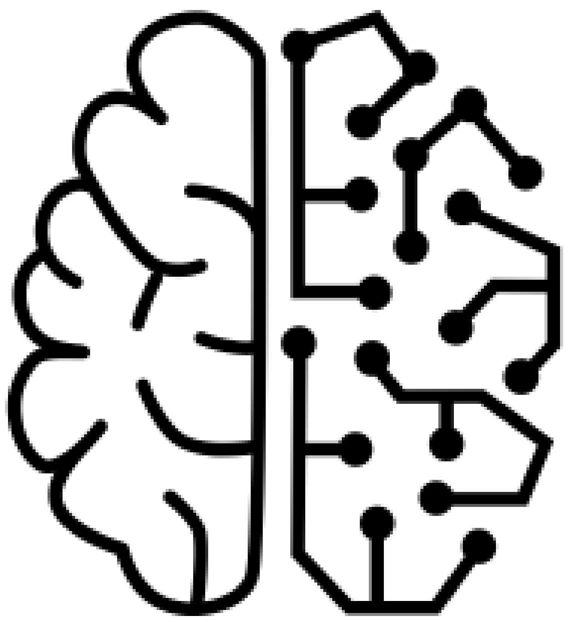
# Specialist Certificate in Data Analytics Essentials

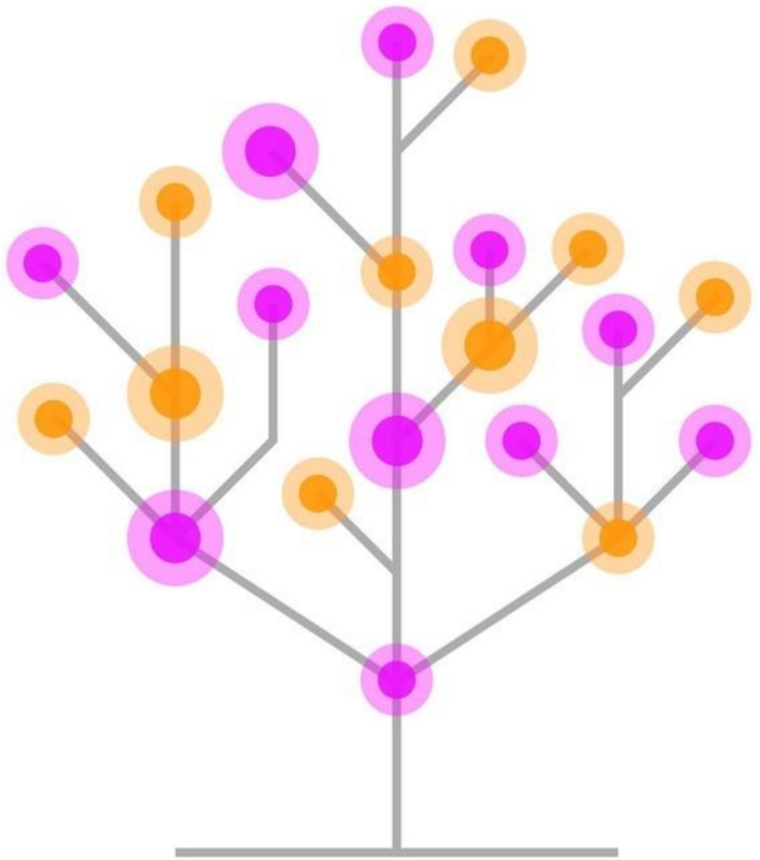
# Overview

- Model Architecture
    - Model Complexity
    - Hyperparameters
- Overfitting vs Underfitting
- Hyperparameter Tuning
- Bias-variance Tradeoff
- Ensemble Learning
    - Parallel Learning
        - Bagging
    - Sequential Learning
        - Boosting
    - Stacking



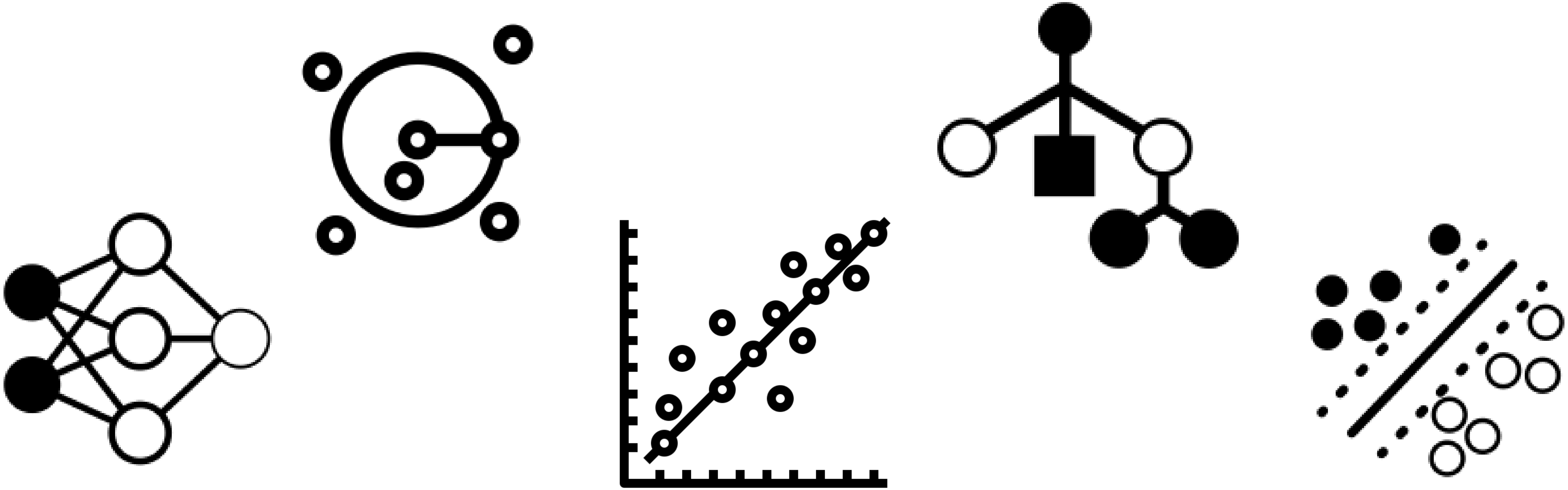
# Model Architecture

- So far we have looked at how to train a model. This means using machine learning to arrive at the **parameter values** that maximise accuracy (or minimise error).
- When talking about the model architecture, there are two main considerations:
  - The **type of algorithm** used by the model
  - And the **complexity** of the model



# Model Algorithm

- The type of algorithm you choose will depend on the **type of problem** you are solving and the **type of data** you are working with.
- Examples include: Linear regression, Logistics Regression, Decision Trees, K-nearest neighbours, Support Vector Machines, Neural Networks

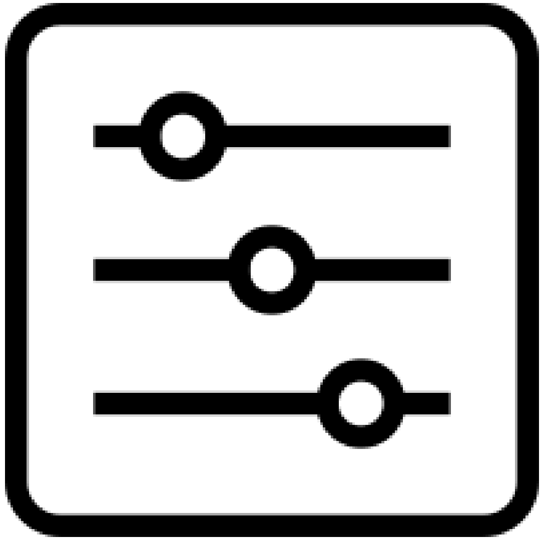


# Model Complexity

- The complexity of a model depends on **how many parameters** are available for the model to control.
- Fortunately, there are parameters that **control the number of parameters** in a model.
- These are called **hyperparameters.**
- The goal is to have a robust, accurate, and not-overfit the model

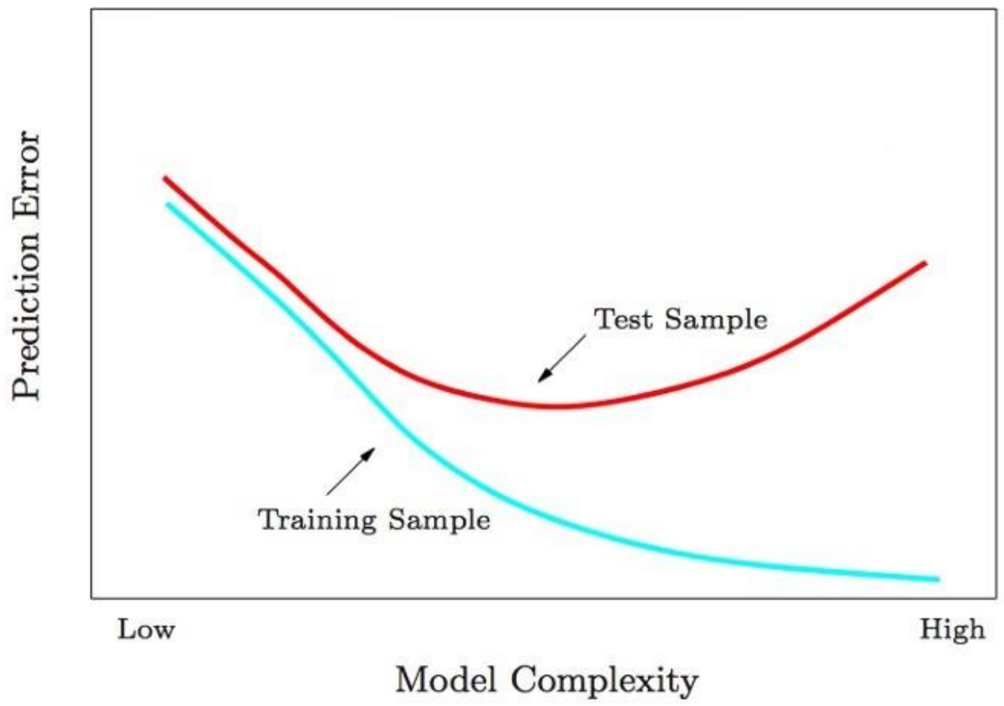
# Hyperparameters

- Decision tree:
  - Hyperparameters choose the **number of splits, max_depth** (tree depth), **etc**
- Regression
  - Hyperparameters choose the **degree or order** of the line
- Random forest
  - Hyperparameters include n_**estimators** (number of trees), **max_features** (number of features), **max_depth** (tree depth)
- K-Nearest Neighbours
  - Hyperparameters control the **number of neighbours**



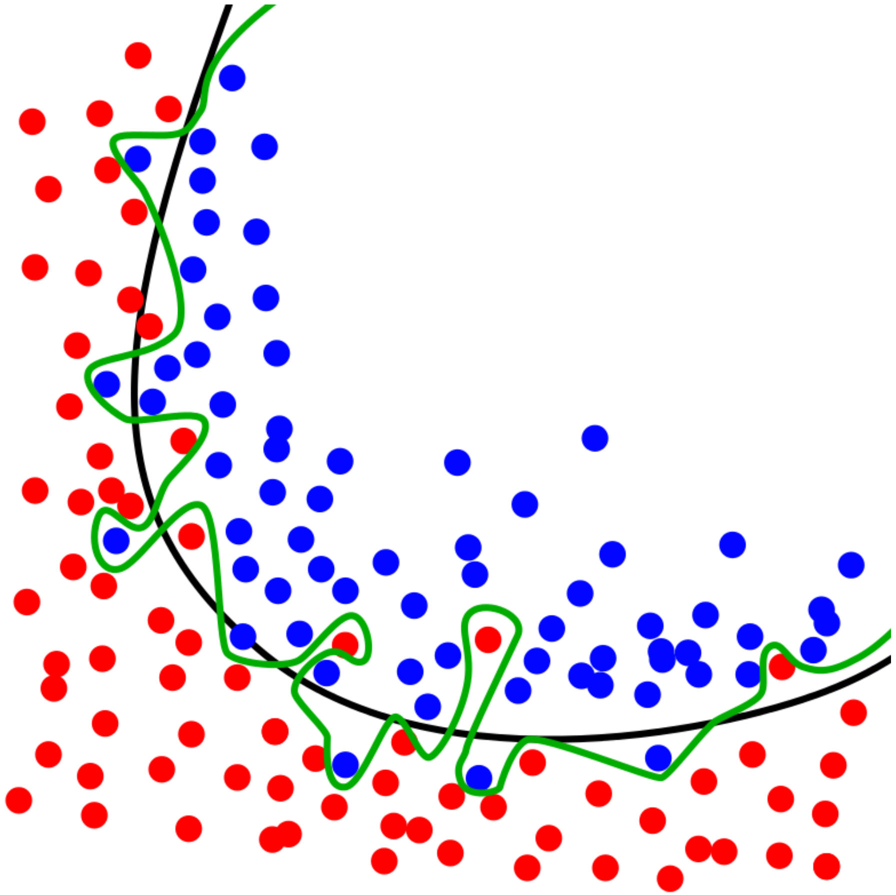
# HIGH-COMPLEXITY MODELS

- The more parameters, the more complex the model, **the better chance the model has to fit the training data**.
- When training our models, there is a temptation to increase the complexity of the model so that it yields the **lowest error possible**.
- However, something strange happens when we do this: after a certain level of complexity our model starts **performing worse on the test set.**
- This is because in order to get as high an accuracy as possible on our training set, our model has succumbed to learning **spurious correlations.**



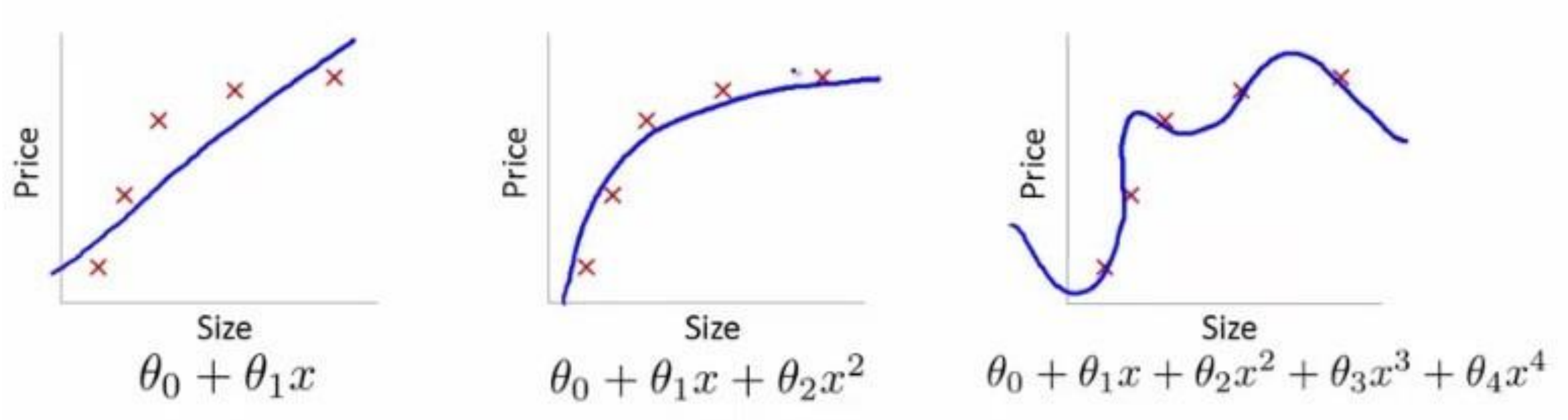
# Overfitting vs. Underfitting

- This is a downside risk associated with increased model complexity. It is known as **overfitting** the training data.
- When errors are caused by an oversimplified model it is known as **underfitting**.
- We can control the complexity of the model, and thus find the balance between these two errors, by **tuning the hyperparameters.**



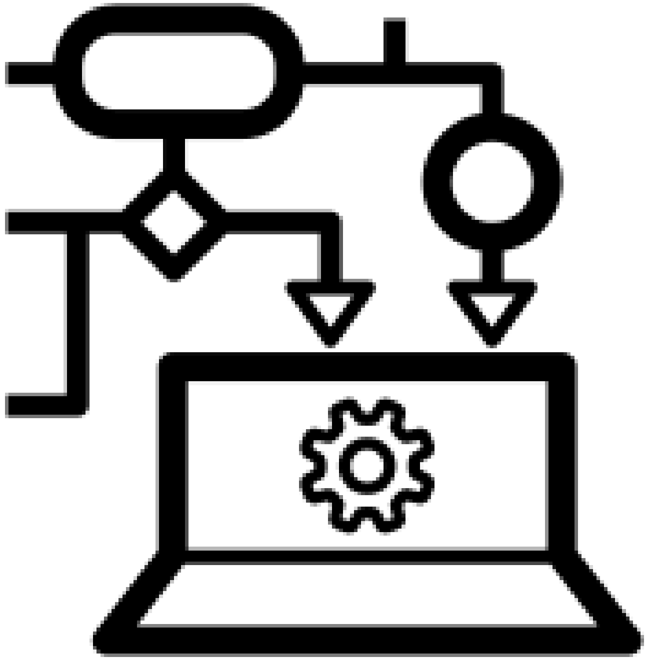
# HYPERPARAMETER TUNING

- Finding the **balance** between the errors caused by an oversimplified and an overcomplicated model
- You can find this balance by **hyperparameter tuning**
- Mathematically, this is known as the **bias-variance tradeoff**



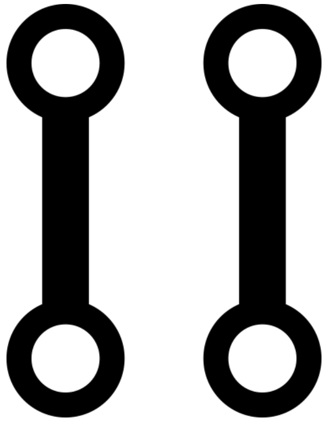
# Ensemble Learning

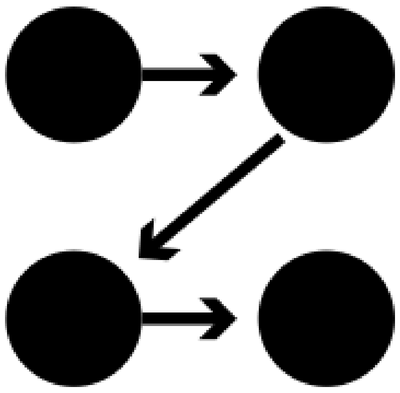
- So far, we have chosen the model with the highest accuracy and then discarded the rest.
- However, instead of discarding the all the other models, we could **combine multiple models together** to get a higher accuracy than any individual model
- This is known as **ensemble learning.**



# Ensemble Learning Techniques

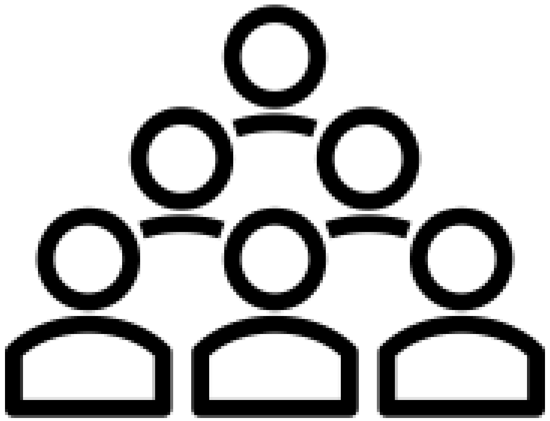
- Broadly speaking, there are two methods to combine models: **parallel** or **sequential** ensemble techniques.
- This will determine whether we train our models through **collective learning** or **gradual learning**.
- By developing the models independently and averaging we tend to **reduce the variance**
- Whereas an adaptive, sequential, iterative approach would focus on **reducing model bias**.





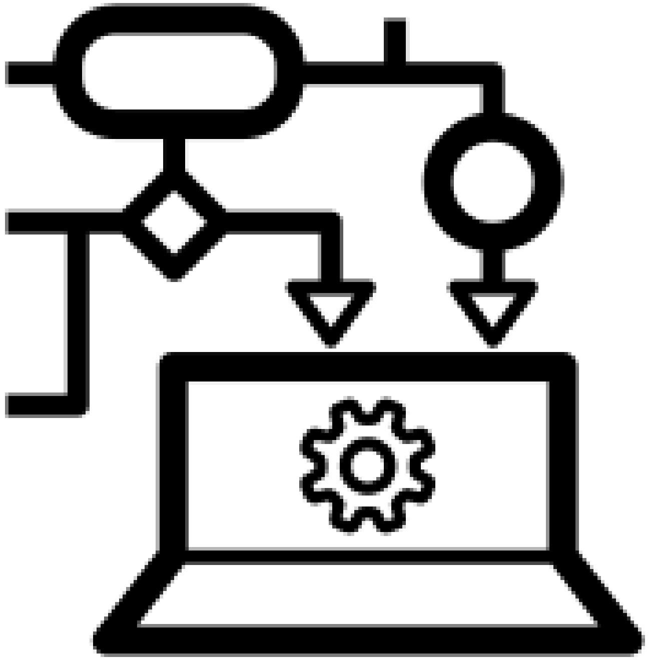
# Parallel Learning

- Training the models in parallel is a form of **collective learning**.
- The principle of collective learning is the **wisdom of the crowd**
- Here, models are trained **independently**
- By developing the models independently and averaging we tend to **reduce the variance**
- These ensembles can be either composed of models of the **same** or **different** algorithms.
- An ensemble of models using the same algorithm is known as a **homogeneous ensemble**
- An ensemble from different algorithms is a **heterogeneous ensemble**.



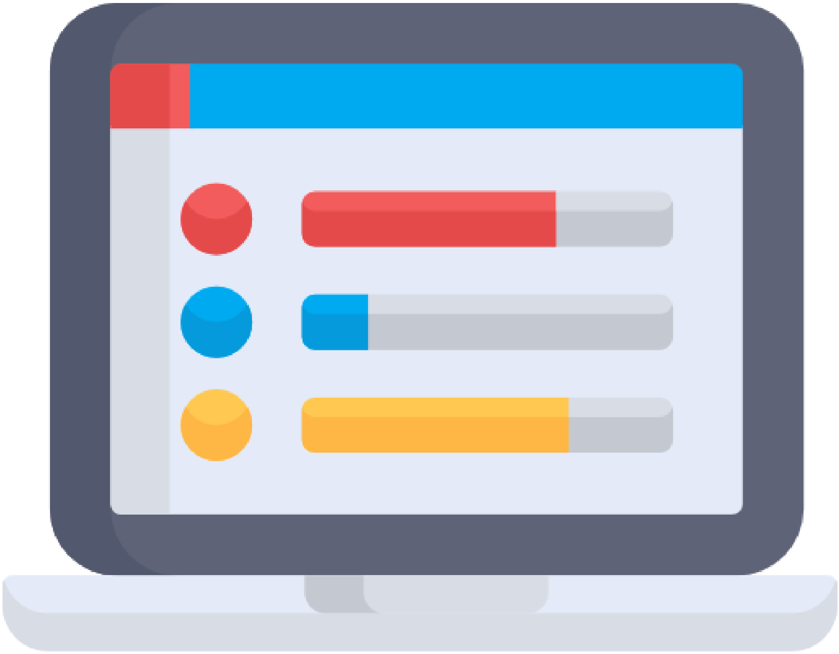
# Heterogeneous Ensembles

- In heterogeneous ensembles, we aim to have a collection of as **diverse models** as we can.
- These ensembles are most effective when the models are as **different as possible** (different algorithms or trained on different datasets).
- They should also be **independent** and **uncorrelated.**
- Similar to how combining multiple diverse assets into a portfolio reduces its **variance**, we can do the same with machine learning models.
- By reducing the error due to variance, this makes the model **more robust.** This makes it **more generalizable** and **less susceptible to small changes** in the data.



# HETEROGENOUS ENSEMBLES — MAJORITY VOTING

- Each model **votes** on the answer. This works for classification problems.
- This ***wisdom-of-the-crowd*** technique works by canvassing diverse opinions.
- This is also known as **hard voting.**

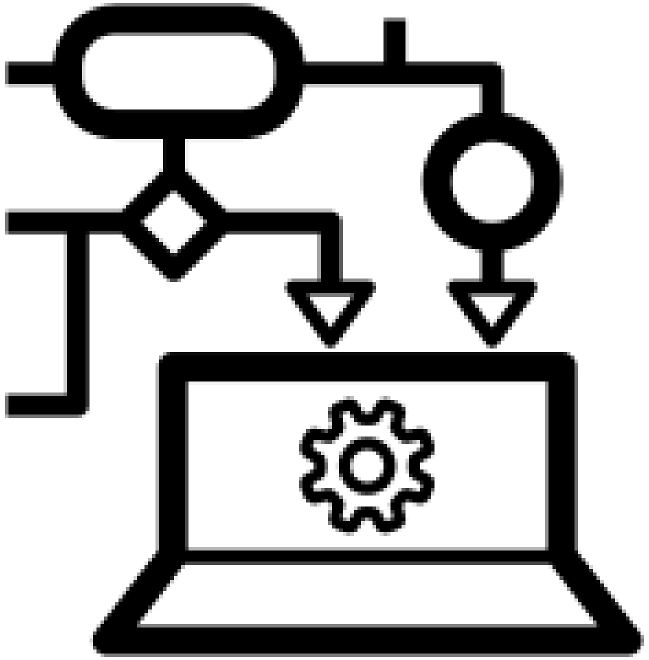


# Heterogenous Ensembles — Averaging

- Another approach would be to **averages** the guesses of each model. This technique also works on regression
- For regression, it gets the average of the **predicted values** from each model
- For classification, it gets the average of the **predicted probabilities**.
- This is also called **soft voting**

# Homogeneous Ensembles — Weak Learners

- Homogeneous ensembles aggregate models of the **same algorithm type**.
- When using the same algorithm, we achieve model diversity by training each model on **different parts of the dataset**.
- Where heterogenous ensembles use accurate and finely-tuned disparate models to reduce variance, homogenous ensembles combines many **low-performing models** to increase accuracy.
- These low-performing models are known as **weak learners.**
- Weak learners are any models that perform **slightly better than chance/random**.

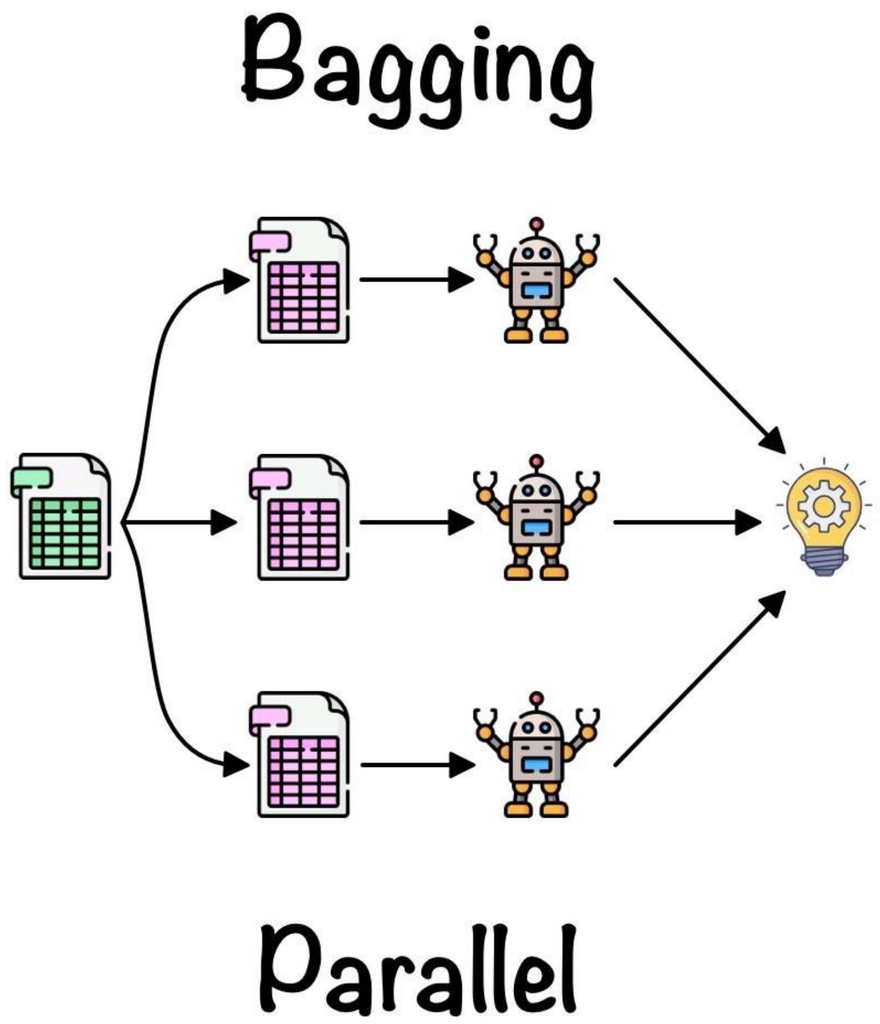


# BOOTSTRAP METHOD

- In order to train multiple models on the same dataset, we **randomly sample the dataset** multiple times.
- This is done by randomly picking out samples and then returning them to the dataset
- The same samples can be picked out more than once; this is called **sampling with replacement**

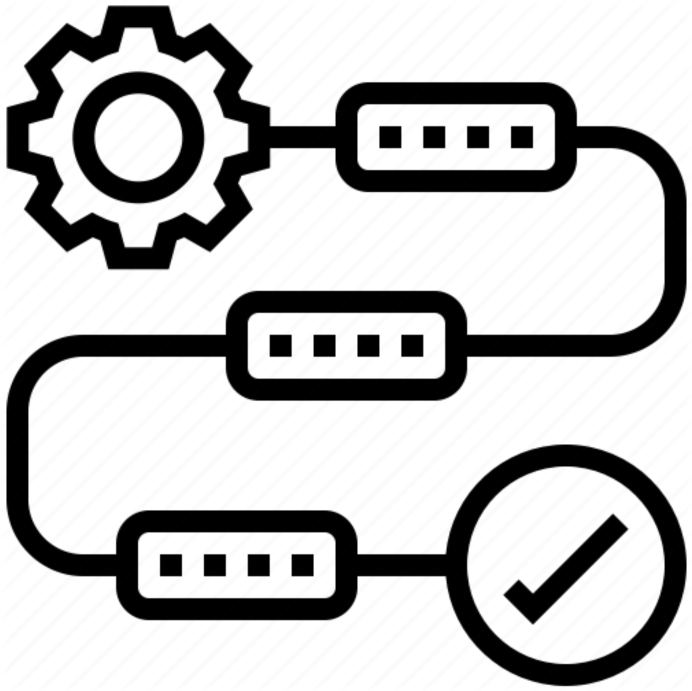
# Bootstrap Aggregating

- One **parallel** approach is to each model independently on bootstrapped data.
- Due to the variance in the training datasets, we obtain **different models each time**
- The outputs of these models are then averaged using the hard- or soft-voting methods discussed earlier.
- By aggregating multiple models together through bootstrapping, we create a model **more robust** than any of the individual models.
- "*Bootstrap aggregating*" is more commonly referred to as **bagging.**



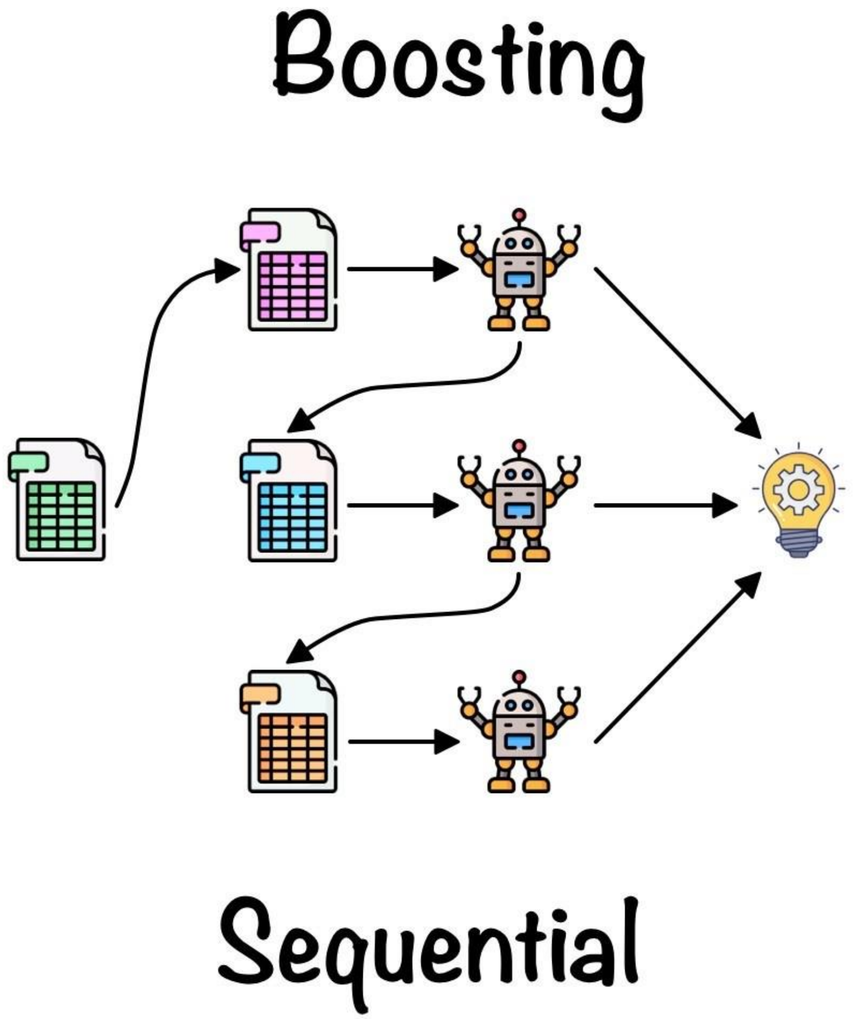
# SEQUENTIAL LEARNING

- The alternative method is to train the models **sequentially**.
- The principle here is one of **gradual** and **iterative learning**.
- In this methodology, the training of each model **depends on the previous one**.
- It is based on **receiving feedback** and **correcting errors** of previous models



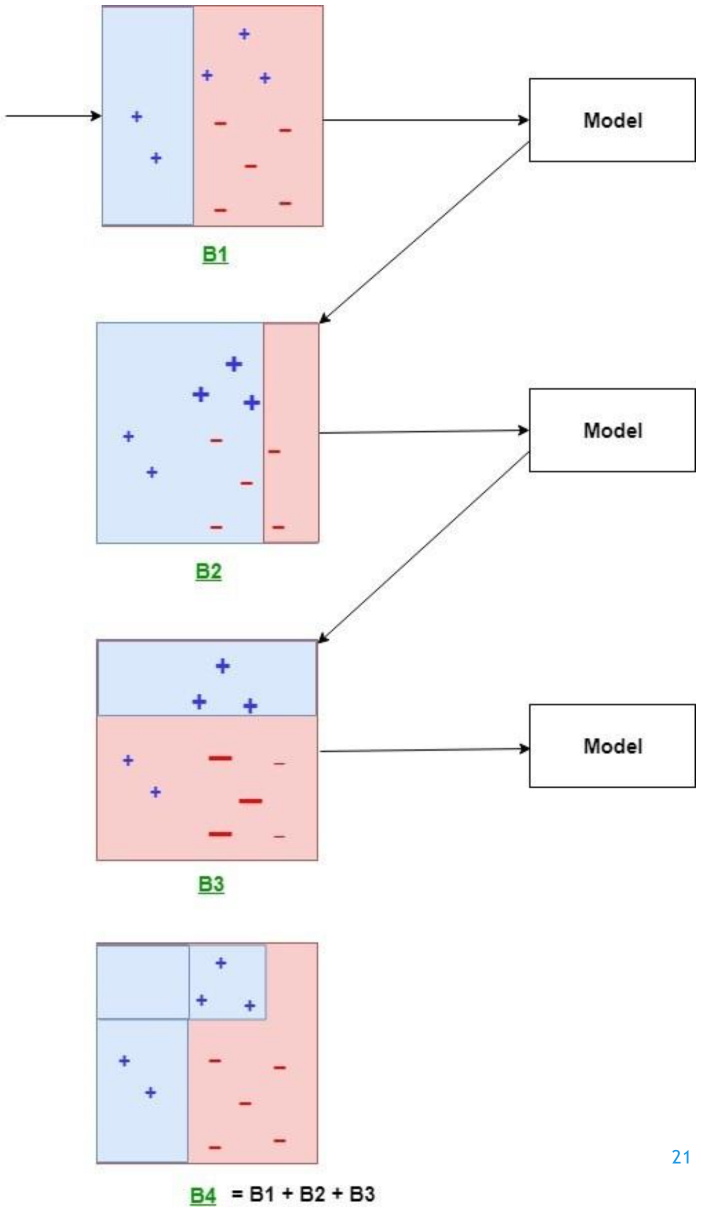
# BOOSTING

- One gradual learning approach consists of iteratively training weak models and adding them to a final strong model.
- This is known as **boosting.**
- This approach mainly focuses on **reducing bias**.



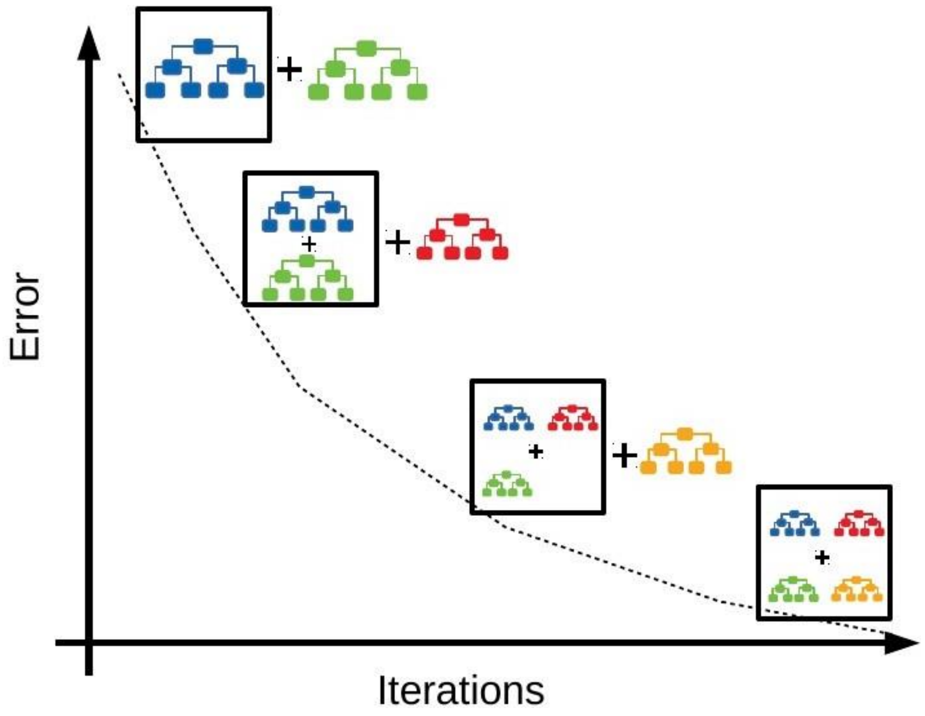
# ADAPTIVE BOOSTING

- One type of boosting weighs the data such that the next model will focus its efforts on the **most difficult (wrong ones) observations to fit**.
- This focuses subsequent models on areas the previous models were weak on.
- This is known as **adaptive boosting** (AdaBoost).
- Each model is designed to focus on where the performance of the previous model was poor.
- The weak learners are added and **weighted** in accordance with their performance



# GRADIENT BOOSTING

- Instead of weighting the samples to focus on certain data points, another approach is to train the model purely on the **remaining errors**.
- This creates a new learner each time, which are **added** together to create the ensemble
- Each successive learner reduces the error, hence the **gradient** reduction.



# STACKING

- First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs.
- The architecture consists of **Level-0 Models** (base models) and a **Level-1 Model** (meta-model)
- We are basically using **machine learning** to figure out which models to trust.
- Stacking typically yields performance better than any single one of the trained models
- The two top-performers in the Netflix competition used *blending*, which may be considered to be a form of stacking