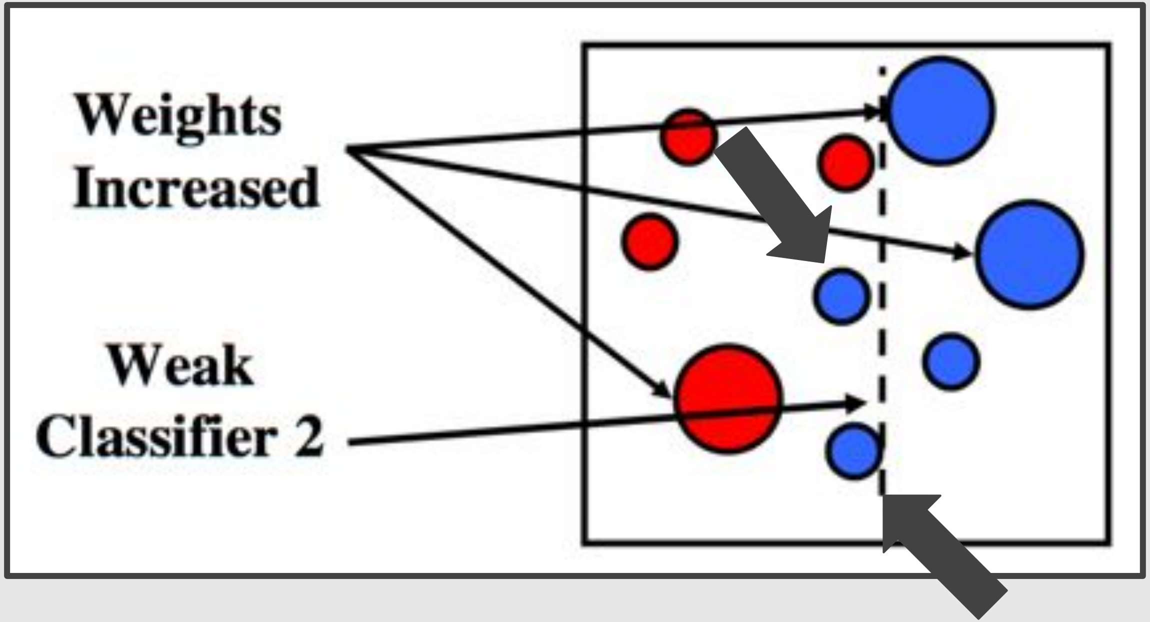
AdaBoost



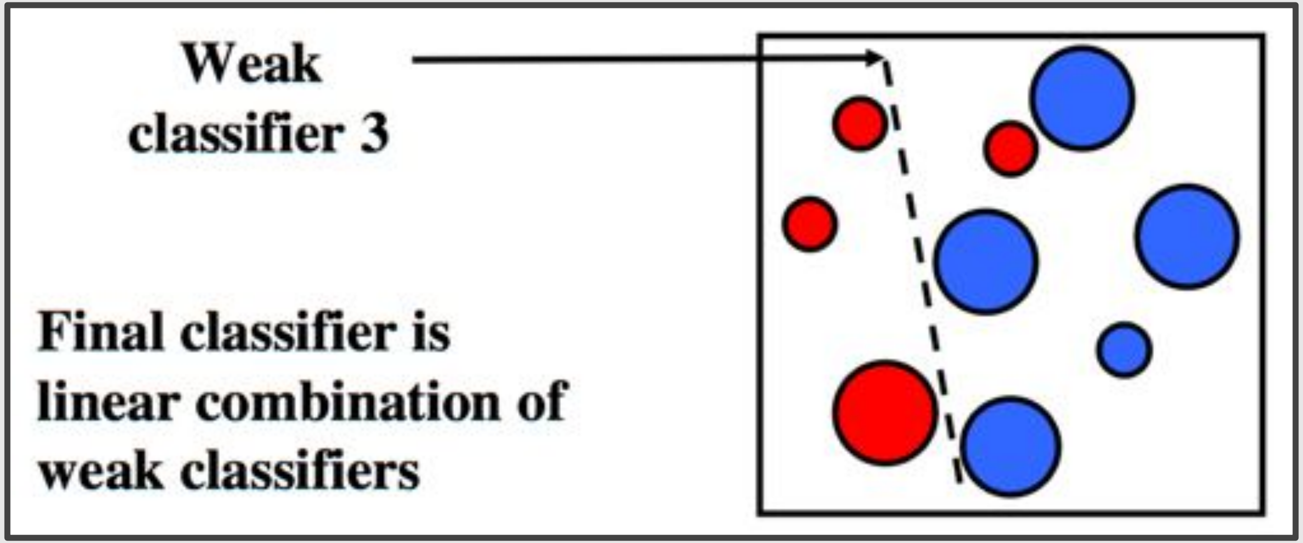
AdaBoost



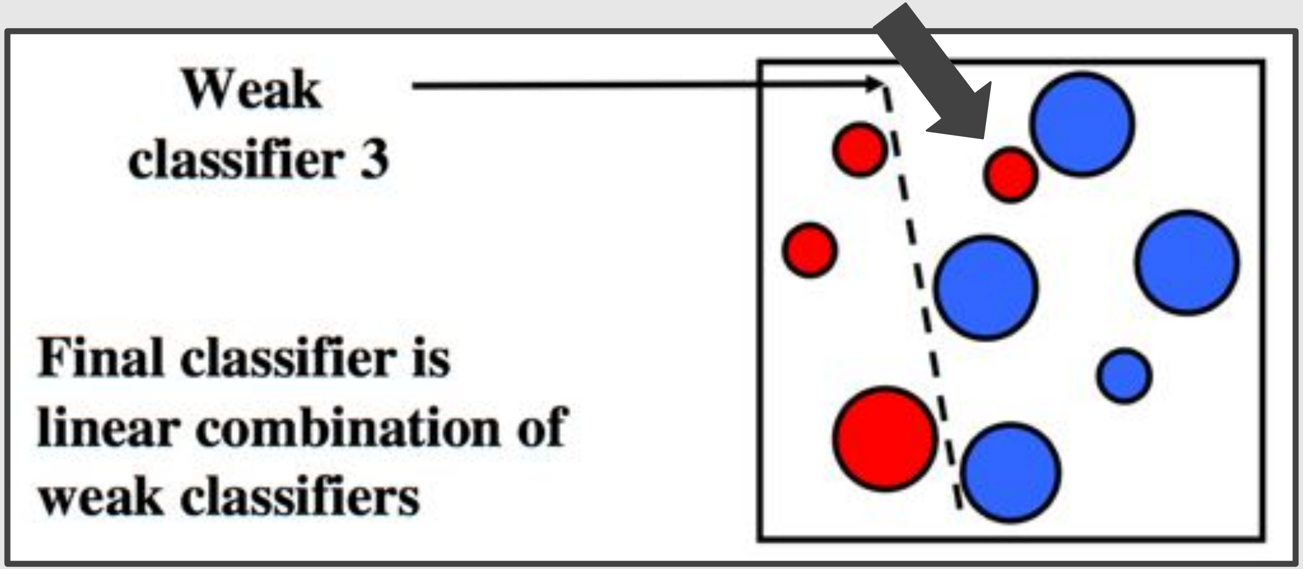
AdaBoost



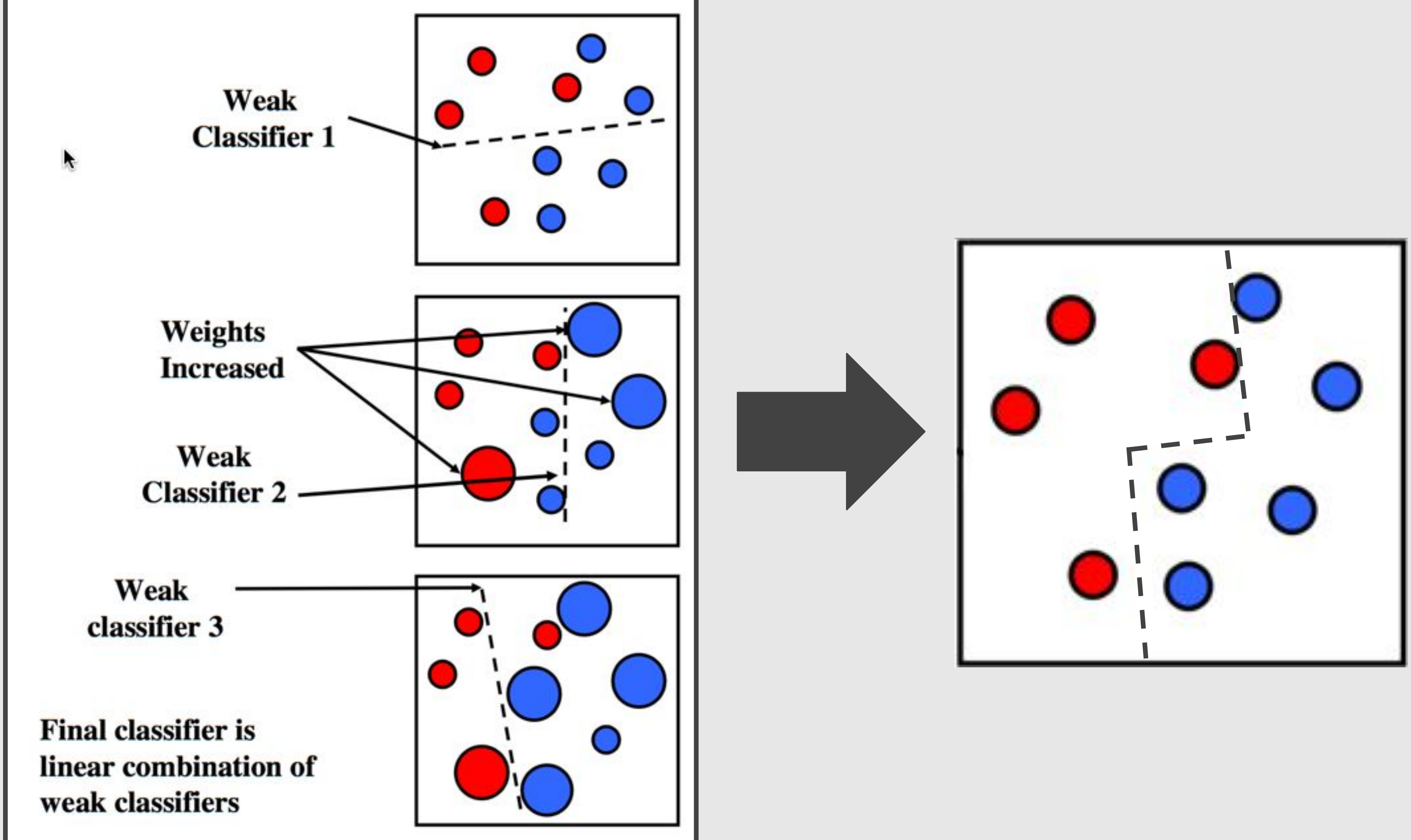
AdaBoost


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

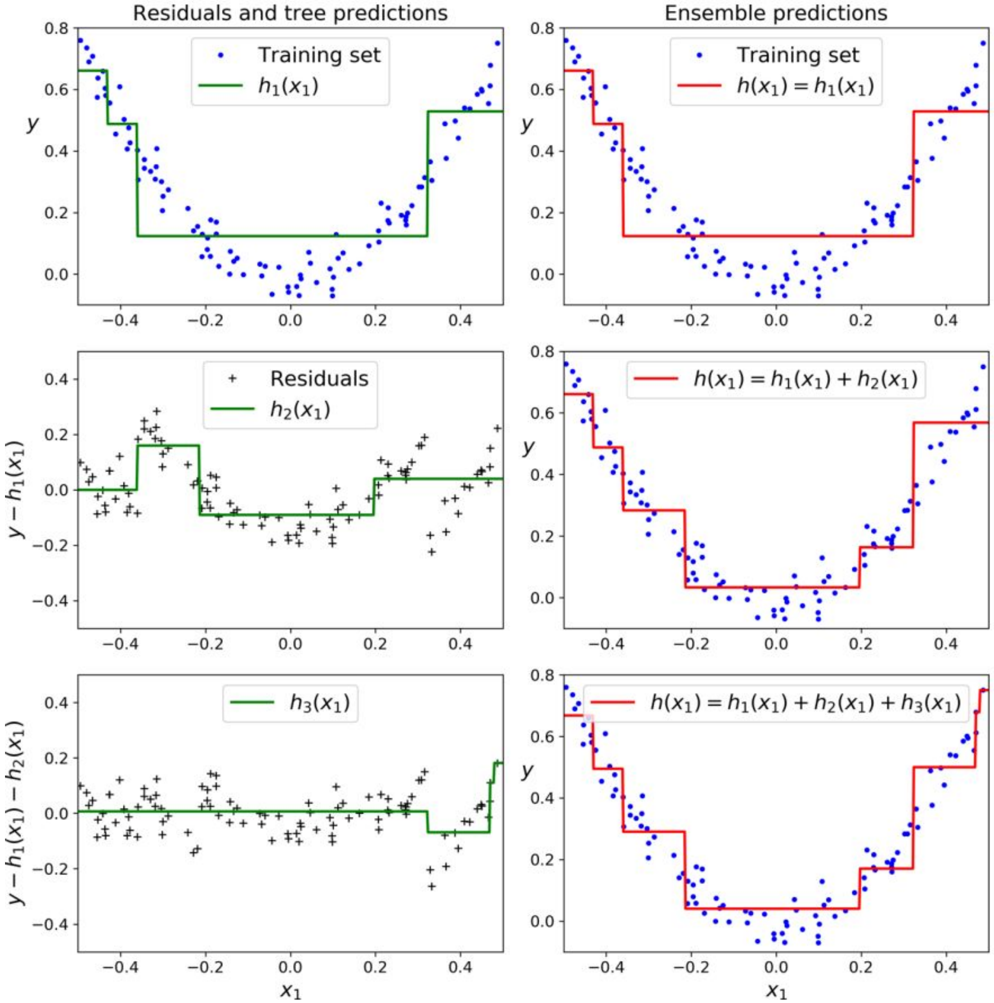
AdaBoost

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.

Chris Albon

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

```python
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)

y_pred = sum(tree.predict(X_new) for tree in (tree_reg1, tree_reg2, tree_reg3))
```

```python
from sklearn.ensemble import GradientBoostingRegressor
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3, learning_rate=1.0)
gbrt.fit(X, y)
```

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**

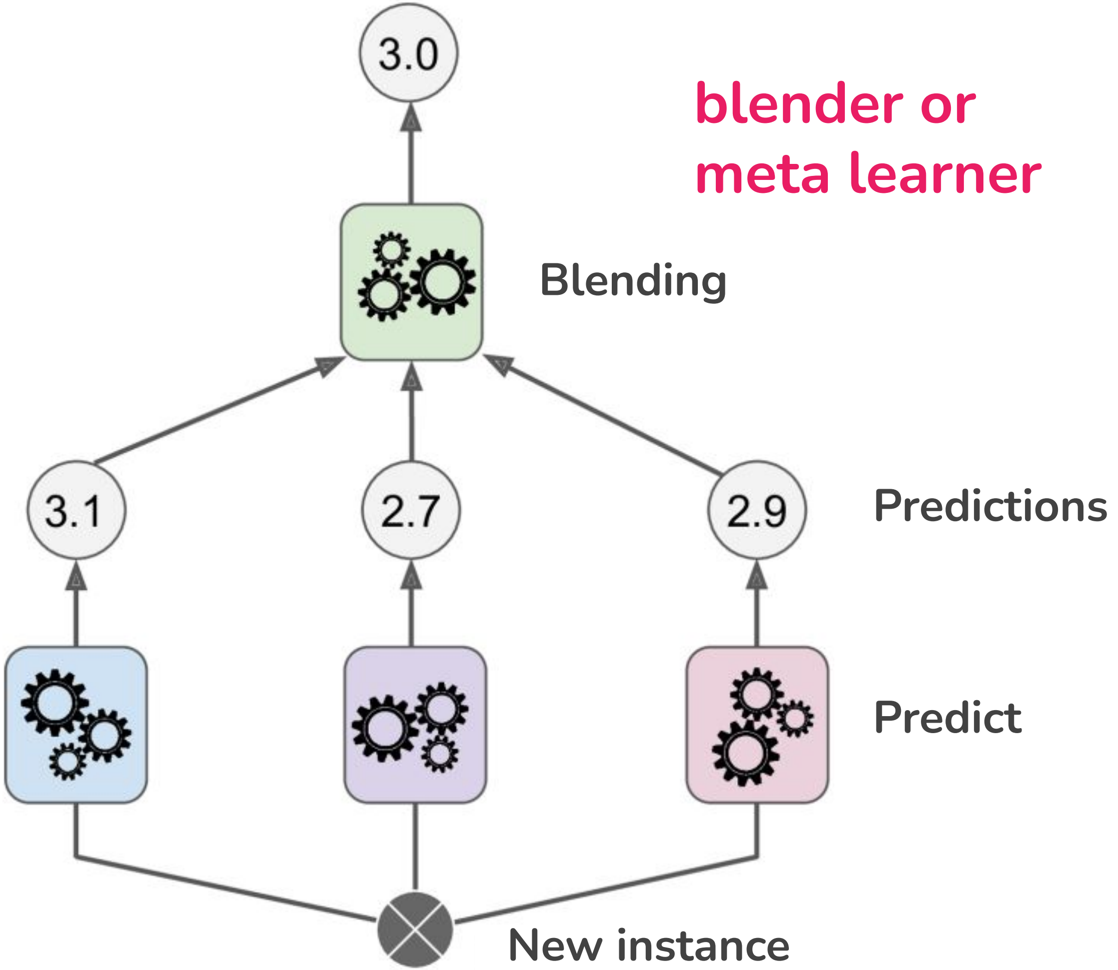
Stacking

Stacking [Wolpert, 1992]

- Stacking (short for Stacked Generalization)
- 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**.

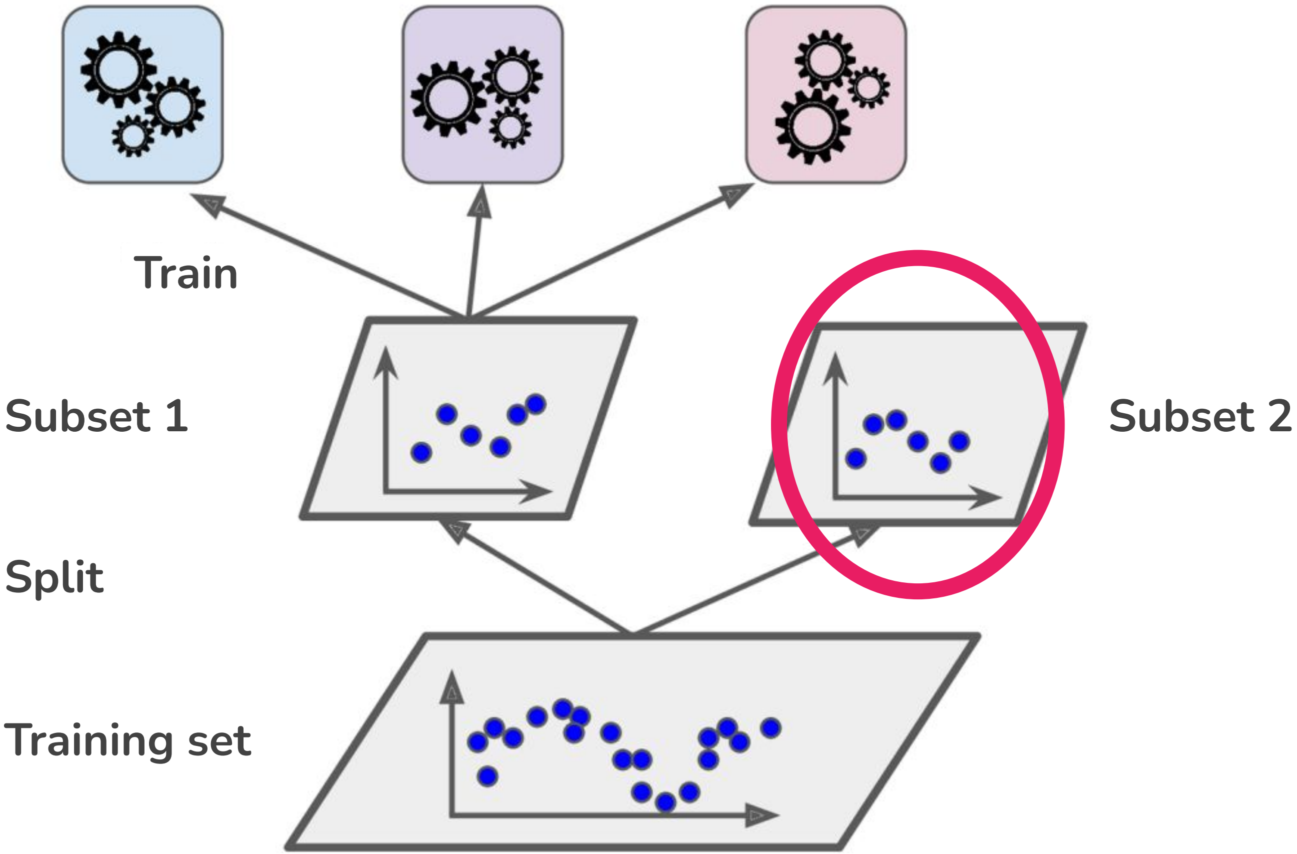
"Stacked Generalization", http://goo.gl/9I2NBw

Stacking

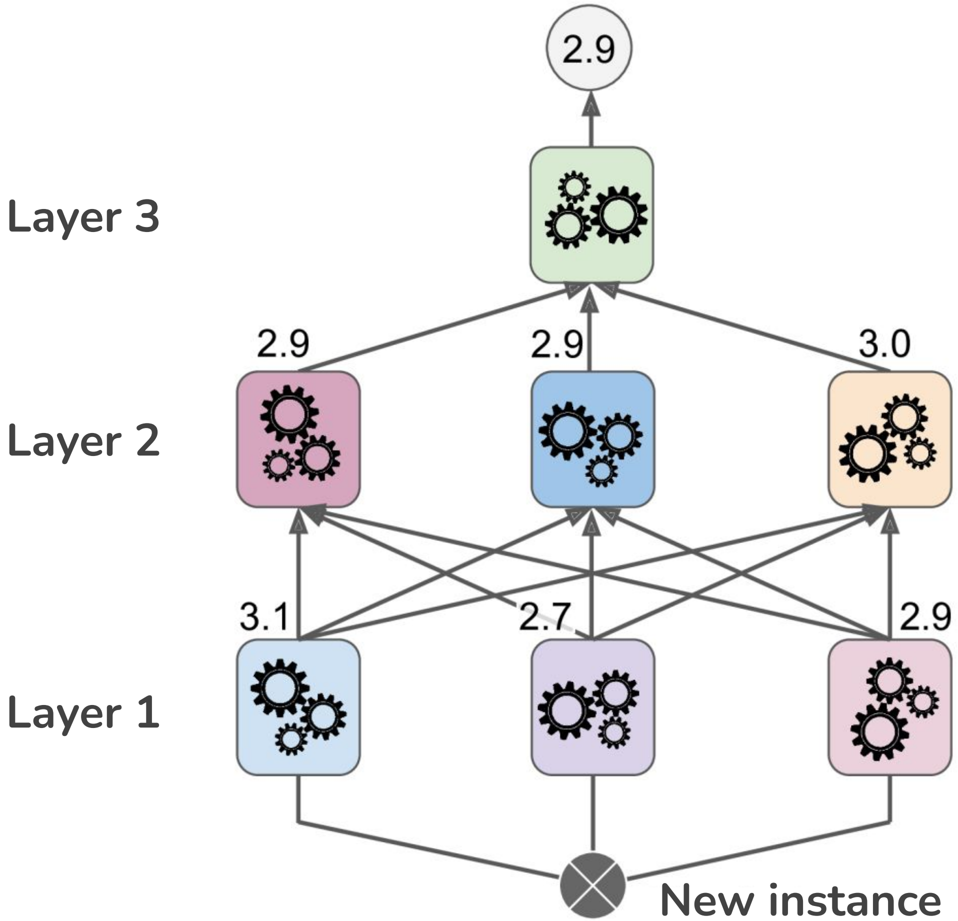


3.0
blender or
meta learner
Blending
3.1
2.7
2.9
Predictions
Predict
New instance

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. =(~~
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

"Stacked Generalization", http://goo.gl/9I2NBw

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)
- "Scikit Learn Ensemble Learning, Bootstrap Aggregating (Bagging) and Boosting" https://youtu.be/X3Wbfb4M33w