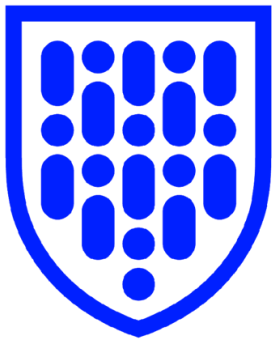


Institute of Data

2021

# Data Science and AI

Module 8

## Ensemble Methods

Institute of Data

# Agenda: Ensemble Methods

- Introduction
- **Bagging**
- **Boosting**
- **Stacking**

Institute of Data

# Introduction

- Introduction
- **Hypothetical Scenario**
- Overview
- **Ensemble Theory**
- Ensembles
- Error in Ensembles (**Variance vs. Bias**)

# Introduction

- Ensemble modelling is a powerful way to **improve the performance** of models
- Ensemble learning can achieve over and above **various models**
- Ensemble models used in competitions like Kaggle have produced **winning results**
- Ensemble learning is a **broad topic** and imagination is the limit

# Hypothetical Scenario: Invest in a company

- Assessing future performance: Approach **experts** with **diverse** experience to get advice on whether the stock price will increase by more than 6% p.a. or not
    a. **Employee**: Knows the internal functionality of the company and has the insider information about the inner workings of the firm, but lacks a broader perspective on how competitors are innovating, how the technology is evolving and what is the impact of this evolution for the company's product. **Has been right 70% times in the past**
    b. **Financial Advisor**: Has a broader perspective on how companies strategy compares, however, lacks a view on how the company's internal policies are fairing. Has been right 75% times in the past
    c. **Stock Market Trader**: Has observed the company's stock price over the past 3 years, knows how the overall market is performing and trends. Has developed a firm intuition on how stocks might vary over time. Has been right 70% times in the past

# Hypothetical Scenario: Invest in a company

- Assessing future performance: Approach experts with diverse experience to get advice on whether the stock price will increase by more than 6% p.a. or not

    d. **Employee of a competitor**: Knows the internal functionality of the competitor firms and is aware of specific changes which are yet to be made, but does not know the company and the external elements which can relate the growth of competitor. Has been right  60% of times in the past

    e. **Market Researcher**: Analyses the customer preference of the company's product over others and how this is changing with time. Is unaware of the changes the company brings because of alignment to its own goals. Has been right 75% of times in the past

    f. **Social Media Expert**: Can help to understand how the company has positioned its products in the market, and how the sentiment of customers is changing over time but is unaware of any details beyond digital marketing. Has been right 65% of times in the past

# Hypothetical Scenario: Invest in a company

- Given the broad spectrum of sources, it is possible to **combine** the information and make an **informed** decision

- The **combined accuracy** rate in a scenario with all six sources determine that it is a good decision (assuming **independence** between the predictions)

$$1 - 30\% \times 25\% \times 30\% \times 40\% \times 25\% \times 35\%$$

$$= 1 - 0.07875\%$$

$$= 99.92125\%$$

# Hypothetical Scenario: Invest in a company

- **New Scenario**: Six experts, **all employees** of the company from the same division with a propensity of 70% to advocate correctly
- **Assumption**: All the predictions are entirely **independent**, although they are expected to be correlated
- Can the confidence be more than 99%?
  - Intuitively not, **as the same information** is the source of all the predictions
  - The only variation in their advice would be due to personal opinions and collected facts about the firm

# Overview

- Ensemble methods use **various** learning algorithms to obtain **better predictive performance** compared to any of the constituent learning algorithms alone

- Supervised learning algorithms perform the task of finding a suitable hypothesis from a **hypothesis space** to make reasonable predictions for a particular problem
  - It may be very challenging to find a good hypothesis even if the hypothesis space contains well-suited ones for a particular problem

- Ensembles **combine multiple hypotheses** to form a better hypothesis hopefully

# Overview

- The word “ensemble” is used with methods that **generate multiple hypotheses** using the same base learner
  - The expanded idea of various classifier systems also covers combinations of hypotheses that are **not induced by the same base learner**

- Ensemble techniques have also been used in **unsupervised learning**, such as **consensus clustering** or **anomaly detection**

# Overview

- To evaluating the prediction of an ensemble requires **more computation** compared to a single model
  - Fast algorithms such as **decision trees** are commonly used in ensemble methods (e.g. random forests)
  - Slower algorithms can benefit from ensemble techniques as well
  - The extra computation is a way to compensate for poor learning algorithms

# Overview

- There are two groups of Ensemble methods
  - **Sequential**: where the base learners are generated sequentially
    (e.g. AdaBoost)
    - The rationale is to **benefit from the relationship between the base learners**
    - The general performance can be boosted by penalising previously mislabeled examples with higher weight
  - **Parallel**: where the base learners are generated in parallel
    (e.g. Random Forest)
    - The motivation is to exploit independence between the base learners since the error can be reduced dramatically by averaging

# Ensemble Theory

- The **trained ensemble** represents a hypothesis
  - This hypothesis is not necessarily contained within the **original hypothesis space**
  - Ensembles can have more flexibility in the functions they can represent
  - In theory, enable ensembles to overfit on the training data compared to a single model would
  - In practice, ensemble techniques (bagging mainly) tend to **reduce issues** related to **overfitting** of the training values

# Ensemble Theory

- Studies show that ensembles can yield better results when there is **significant diversity** among the models
- Although counter-intuitive, more **random algorithms** (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees)
- Many ensemble methods seek to promote diversity among the models they combine
- Using a variety of **robust learning algorithms** has shown to be more effective than using techniques that attempt to simplify the models to promote diversity

# Ensembles

- An Ensemble is a combination of a **diverse set of learners** (individual models) put together to improve on the stability and predictive power of the model
- The term Ensemble Learning in the above example is used to describe the combination of all the predictions together

# Ensembles

- How to get a **different set of learners** as models can be different from each other in any combination of factors such:
  - The **difference in population**
  - The difference in the **hypothesis**
  - The difference in the **modelling technique**
  - The difference in **initial seed**

# Error in Ensembles (Variance vs. Bias)

- Suppose we have a set of points $x_1, x_2, \ldots, x_n$ and real values $y_i$
- Assume we have a function with some noise $y = f(x) + \varepsilon$ with zero mean and variance $\sigma^2$
- Assume we want to find a function $\hat{f}(x)$ that approximates the real function. We express the error of our function by measuring the 'mean squared' error. i.e. we want $\left(y - \hat{f}(x)\right)^2$ to be minimum. Of course, we cannot do this perfectly, because the real function itself has noise.
- So, we can break down our 'expected' error to be:

$$Error(x) = \textbf{Bias}^2 \textbf{ + Variance + Irreducible Error}$$

where:

$$Bias\left[\hat{f}(x)\right] = E\left(\left[\hat{f}(x)\right]\right) - f(x)$$

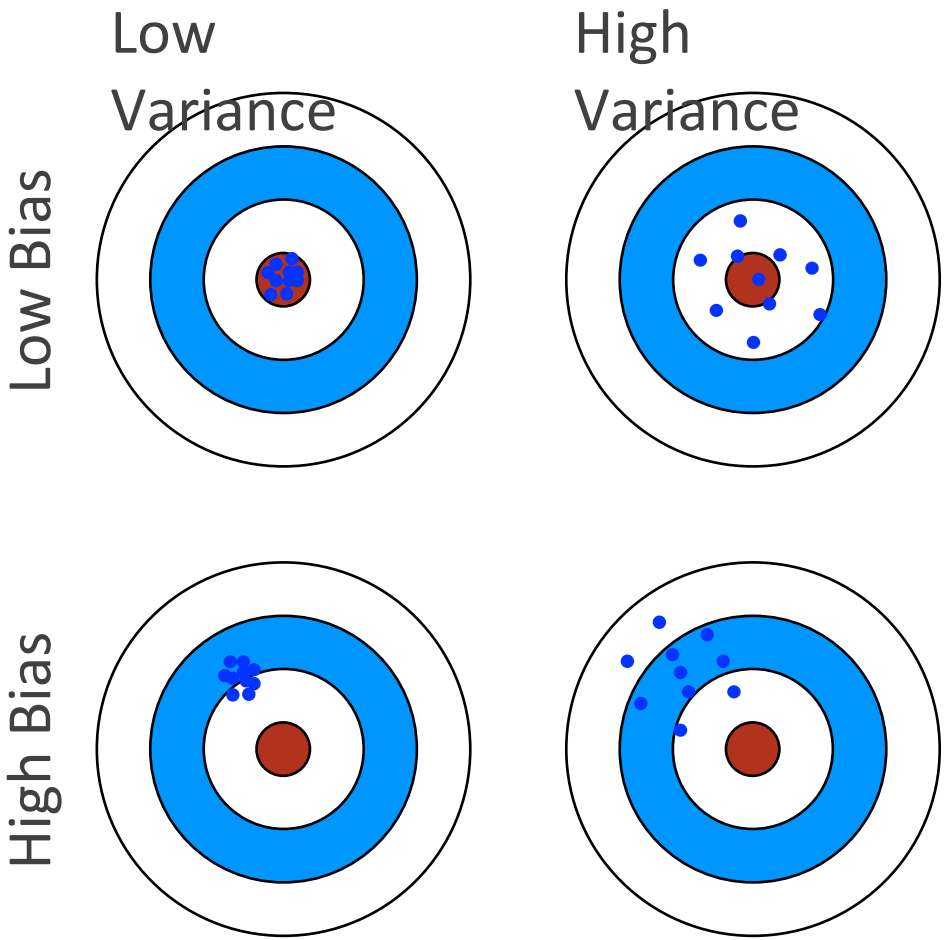
$$Variance\left[\hat{f}(x)\right] = E\left[\left(\hat{f}(x)\right)^2\right] - E\left[\hat{f}(x)\right]^2$$

$$Irreducible\ Error = \sigma^2$$

# Error in Ensembles (Variance vs. Bias)

- **Bias** measures the difference between the predicted and actual values **on average**
  - A high bias means an under-performing model that does not capture important trends
- **Variance** quantifies how are the prediction made on **same observation** different from each other
  - A high variance model **overfits** on the training population and performs poorly on any observation **beyond training data**

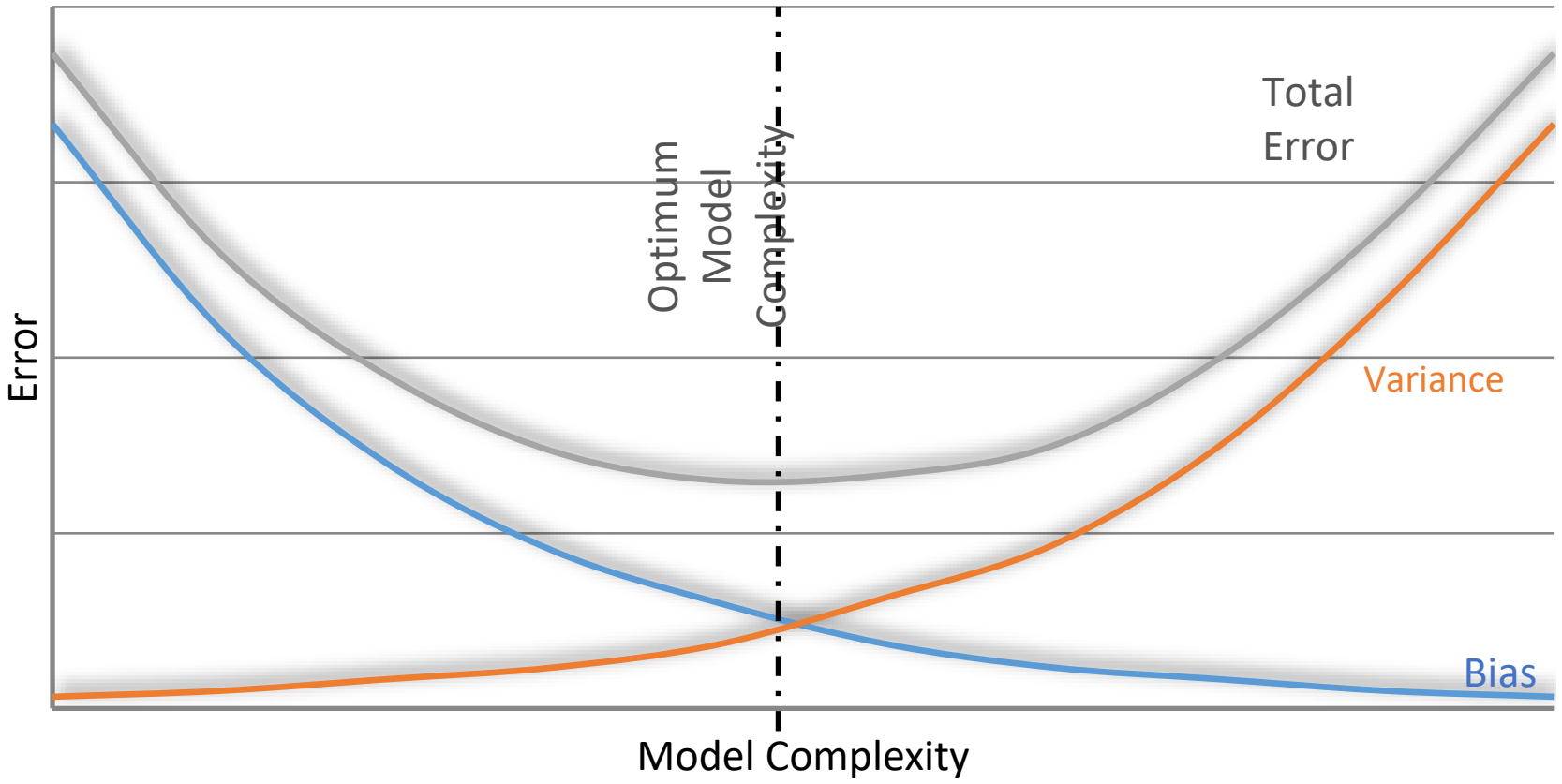
# Error in Ensembles (Variance vs. Bias)



# Error in Ensembles (Variance vs. Bias)

- Typically, as the **complexity** of the model increases, there is a reduction in error due to **lower bias** in the model
  - This only happens up to a particular point
- As the model becomes more complex, it becomes prone to **overfitting**, the model starts suffering from **high variance**
  - An ideal model should keep a **balance** between these bias and variance
  - This is known as the trade-off management of bias-variance errors
- Ensemble learning can manage this **trade-off** analysis

# Error in Ensembles (Variance vs. Bias)



Institute of Data

# Bagging

- Overview
- Algorithm
- Usage
- Methods
- Pros and Cons
- Example

# Bagging - Overview

- Bagging stands for **Bootstrap Aggregation**.
- In statistics, bootstrapping is any test or metric that relies on random sampling **with replacement**.
- One way to reduce the **variance** of an estimate is to **average** together **multiple estimates**
- For example, train M different trees on different **subsets** of the data (**chosen randomly with replacement**) and compute the ensemble output by averaging the output of the trees

# Bagging Overview

- Bagging uses **averaging for regression** and **voting for classification** when aggregating the outputs of base learners

- Bagging uses bootstrap sampling to obtain the training values for the base learners

# Bagging overview in other words

- The idea is to combine the results of **multiple models** (for instance, all decision trees) to get a generalised result
    - Question: Is the combination of all the models on the same set of data useful?
        - There is a high chance that these models give the **same result** since they are getting the same input
    - How can this problem be solved?
        - One of the techniques is **bootstrapping**

# Bagging overview in other words

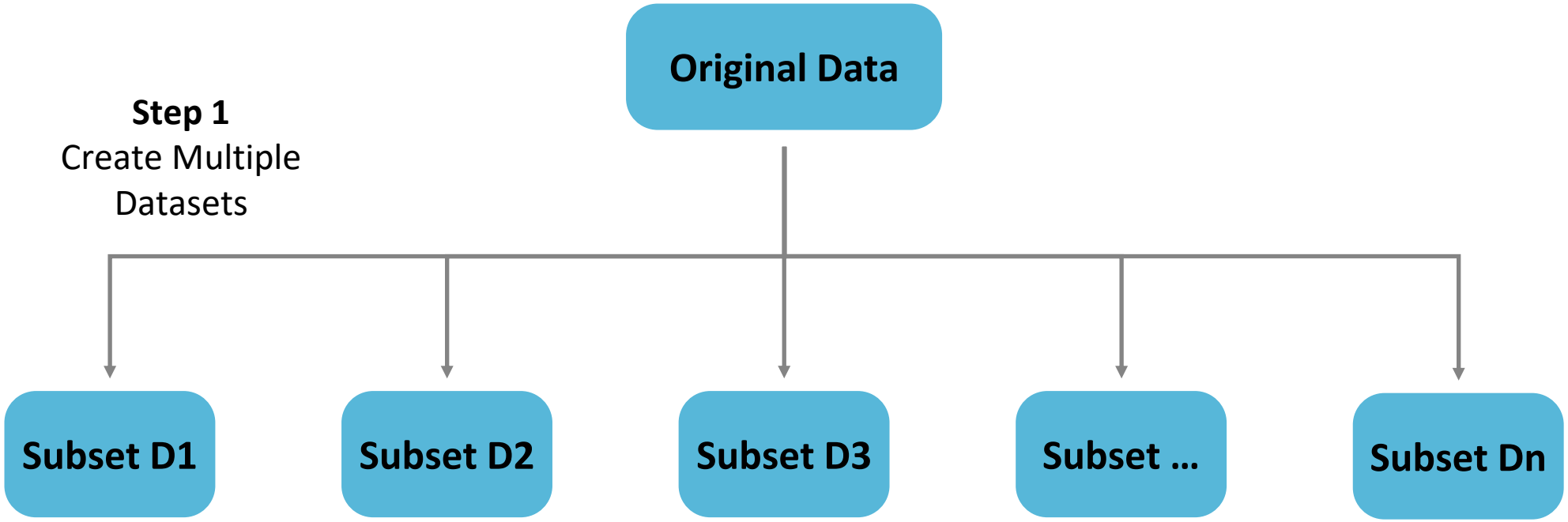
- Bootstrapping is a **sampling** technique of creating subsets of observations from the original dataset, with replacement
  - **The size of the subsets is the same as the size of the original set**

- Bagging uses these subsets (**bags**) to get a fair idea of the distribution (complete set)
  - **The size of subsets created for bagging may be less than the original set**

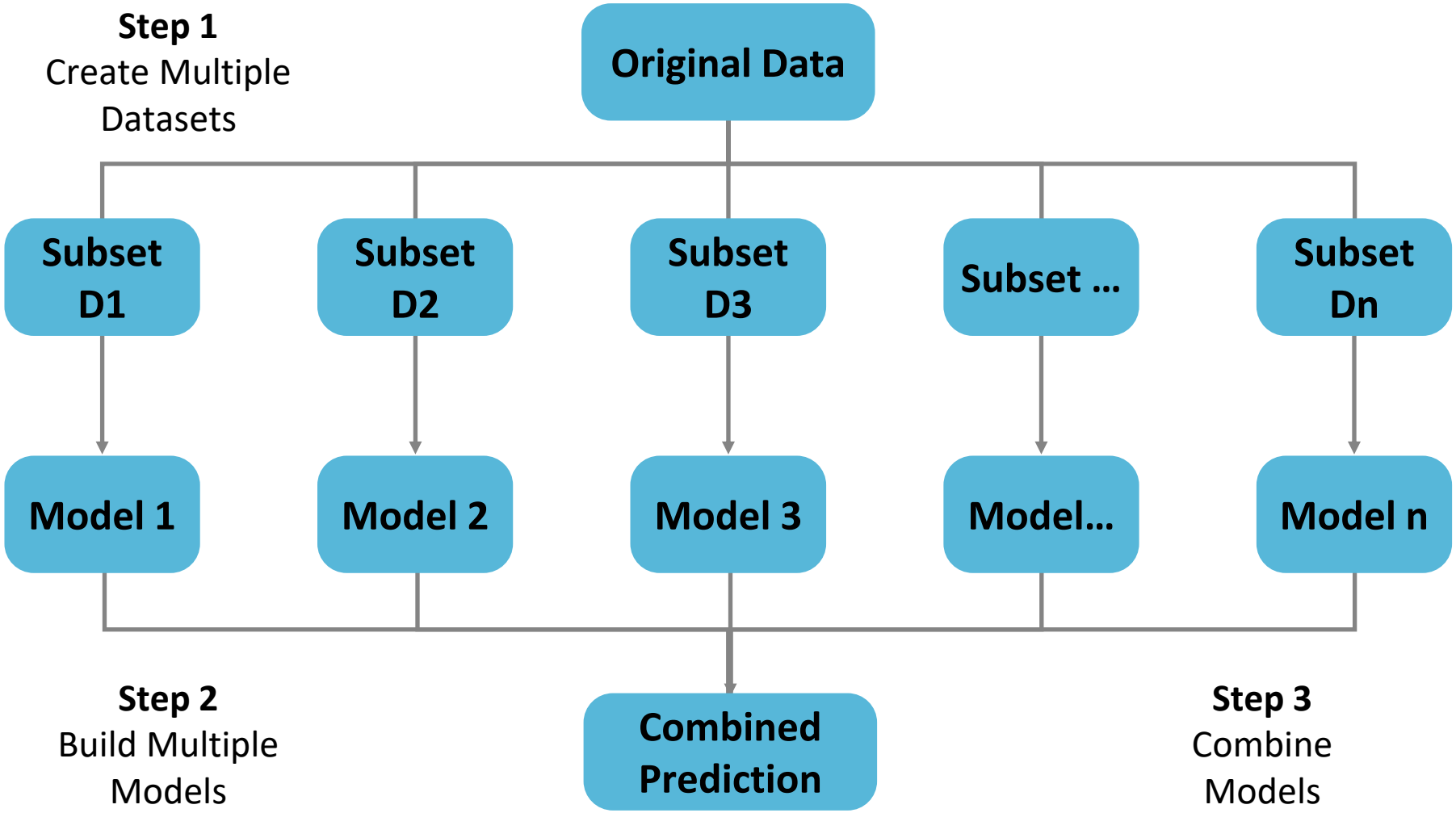
# Bagging Algorithm

1. The original dataset is the source of various **subsets** by sampling with replacement
2. A base model (**weak model**) is created on each of these **subsets**
3. The models are **independent** and can be run in parallel
4. The combination of the predictions from all the models determine the final predictions

# Bagging Algorithm



# Bagging Algorithm



# Bagging Usage

- **Forests of randomised trees** are a type of ensemble algorithm
- In **Random Forests**, each tree is built by bootstrap sampling (drawn with replacement) from the training set
- Also, instead of using all the features, a random **subset of features** is selected, further randomising the tree
- The **bias** of the forest **increases** slightly as a consequence, but the **variance decreases** with the averaging of less correlated trees, producing an overall better model

# Bagging Usage

- Algorithm randomness goes one step further in vastly randomised trees
  - The **splitting thresholds** are randomised
    - Thresholds are drawn at random for each available feature (opposed to the most discriminative threshold), and the best of these randomly-generated thresholds is picked as the splitting rule
  - This usually allows for an extra reduction of the variance of the model, with a slightly higher increase in bias

# Bagging Methods

- Some Bagging algorithms
  - Bagging meta-estimator
  - Random forest

# Bagging Pros

- Robust against **outliers and noise**
- Fast runtime
- **Reduces variance** and typically avoids overfitting
- Offers most of the **advantages of trees** like automatic selection of variable importance, handling missing values, and accommodating highly non-linear interactions
- Easy to use with limited well-established default parameters and works well off-the-shelf requiring **little additional tuning**

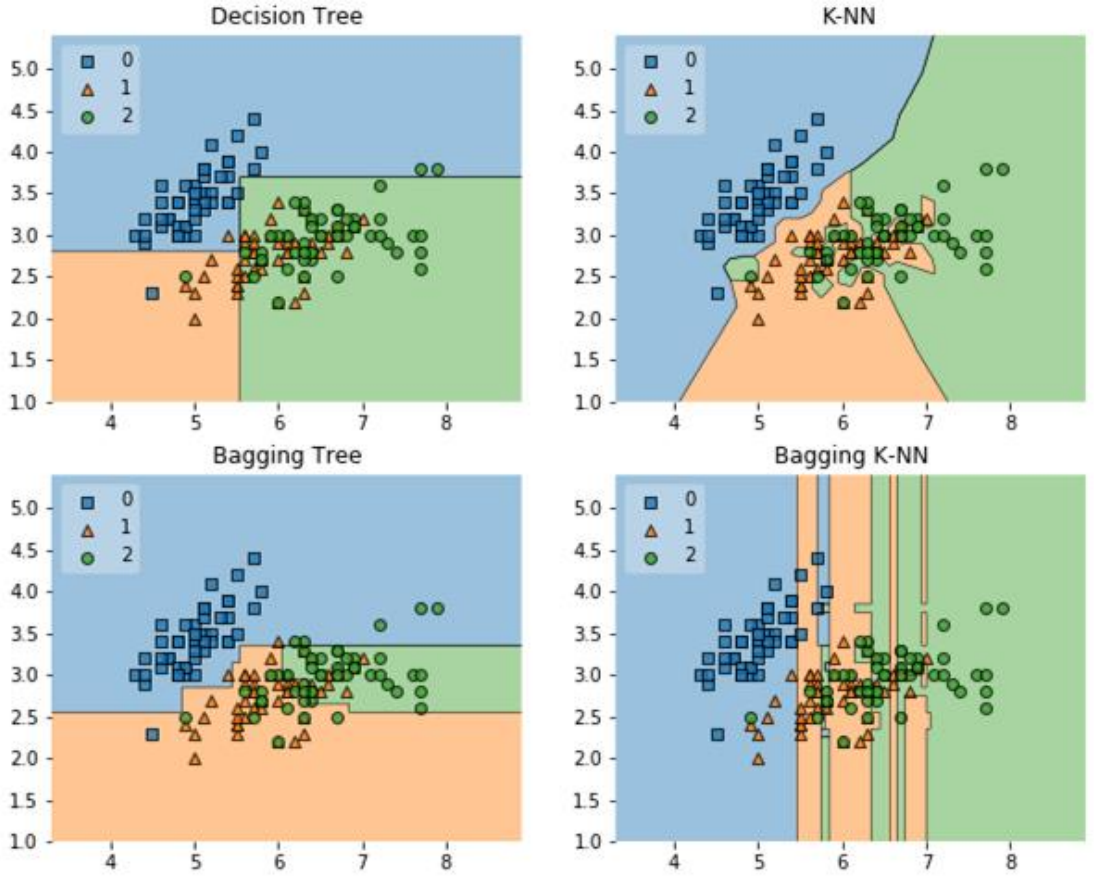
# Bagging Cons

- The complexity of multiple trees **removes transparency** (**black box**)
- Can be slow to compute as complexity increases

# Bagging Example

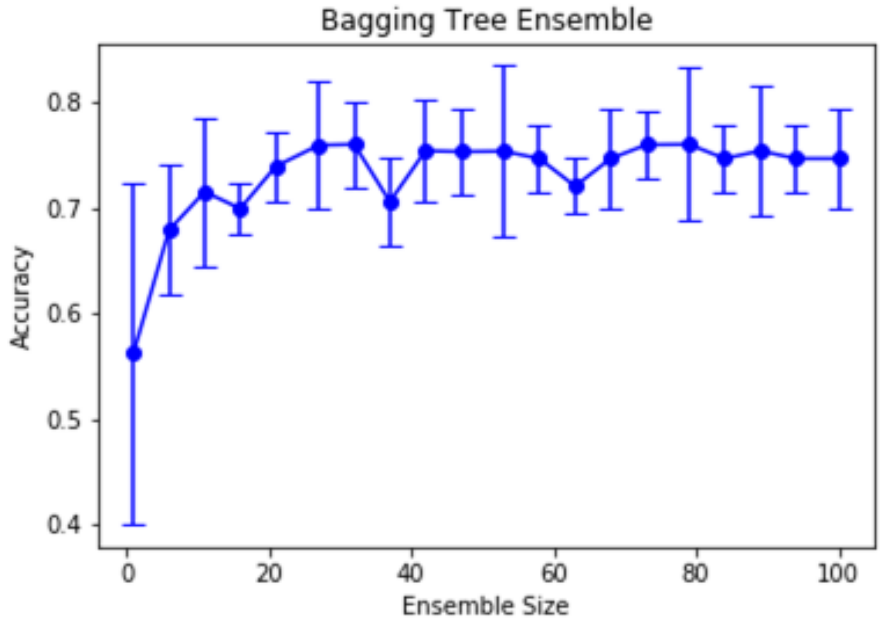
- Classifying the Iris dataset, choose two base estimators
  - Decision tree and a K-NN classifier
- Decision boundary of the base estimators and their bagging ensembles

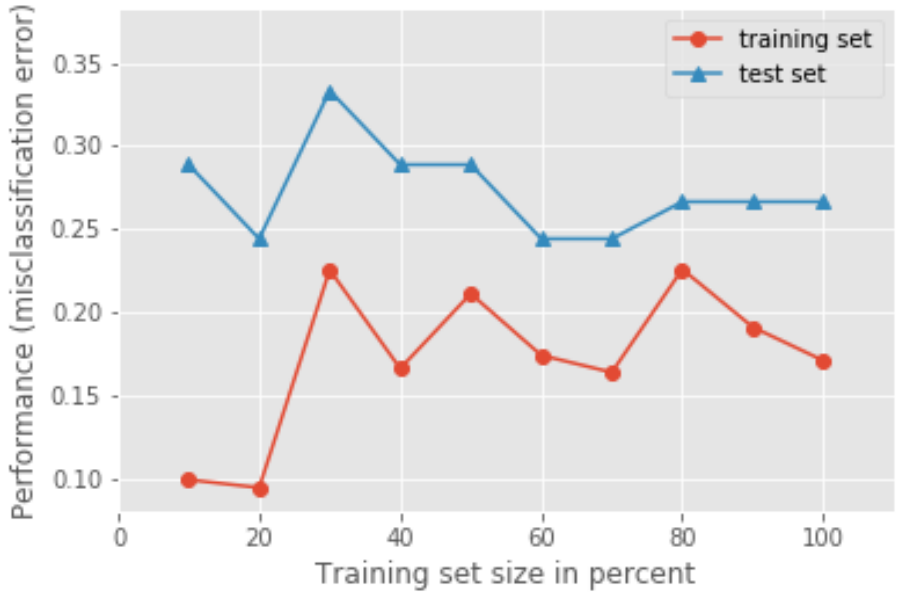
| | Accuracy | |
|---|---|---|
| | Mean | Standard Deviation |
| Decision Tree | 0.63 | ±0.04 |
| K-NN | 0.70 | ±0.02 |
| Bagging Tree | 0.70 | ±0.08 |
| Bagging K-NN | 0.61 | ±0.02 |



# Bagging Example

- K-NN are less affected by perturbation on training samples being called stable learners
- Combining **steady learners** does not improve results as the ensemble does not help improve generalisation performance



# Demo 8.2: Bagging

- Purpose
  - Understand the concept and potential of Bagging
- Resources
  - Sample data from SciKit-Learn
- Materials
  - Jupyter Notebook (Demo-8_2)

# Lab 8.1: Bagging

- Purpose
  - Work with Bagging
  - Practice some coding in Python
- Resources
  - Sample data from SciKit-Learn
- Materials
  - Jupyter Notebook (Lab-8_1)

Institute of Data

# Boosting

- Overview
- Algorithm
- Usage
- Methods
- Pros and Cons
- Example

# Boosting Overview

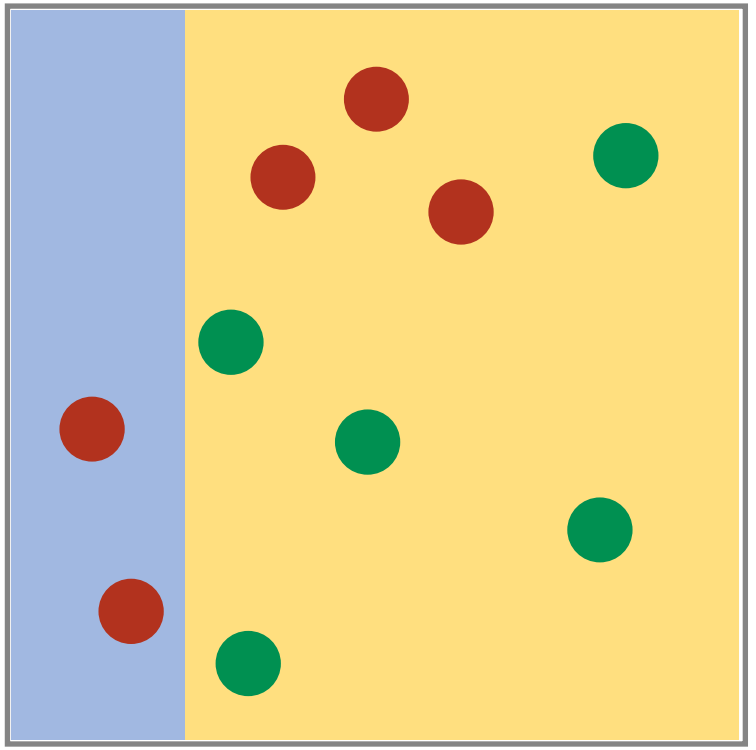
- Boosting refers to a category of algorithms that can convert weak learners to strong learners
  - The principle of boosting is to fit a sequence of weak learners **to weighted versions** of the data
  - Weak learners are models that are only slightly better than random guessing, such as small decision trees
  - **Examples that were misclassified by earlier rounds get more weight**

# Boosting Overview

- The sequential training of base learners on **a weighted version** of the data is the main difference between boosting and committee methods (like bagging)
- The predictions are then **aggregated by a weighted sum** (regression) or weighted majority vote (classification) to produce the final prediction.
- Boosting is processed **in sequence**, so the next step's model attempts to **correct the errors** of the model form the previous step
- The following models are **dependent** on the previous model

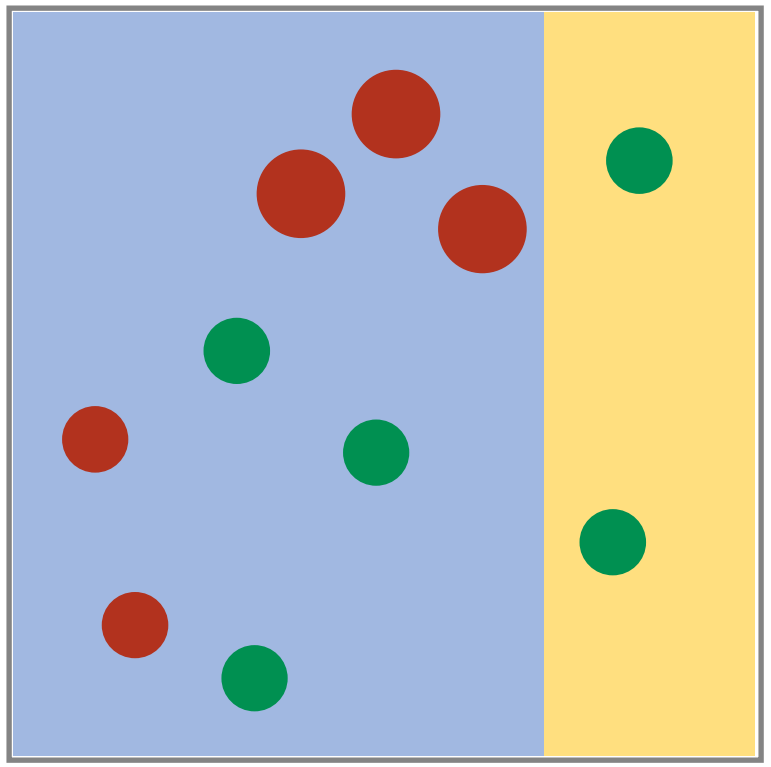
# Boosting Algorithm

- The original dataset is the source of a subset
- Initially, the weights of all data points are equal
- A base model is created from this subset
- The base model serves to make predictions on the whole dataset



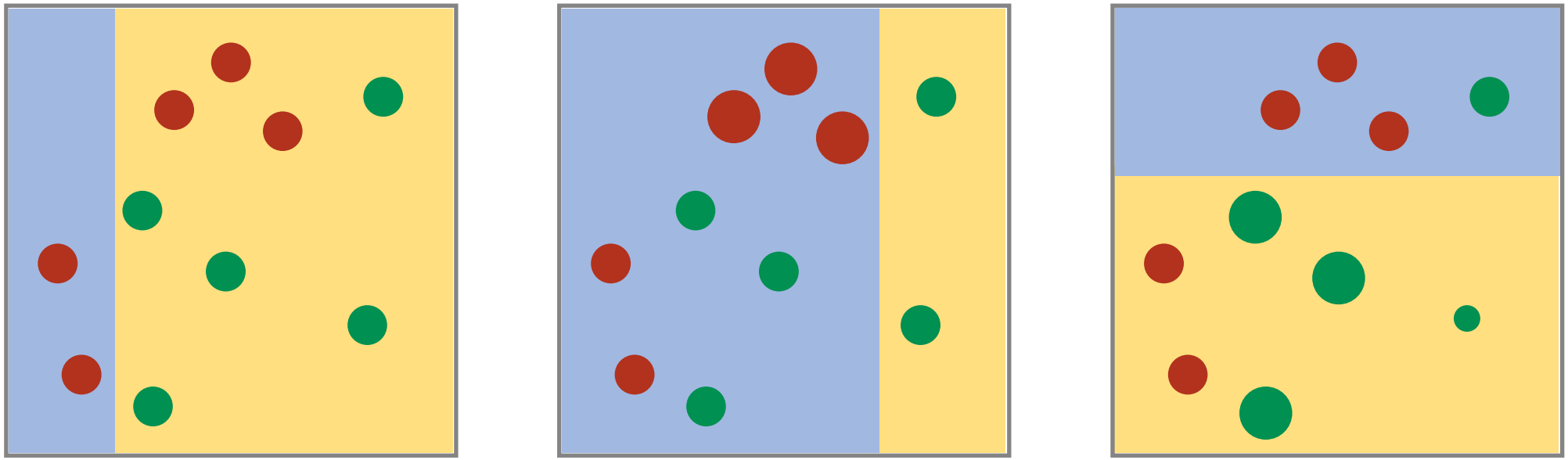
# Boosting Algorithm

- The predicted and actual values are used to calculate the error
- The observations get higher weights if their predictions are incorrect
  - The three misclassified red points get bigger weights
- A new model is created producing predictions from the dataset
  - The model tries to correct the errors from the previous model



# Boosting Algorithm

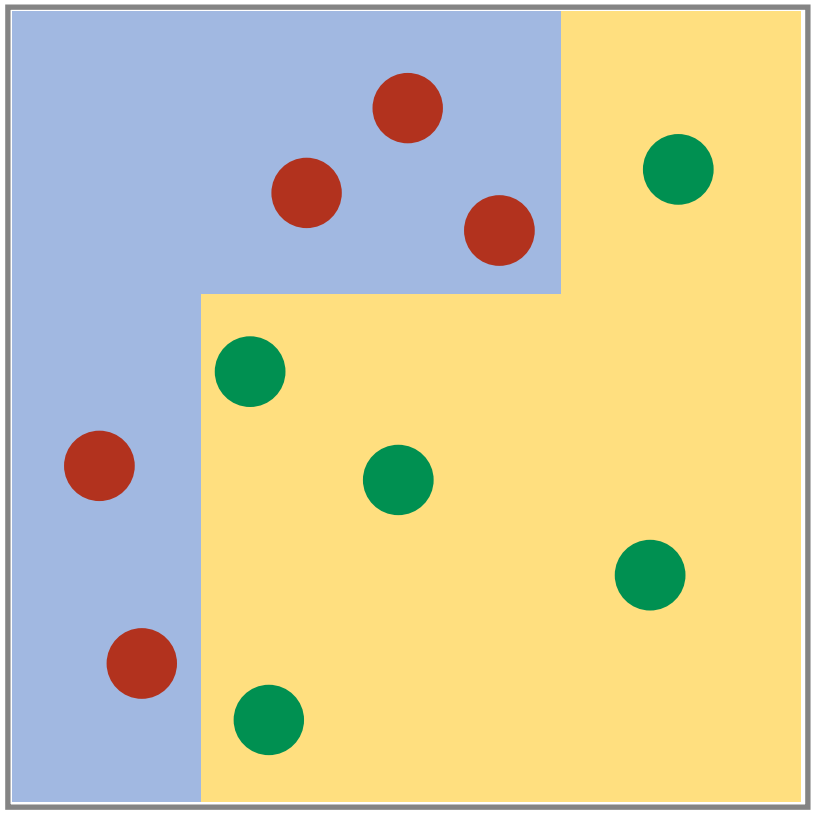
8. Each of multiple new models corrects the errors of the previous model



# Boosting Algorithm

- The weighted mean of all the previous models (weak learners) create the finished model (strong learner)

- The separate models would not have a great performance on the entire dataset, but combined they work well for parts of the dataset, so each model boosts the performance of the ensemble



# Boosting Algorithm (AdaBoost)

- AdaBoost (Adaptive Boosting)
- The **base classifier $y_1(x)$** is trained using equal weighted coefficients
- The weighted coefficients in subsequent rounds are
  - **Increased** for data points that are classified **incorrectly**, and
  - **Decreased** for data points that were classified **correctly**
- Each of the base classifiers has an epsilon quantity that represents a **weighted error rate**
- The weighting coefficients alpha attributes **bigger weight** to the **more accurate** classifiers

# Boosting Usage

- Gradient Tree Boosting is a generalisation of boosting to arbitrary differentiable loss functions
- It can solve both regression and classification problems
- Gradient Boosting builds the model in a sequential way

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

# Boosting Usage

- The decision tree $h_m(x)$ is chosen at each stage to minimise a loss function $L$ given the current model $F_{m-1}(x)$

$$F_m(x) = F_{m-1}(x) + \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x) + \gamma_m h_m(x))$$

- The type of loss function used is the difference between classification and regression algorithms

# Boosting Methods

- Some Boosting algorithms
  - AdaBoost (Adaptive boosting
  - GBM (Gradient boosting): uses differentiable loss function.
  - XGBoost (eXtreme Gradient Boosting)
  - Light GBM

# Boosting Pros

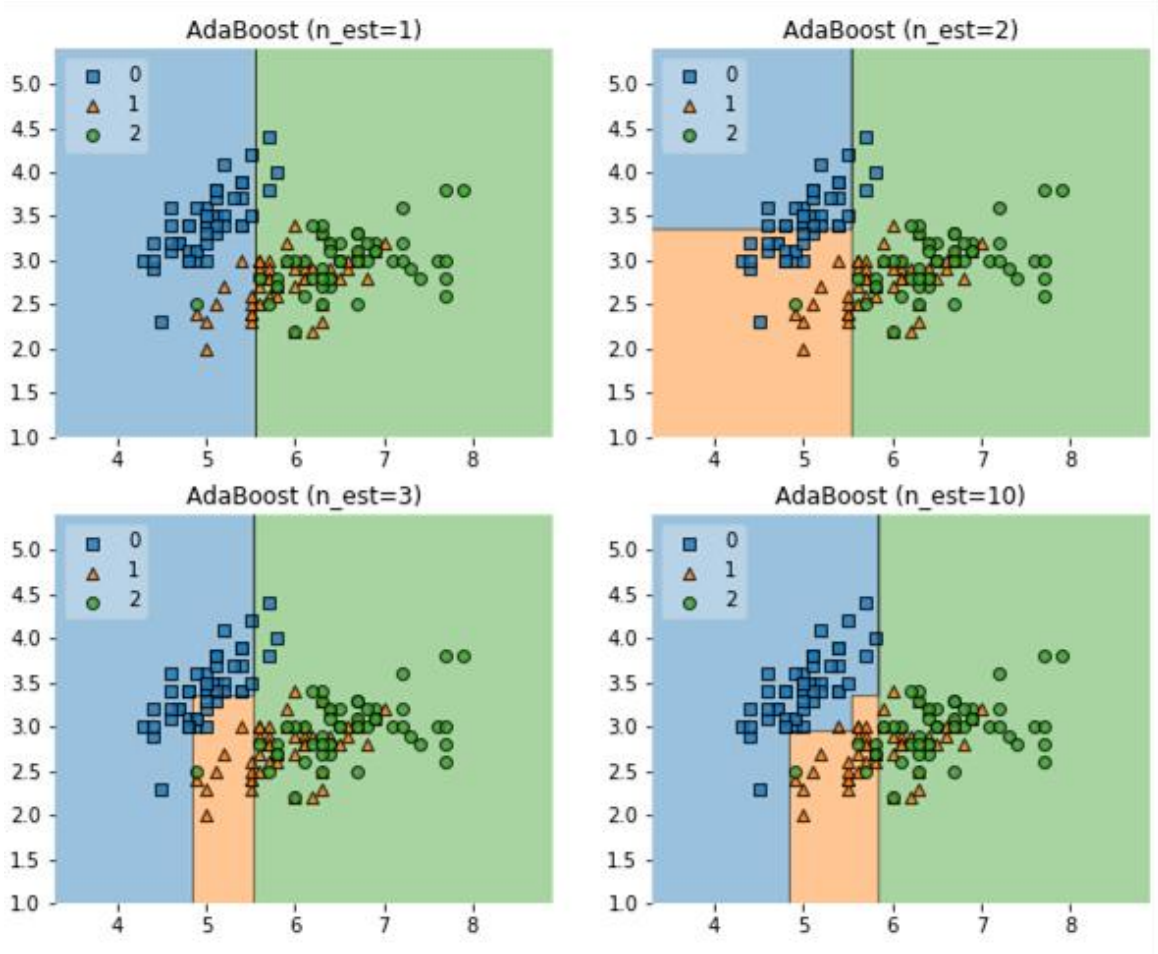
- Often the best possible model
- Directly optimises the cost function

# Boosting Cons

- Not **robust** against **outliers and noise**
- Can overfit
- Need to find a proper stopping point
- Several **hyper-parameters**
- The complexity of multiple trees lacks transparency (**black box**)

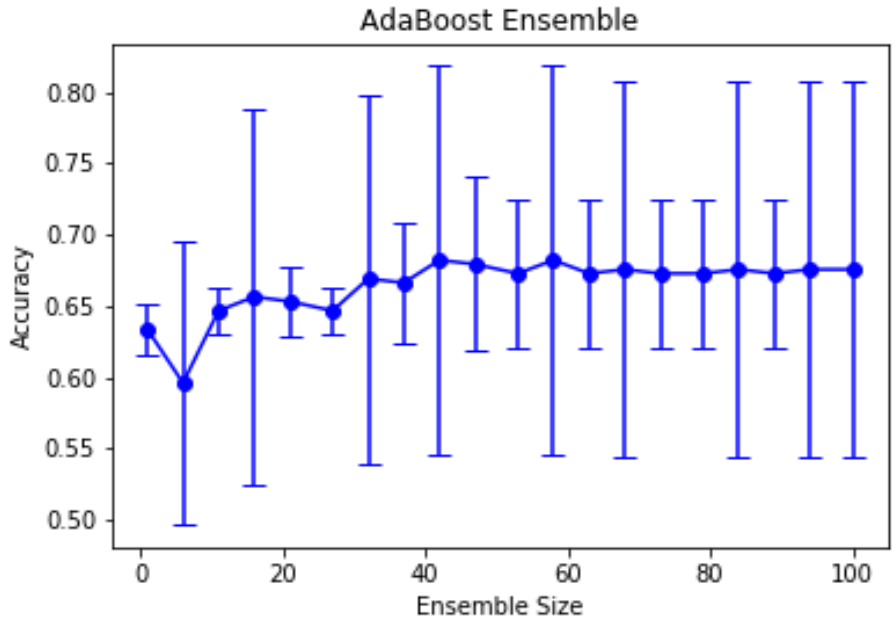
# Boosting Example

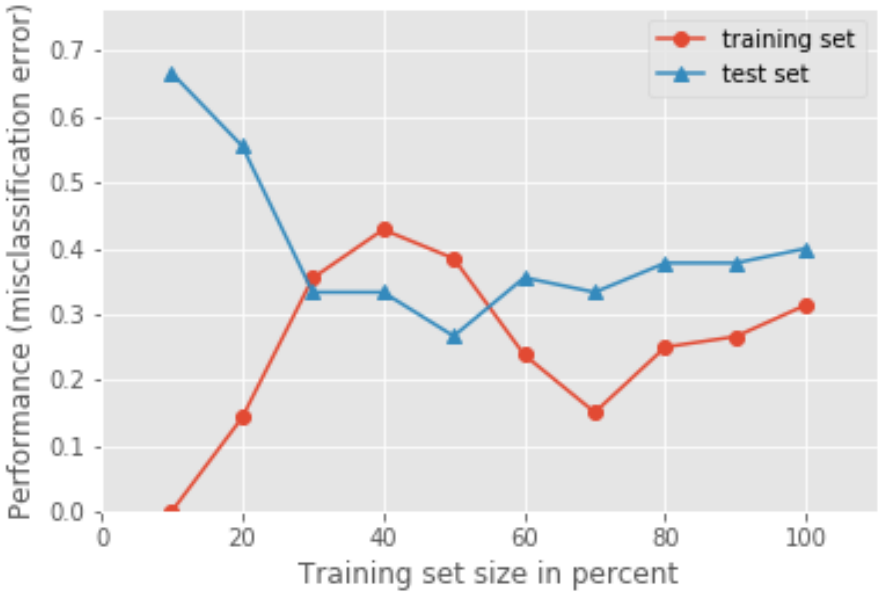
- Each base learner is a depth 1 decision tree which separates the space into two regions based on a feature threshold that partitions separated by a linear decision surface that is parallel to one of the axes



# Boosting Example

- The test accuracy grows, up to a point, with the size of the ensemble (on the left)
- The figure on the right shows the learning curves for training and testing data



# Demo 8.3: Boosting

- Purpose
  - Understand the concept and potential of Boosting
- Resources
  - Sample data from SciKit-Learn
- Materials
  - Jupyter Notebook (Demo-8_3)

# Lab 8.2: Boosting

- Purpose
  - Work with Boosting
  - Practice some coding in Python
- Resources
  - Sample data from SciKit-Learn
- Materials
  - Jupyter Notebook (Lab-8_2)

Institute of Data

# Stacking

- Overview
- Algorithm
- Pseudocode
- Example

# Stacking Overview

- Stacking is an ensemble learning approach that combines multiple regression or classifications models via a meta-regressor or a meta-classifier
- The **outputs of the base level model serve as features to train the meta-model**
- The base level models are trained based on **a complete training set**
- The base level often consists of **different learning algorithms** and therefore stacking ensembles are often heterogeneous

# Stacking overview in other words

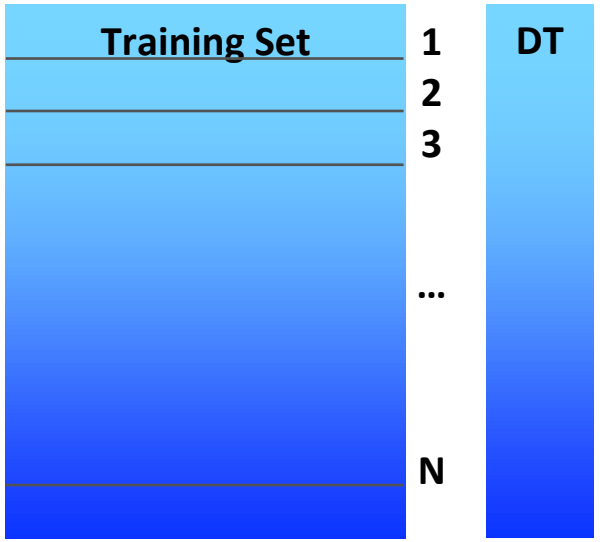
- Stacking is an ensemble learning approach that **uses predictions from various models to construct a new model**
  - A base model such as Decision Tree, K-NN or SVM
- The new model can then make predictions on the test set

# Stacking Algorithm

1. The train set is split into ***N*** parts

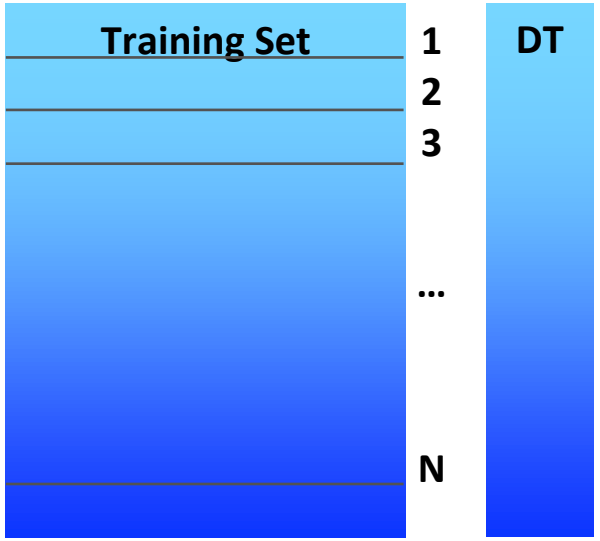

# Stacking Algorithm

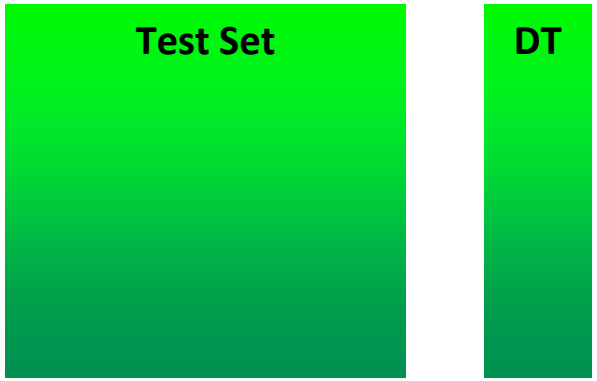
2. A base model (like CART) is fitted on ***N-1*** parts, and predictions are made for the ***N*** part
   - This applies to each part of the train set




# Stacking Algorithm

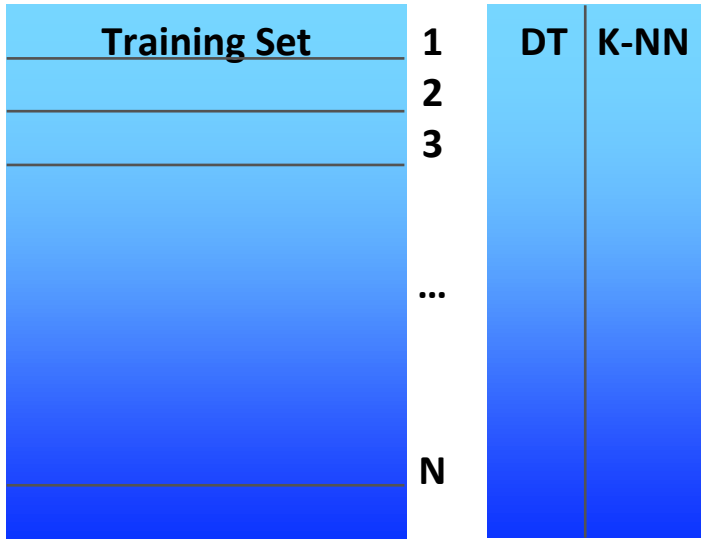
3. The base model (decision tree) is then fitted on the whole train dataset
4. Predictions are made on the test dataset using this model

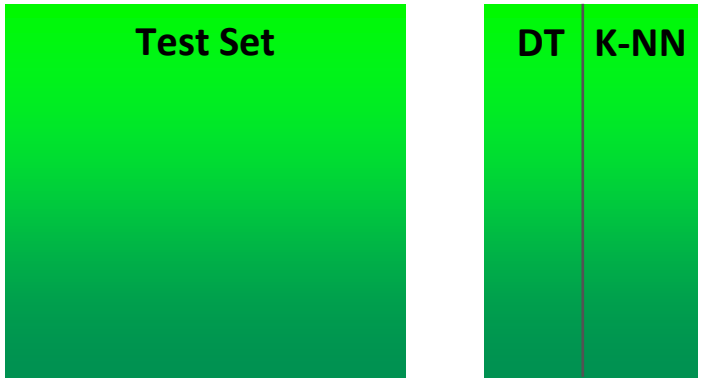


# Stacking Algorithm

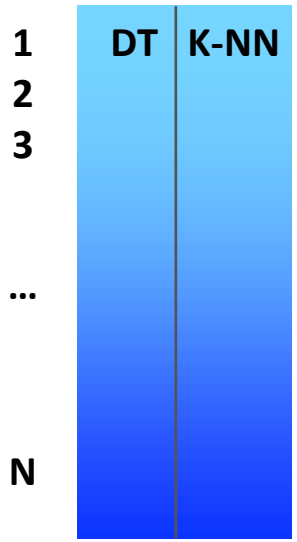
5. All other base models repeat steps 2 to 4 resulting in another set of predictions for both the train set and test set

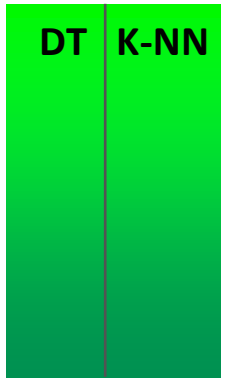


# Stacking Algorithm

6. The predictions from the train dataset serves as features to build a new model
7. The new model is used to make final predictions on the test prediction set



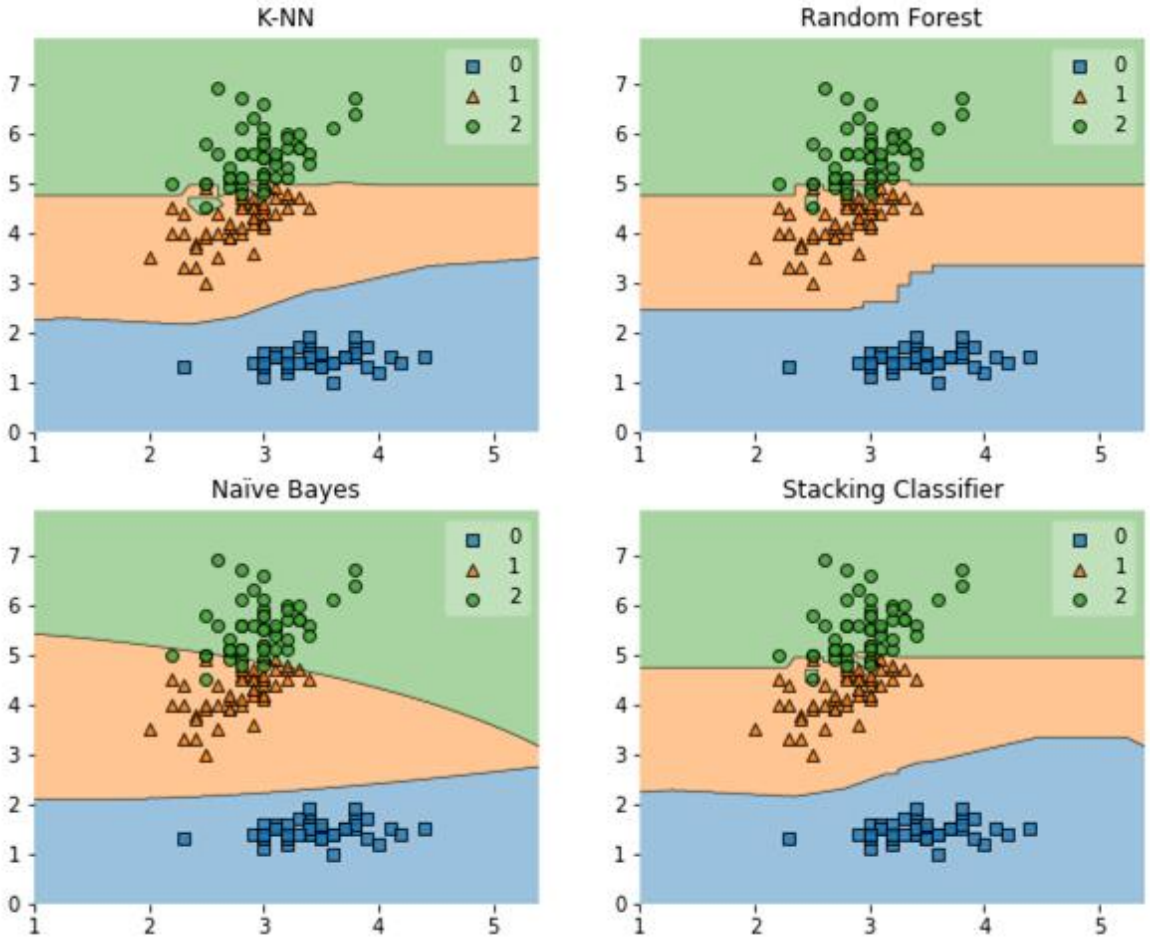
# Stacking Algorithm

1: $Input: training data D = \{x_i, y_i\}_{i=1}^{m}$
2: $Output: ensemble classifier H$
3: $\#Step1: learn\ base\ level\ classifier$
4: $for\ t = 1 to T$
5: $learn\ h_t based\ on\ D$
6: $\#Step2: construct\ new\ dataset\ of\ predictions$
7: $for\ i = 1\ to\ m$
8: $D_h = \{x_i', y_i\}, where x_i' = \{h_1(x_i), \cdots, h_T(x_i)\}$
9: $\#Step3: learn\ a\ meta\ classifier$
10: $learn\ H\ based\ on\ D_h$
11: $return\ H$

# Stacking Example

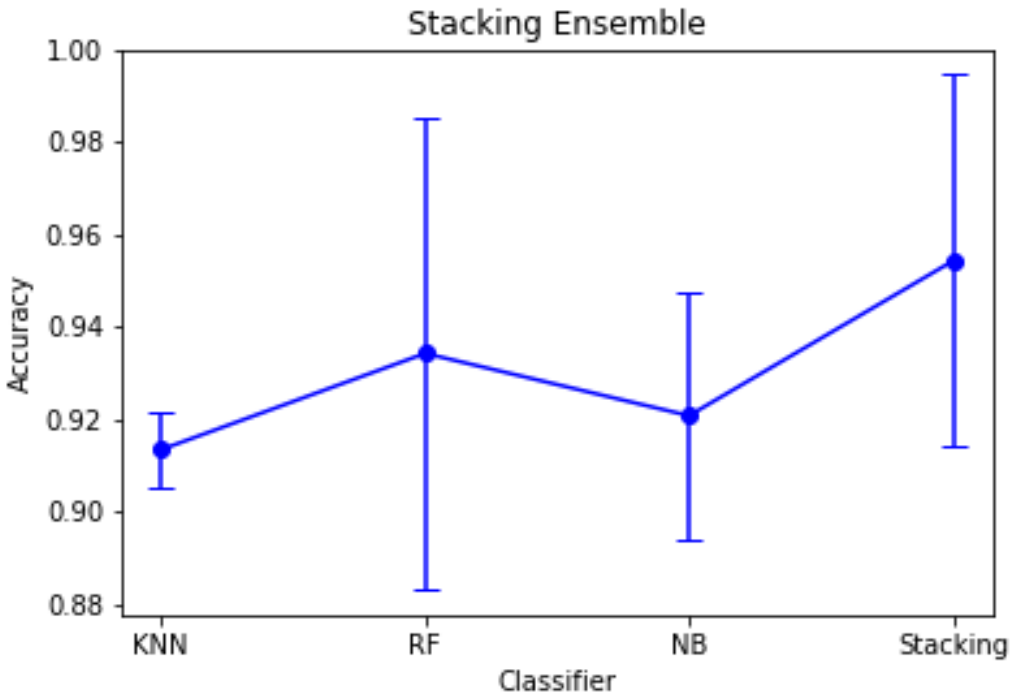
- The decision boundaries blend by the effect of the stacking classifier
- The stacking achieves higher accuracy than individual classifiers

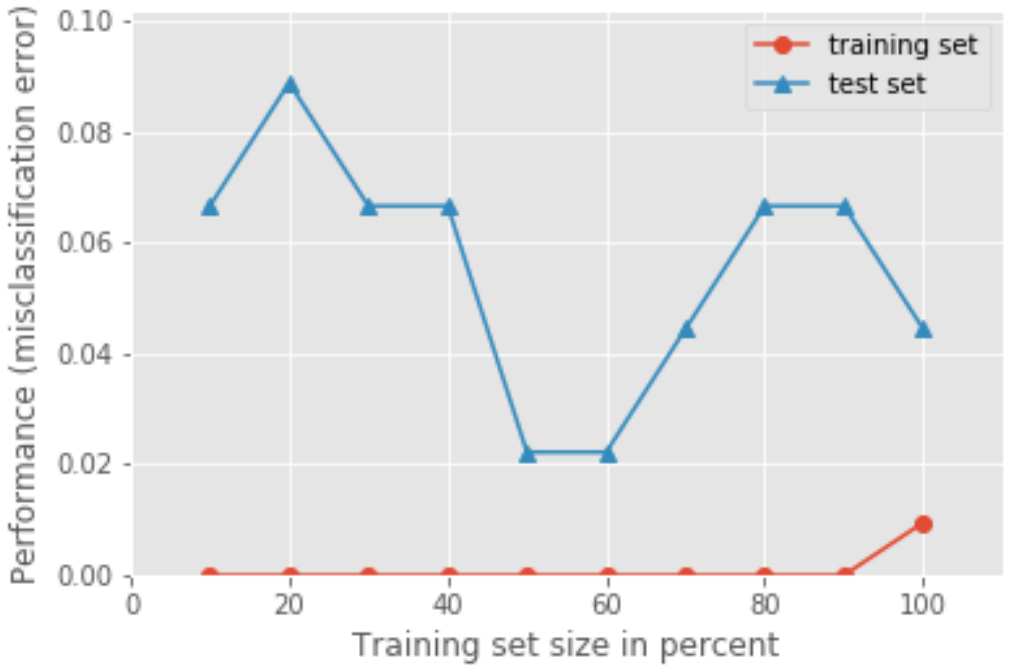
| | Accuracy | |
|---|---|---|
| | Mean | Standard Deviation |
| N-NN | 0.91 | ±0.01 |
| Random Forest | 0.93 | ±0.05 |
| Naïve Bayes | 0.92 | ±0.03 |
| Stacking | 0.95 | ±0.04 |



# Stacking Example

- The learning curves show no signs of overfitting



# Demo 8.4: Stacking

- Purpose
  - Understand the concept and potential of Stacking
- Resources
  - Sample data from SciKit-Learn
- Materials
  - Jupyter Notebook (Demo-8_4)

# Lab 8.3: Stacking

- Purpose
  - Work with Stacking
  - Practice some coding in Python
- Resources
  - Sample data from SciKit-Learn
- Materials
  - Jupyter Notebook (Lab-8_3)

# Questions?

# Appendices

# End of Presentation!