DataSci 420

lesson 6: ensemble models

Seth Mottaghinejad

today's agenda

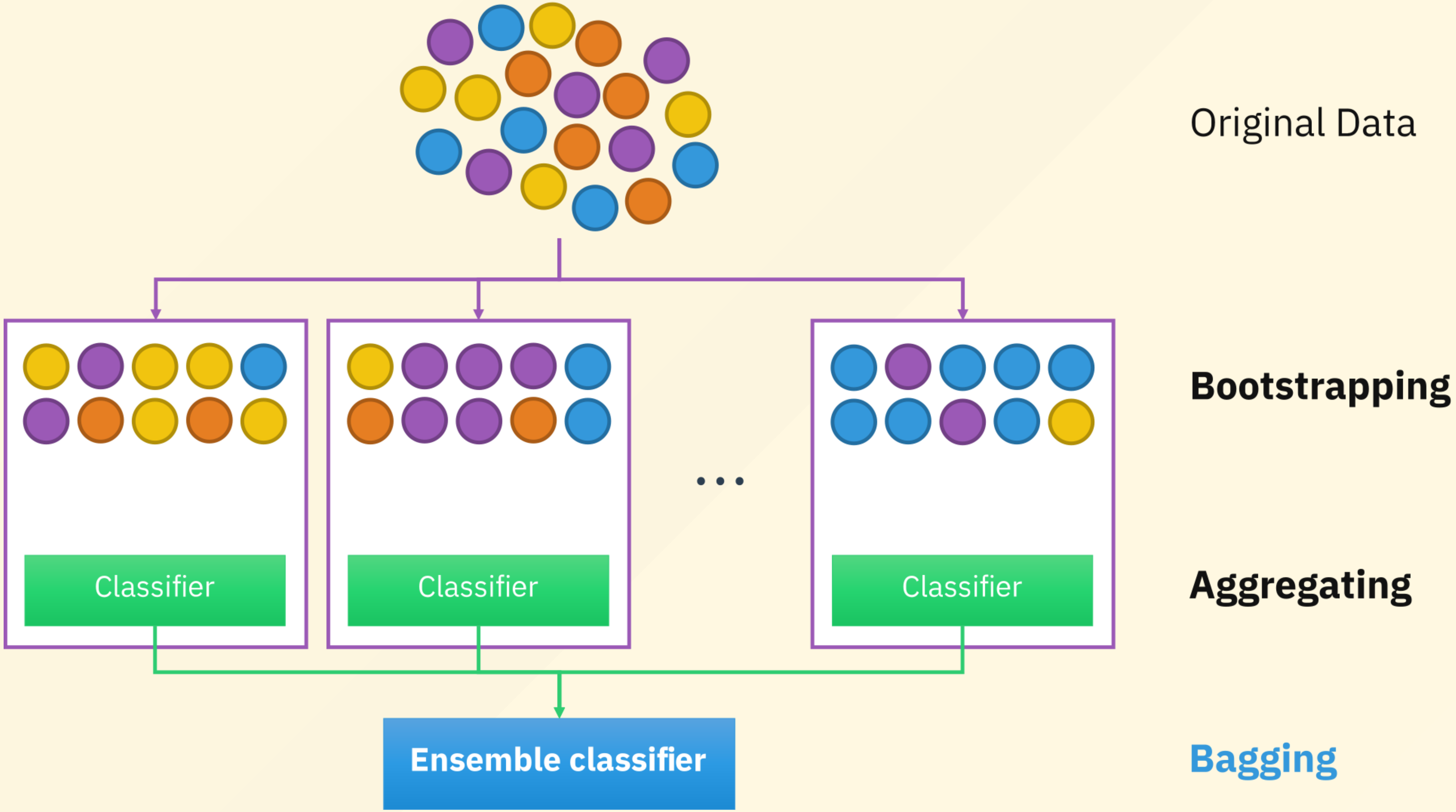
- ensemble models
- two way to do it:
  - bagging
  - boosting
    - adaboost
    - gradient boosting
- pros and cons of each

ensemble modeling

- the first **serious** models we learn about
- **wisdom of the crowd** - train lots of models (called **base learners**) and create a super-model (not a technical term)
- often base learners are **tree-based**: decision trees
- **bagging** and **boosting** are two common ways we can do it
- it's simple to implement, but has high **computational cost** so implementing it **efficiently** is not that simple
- some versions of ensemble models, like XGBoost stand among the modern giants and still win **Kaggle competitions**

ensemble modeling: bagging

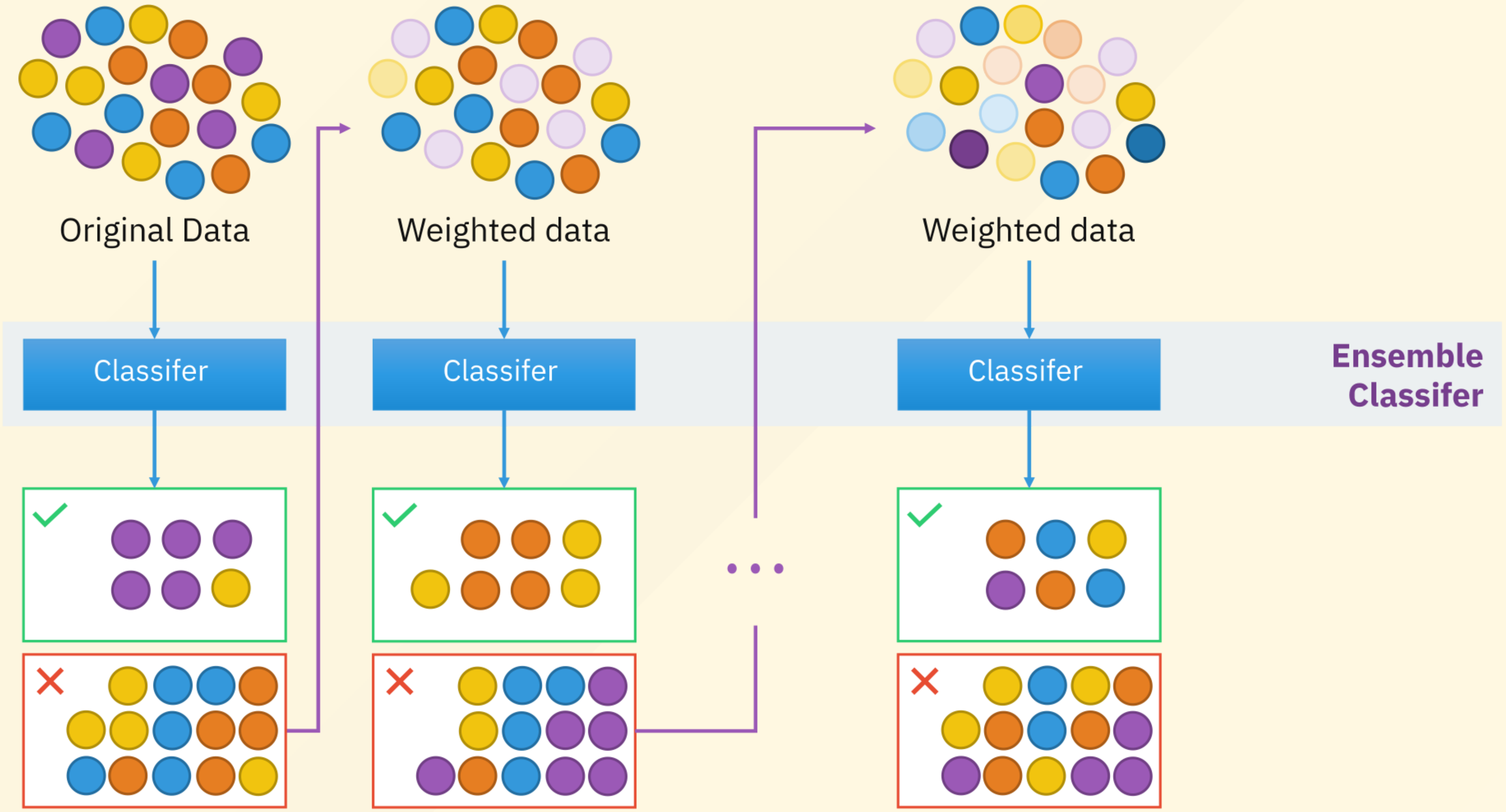
- **bagging** means trains lots of models **independently**
- can be done **in parallel** to decrease runtime
- each model returns a prediction and we **aggregate** them:
  - **regression:** simple average, weighted average based on accuracy
  - **classification:** majority vote, some other voting scheme
- goal is to **decrease variance**: suitable for complex base learners, i.e. base-learners that intentionally overfit
- bagging **dampen** the high variance due to overfitting



source: [Wikipedia]

ensemble modeling: boosting

- **boosting** means train lots of models **sequentially** (the next model depends on previous one)
- cannot be done in parallel, but there are efficient implementations of it such as XGBoost
  making every learner **better than the previous** at learning the examples (**rows**) the previous learner **didn't learn well**
- tries to **decrease bias**: suitable for simple models where we let each base learner be biased but in a different way
- boosting **overcomes** the high-bias due to underfitting



source: Wikipedia

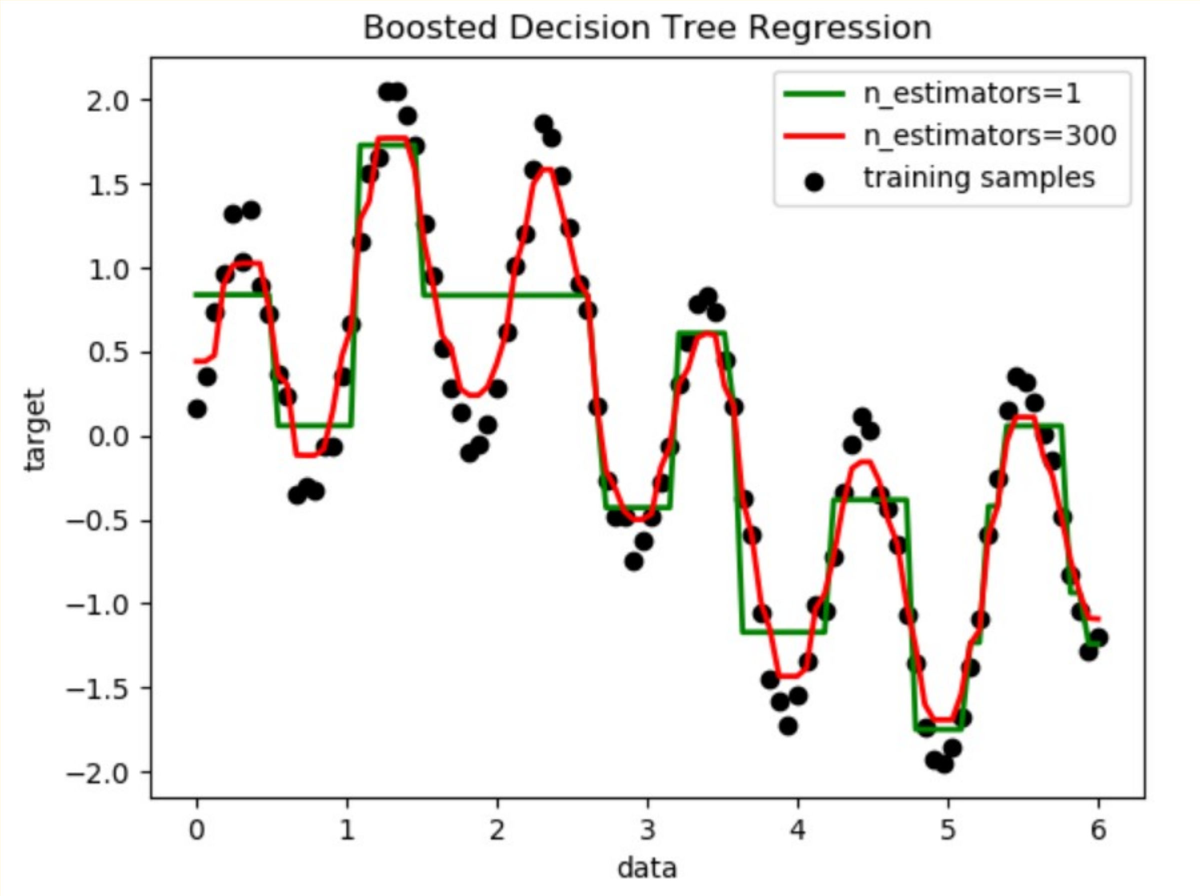
break time

two ways to boost

- **adaboost:**
  - each base learner learns from a **weighted sample** of the data
  - the data for the next learner is sampled using the previous learner's prediction error as **sampling weight**
- **gradient boosting:**
  - next learner is trained to predict the **residual** (prediction error) of previous learner
  - apply a **shrinkage rate** to let learning *decay* over each iteration
  - the final prediction is the **sum of all previous predictions**

boosted regression example

- numeric target
- one numeric feature
- tree prediction is like a **step function**
- ensemble of trees prediction looks much **smoother**

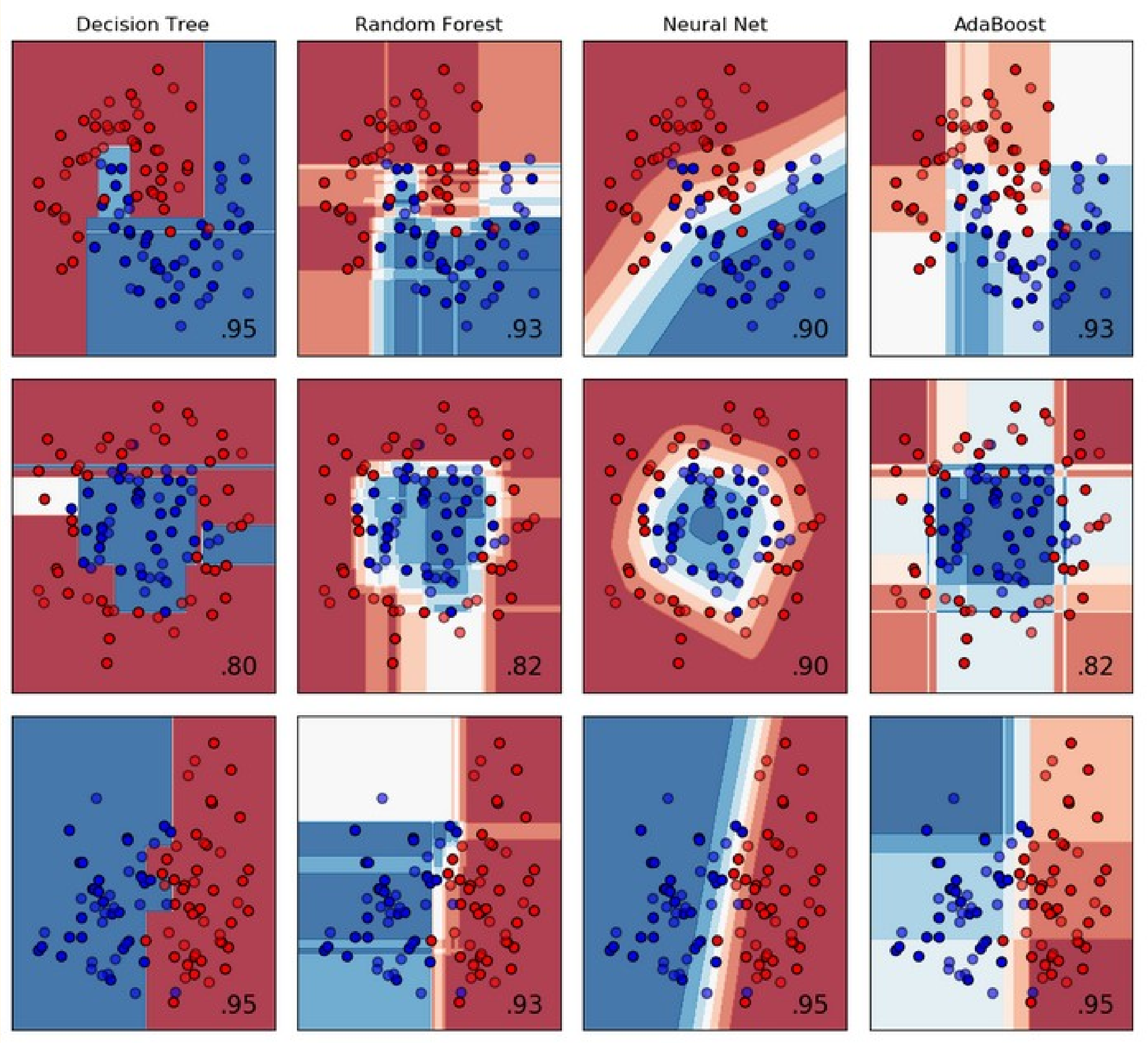


source: https://scikit-learn.org

model comparison

- binary target
- two numeric features
- decision boundary **spiral** (top), **circular** (middle), and **linear** (bottom)
- **shading** indicates model's **confidence** in its prediction

source: https://scikit-learn.org



the end