Topic 4
Python Supervised Machine Learning

ST0249 (AIML) AI & MACHINE LEARNING

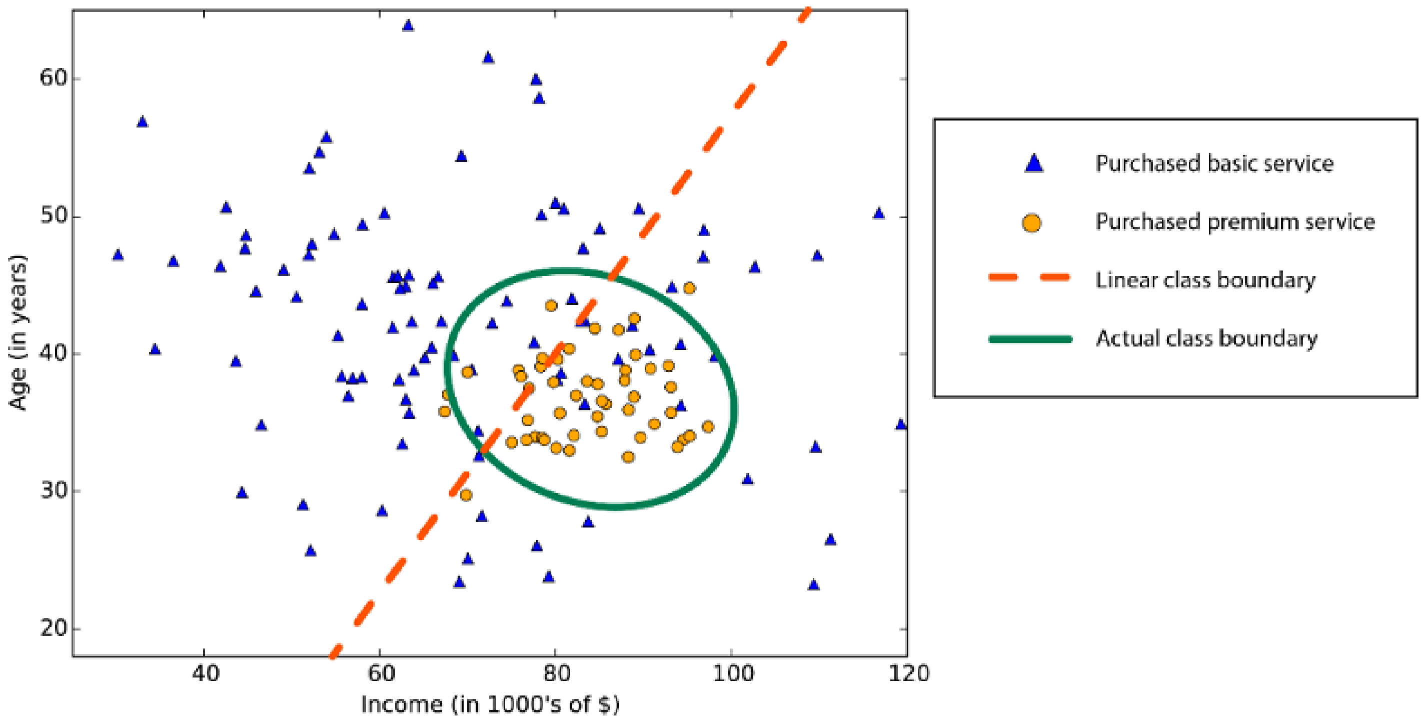
Learning Outcomes

- ❑ Understand Classification and Regression
  - Explain use cases for classification and regression
  - Explain generalization, overfitting and underfitting
- ❑ Supervised Machine Learning Algorithms
  - Understand k-Nearest Neighbours
  - Understand Linear Models
  - Understand Naïve Bayes Classifiers
  - Understand Decision Trees and Random Forest
  - Understand Support Vector Machines

Understand Classification and Regression

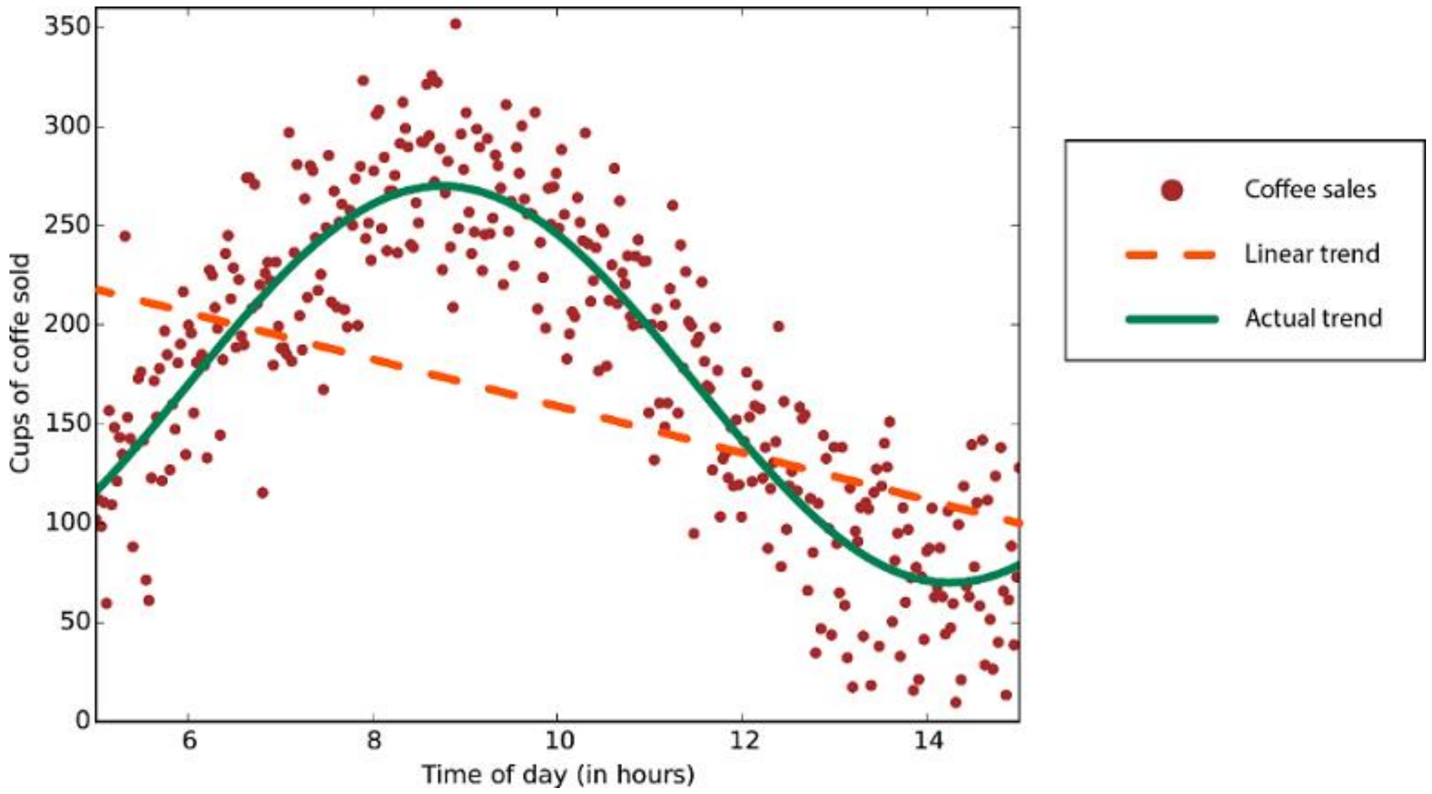
Classification

Predicting/Estimating the class/label/category of the output/target variable



Regression

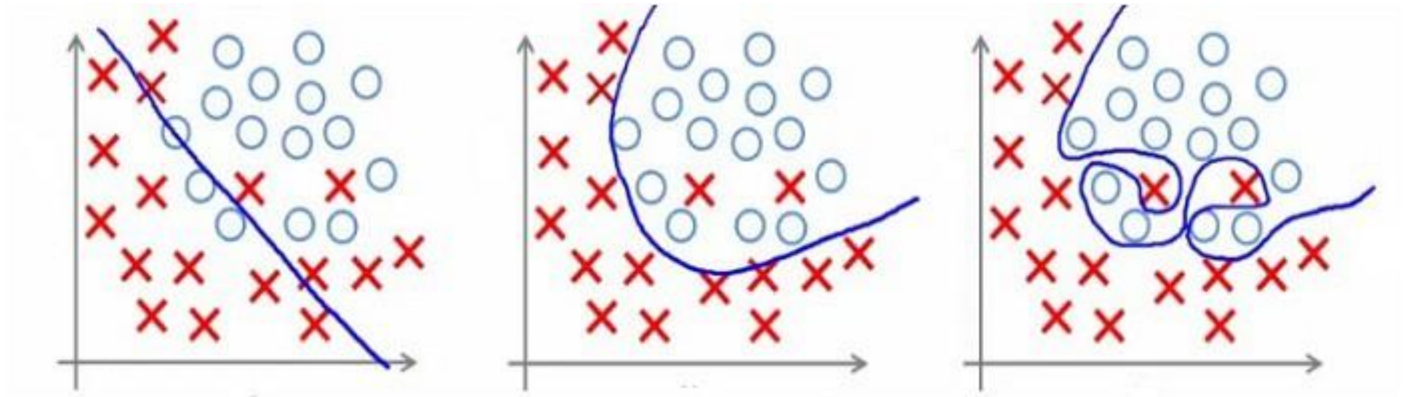
Predicting/Estimating the value/numeric quantity of the output/target variable



Goldilocks problem

HOW TO FIND THE RIGHT-FIT?

Right-fit for Classification: one that generalizes the decision boundary



Under-fitting

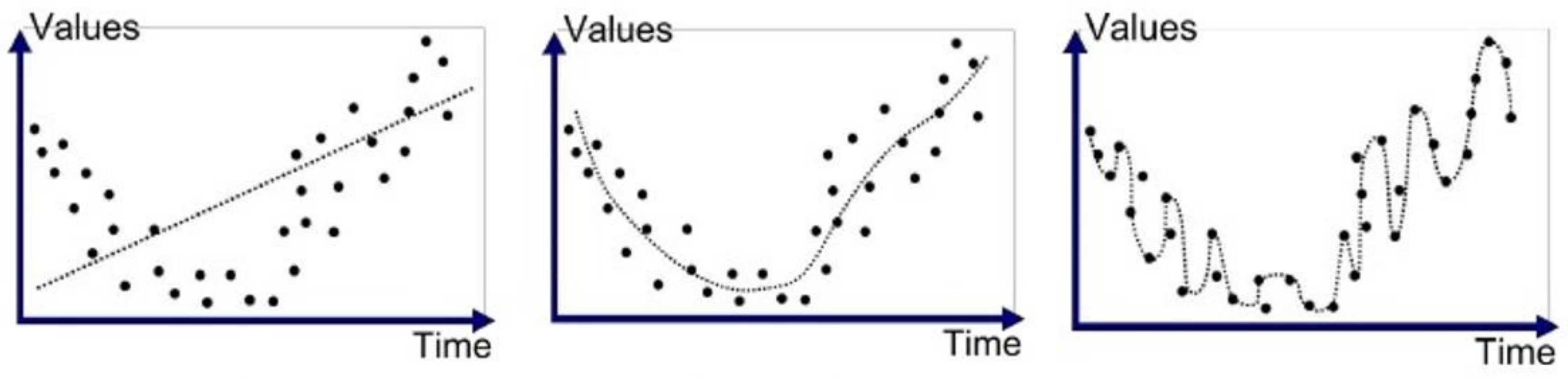
(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

Right-fit for Regression: one that generalizes the trend



Underfitted Good Fit/Robust Overfitted

Bias-Variance Trade-off

Bias-Variance Trade-Off

BIAS

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.

Model with high bias pays very little attention to the training data and oversimplifies the model.

It always leads to high error on training and test data.

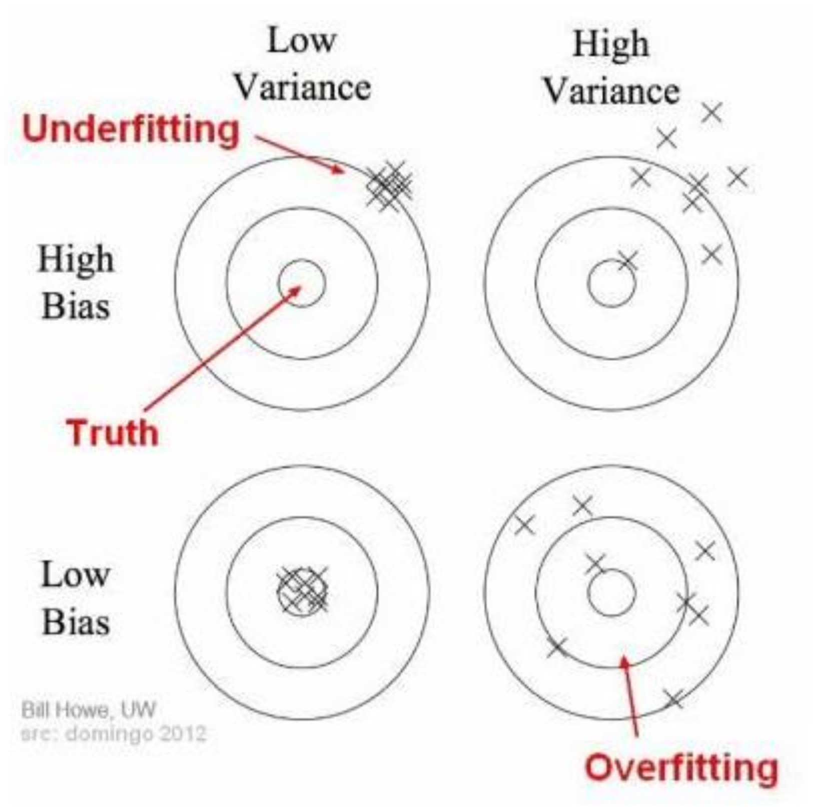
VARIANCE

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data.

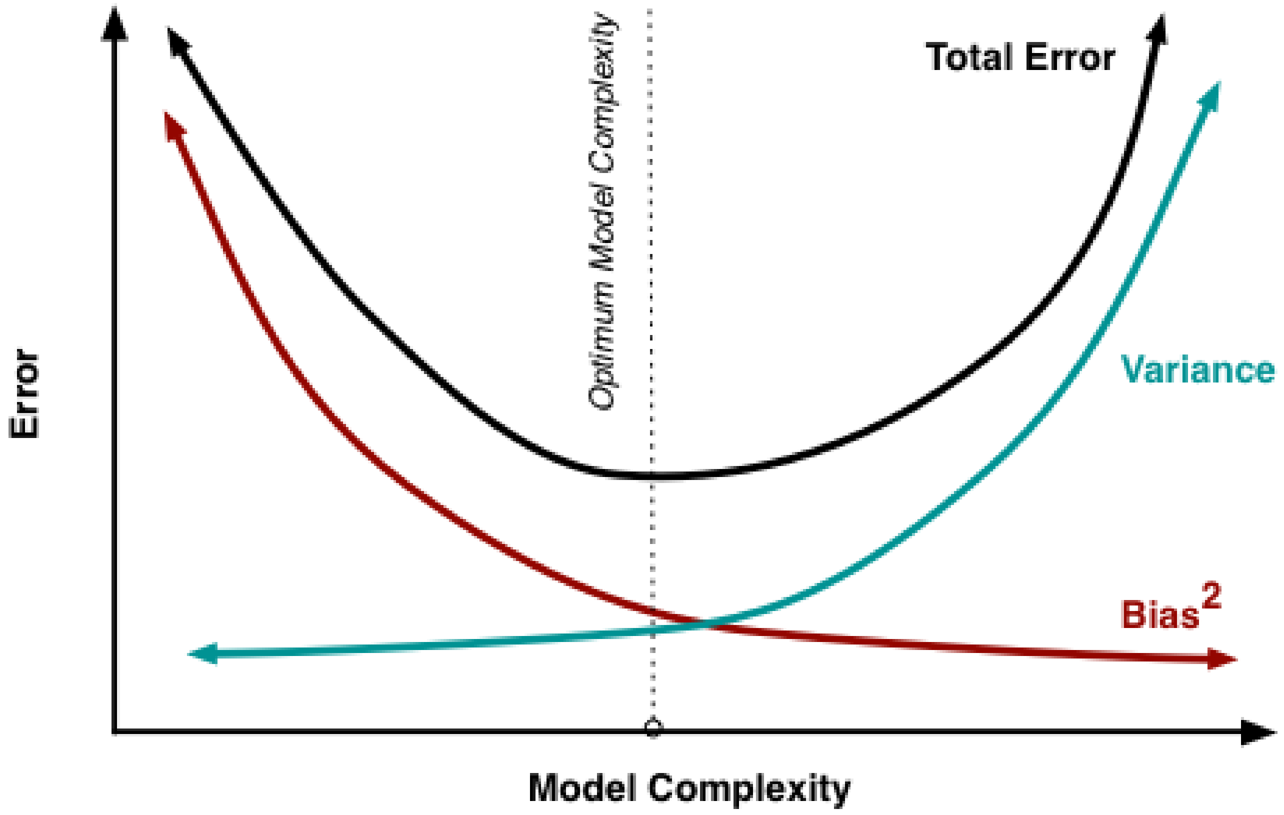
Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before.

As a result, such models perform very well on training data but has high error rates on test data.

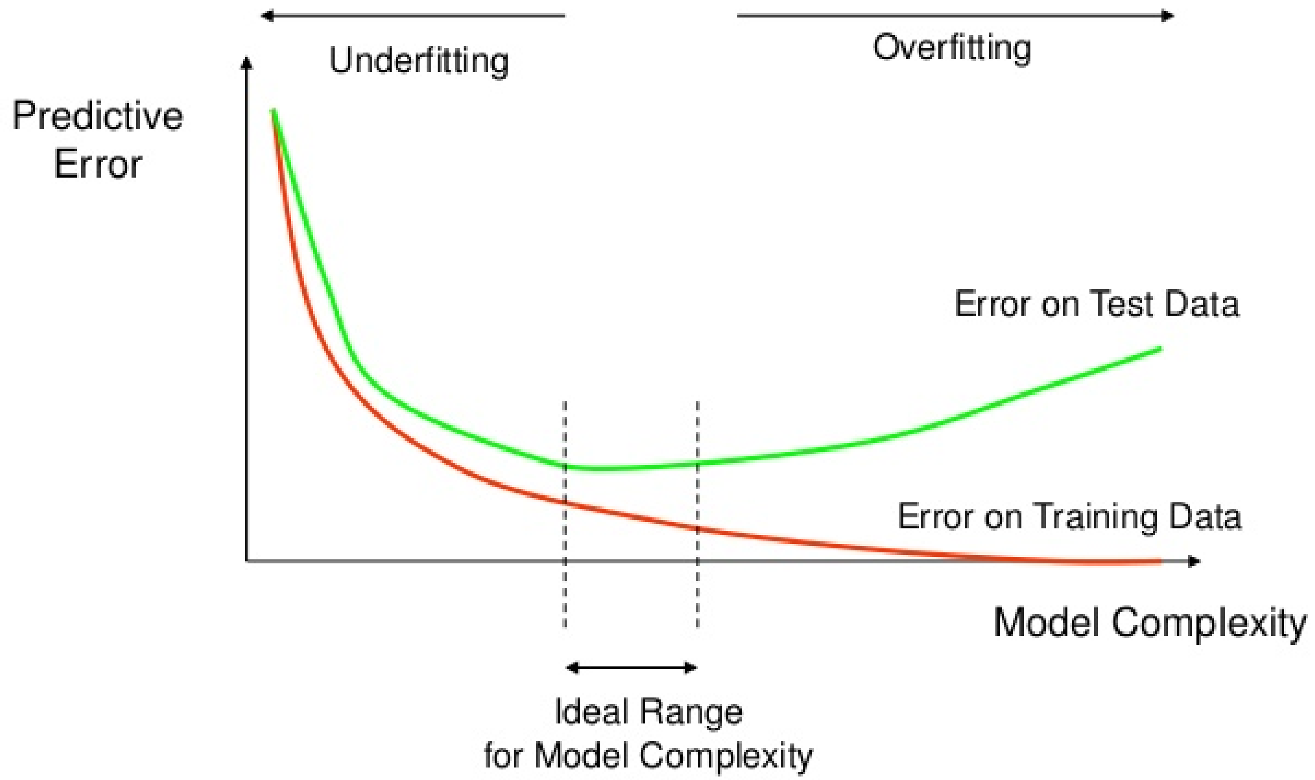
Bias-variance trade-off



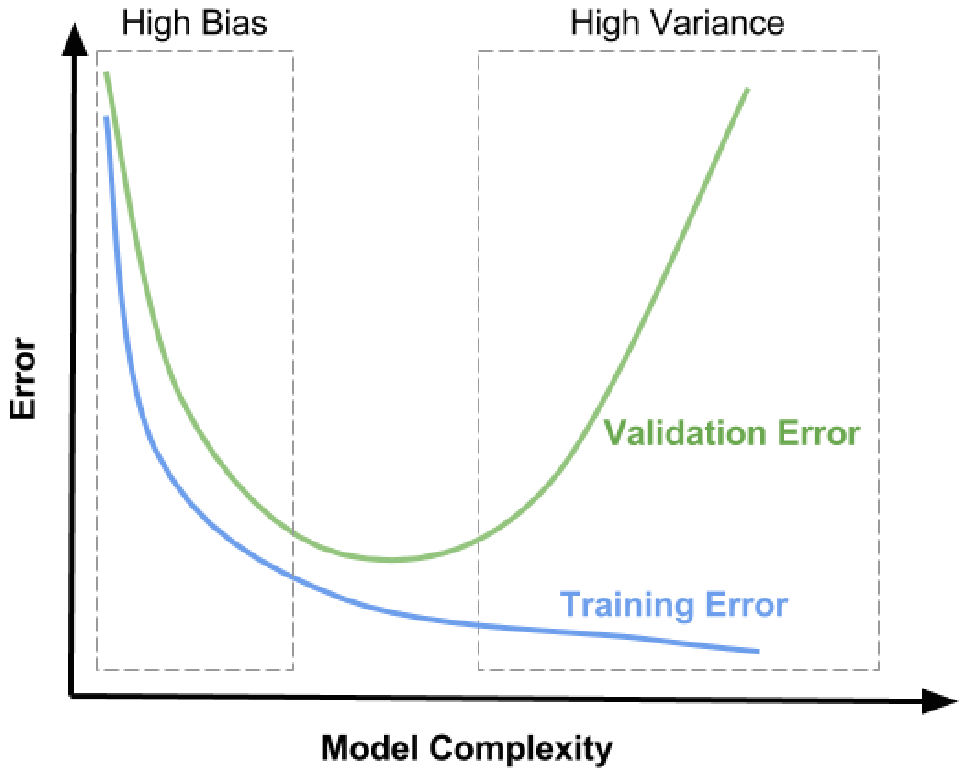
Bias-variance trade-off



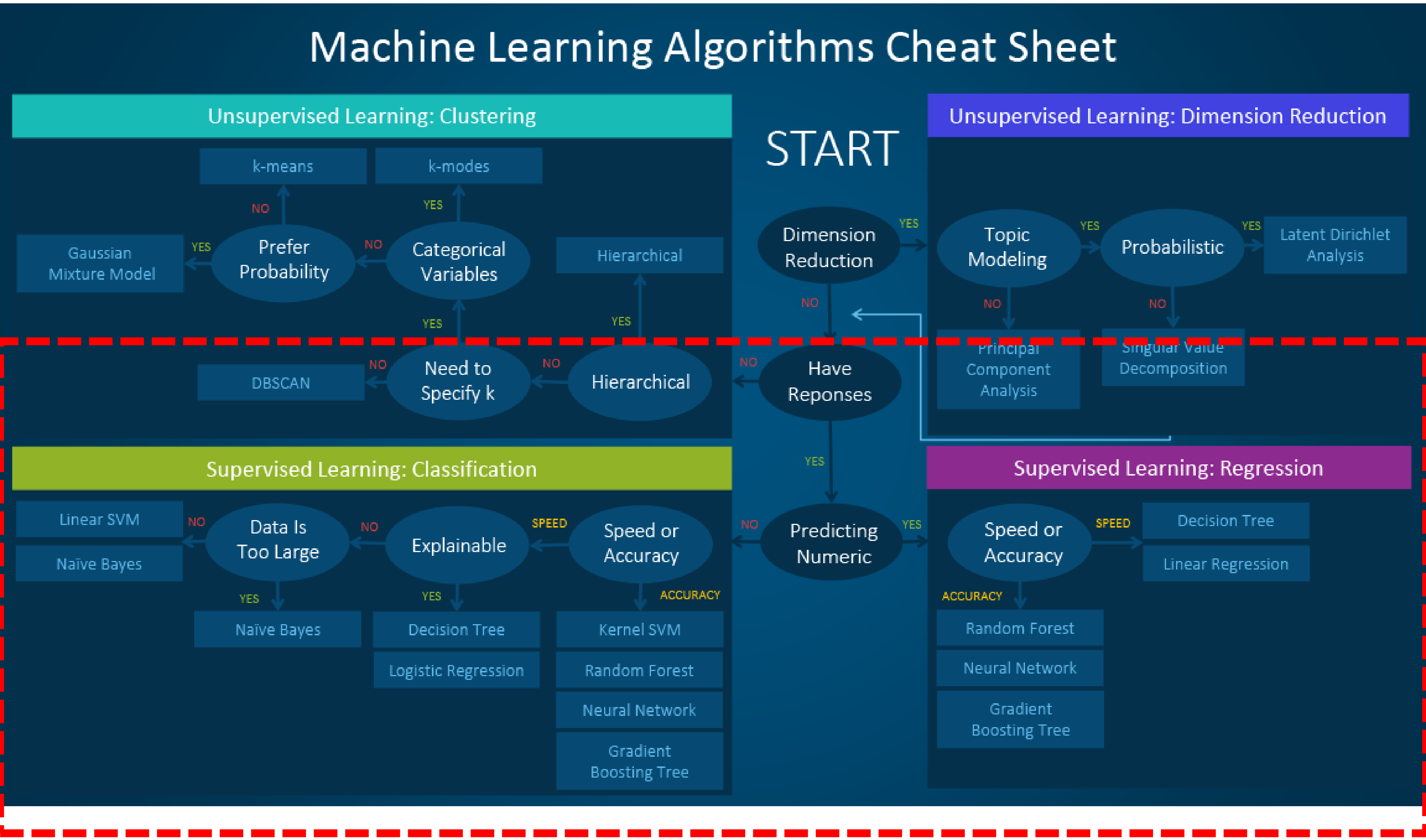
The model complexity must be correct to get the right fit



Bias-variance trade-off



Supervised Machine Learning Algorithms

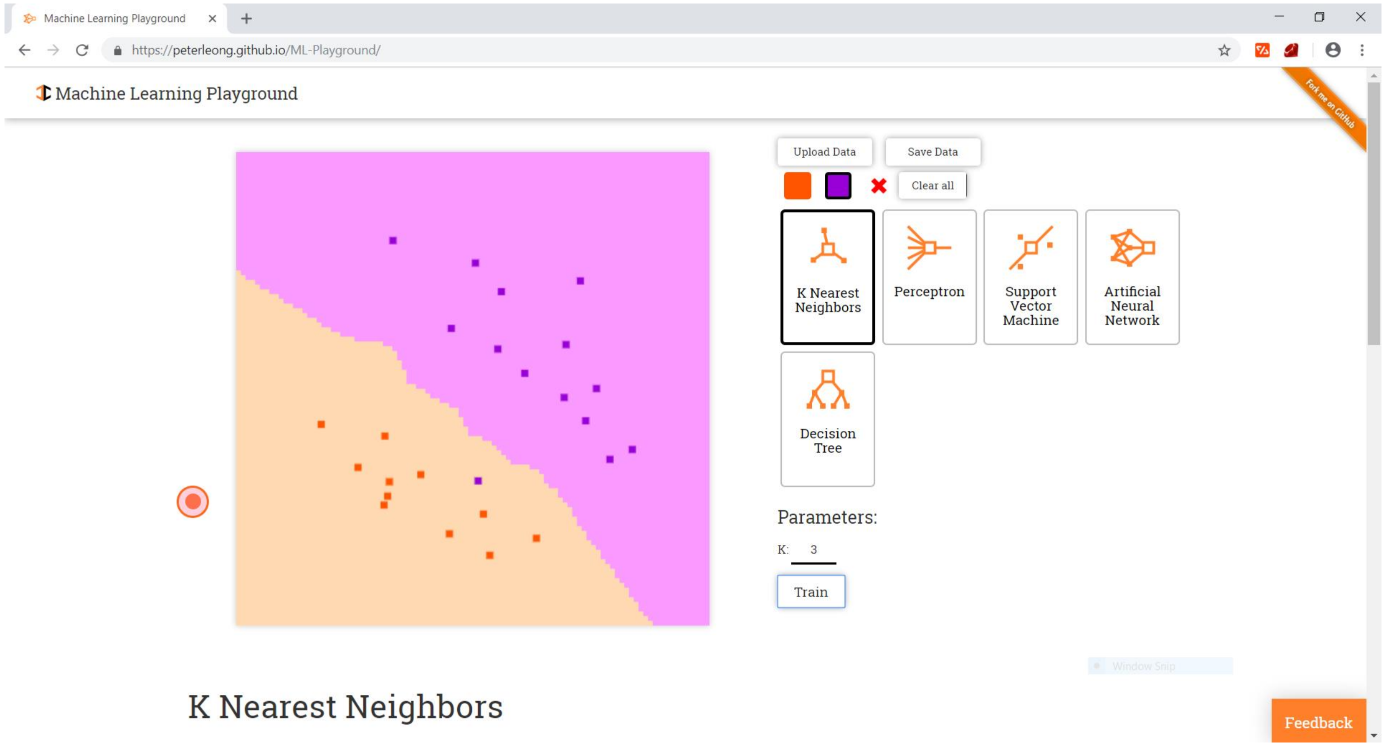


source https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

Activity

Try out the different classification algorithms in the ML Playground.

https://peterleong.github.io/ML-Playground/

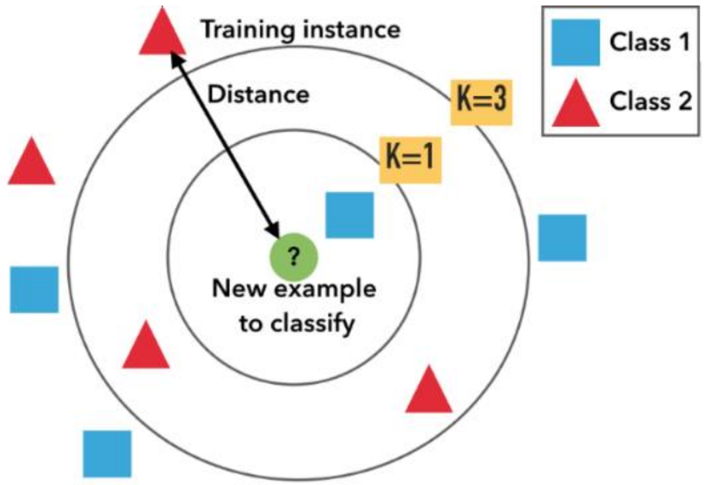


Classification

PREDICTION AN OUTPUT LABEL/CLASS

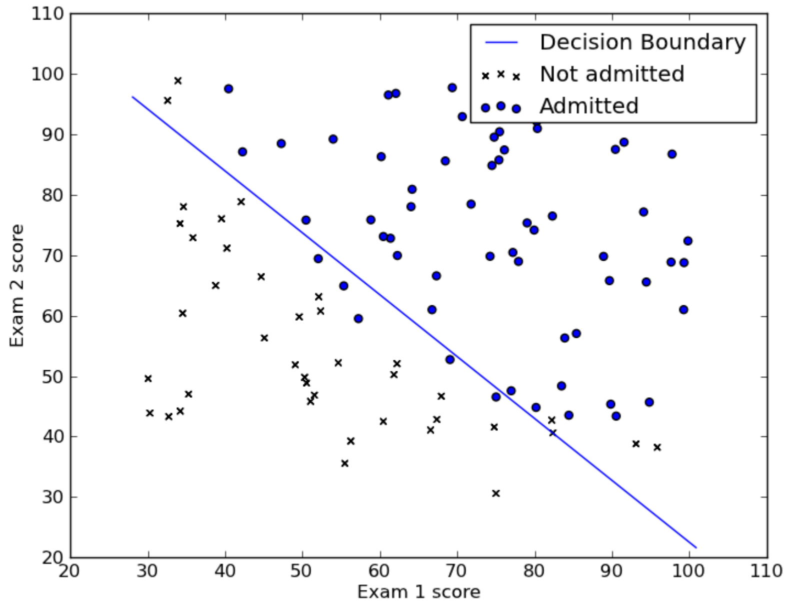
k-Nearest Neighbours

- K-Nearest Neighbour algorithms are interesting and simplest of all machine learning algorithms.
- First of all this algorithm memorize the training dataset with their corresponding label, then to forecast the label of any new query based on the labels of its closest neighbours in the training set.



Logistic Regression

- Logistic regression is used for classification. Logistic regression results in the linear decision boundary.



Naïve Bayes Classifier

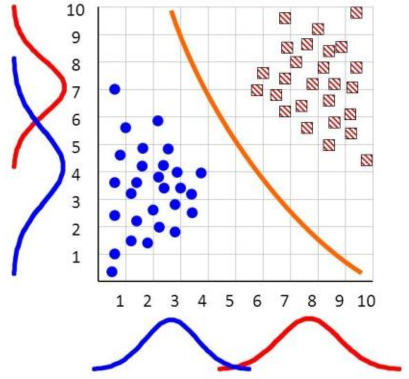
- All naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

$$\begin{aligned} p(C_k \mid x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\ &= p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,. \end{aligned}$$

Bayes classifier uses class label $\hat{y} = C_k$ that is has maximum probability:

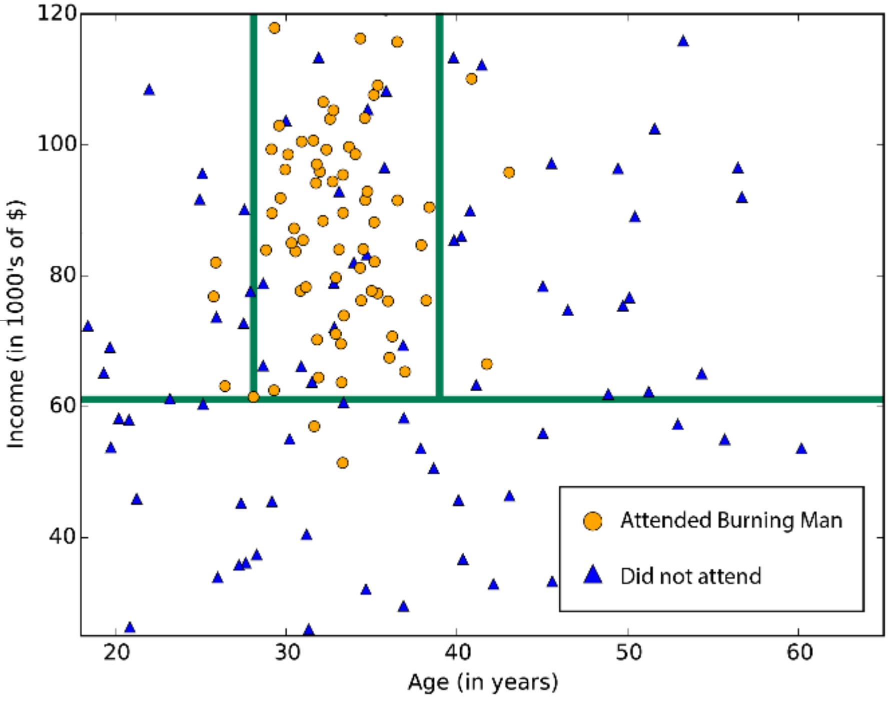
$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}}\; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

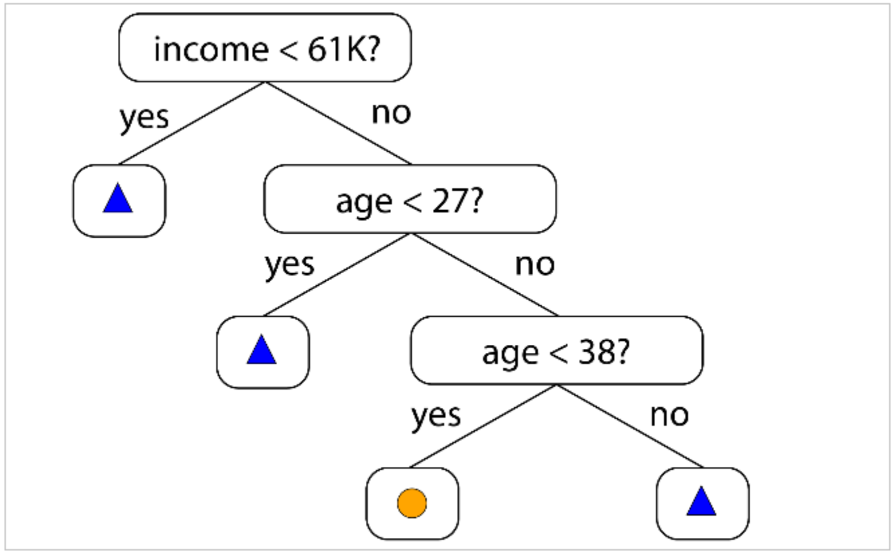
The Naïve Bayesian Classifier has a quadratic decision boundary



Decision Trees

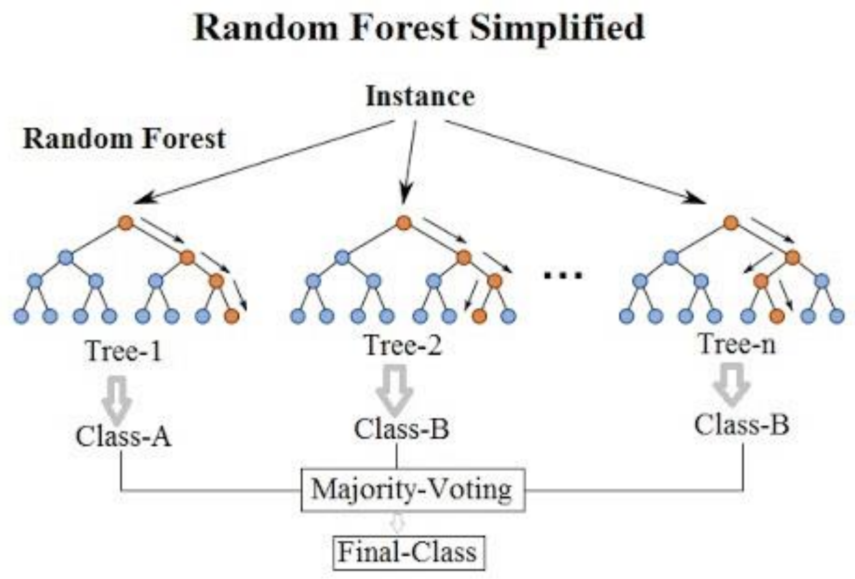
- There are many variants of decision trees, but they all do the same thing—subdivide the feature space into regions with mostly the same label.
- These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.



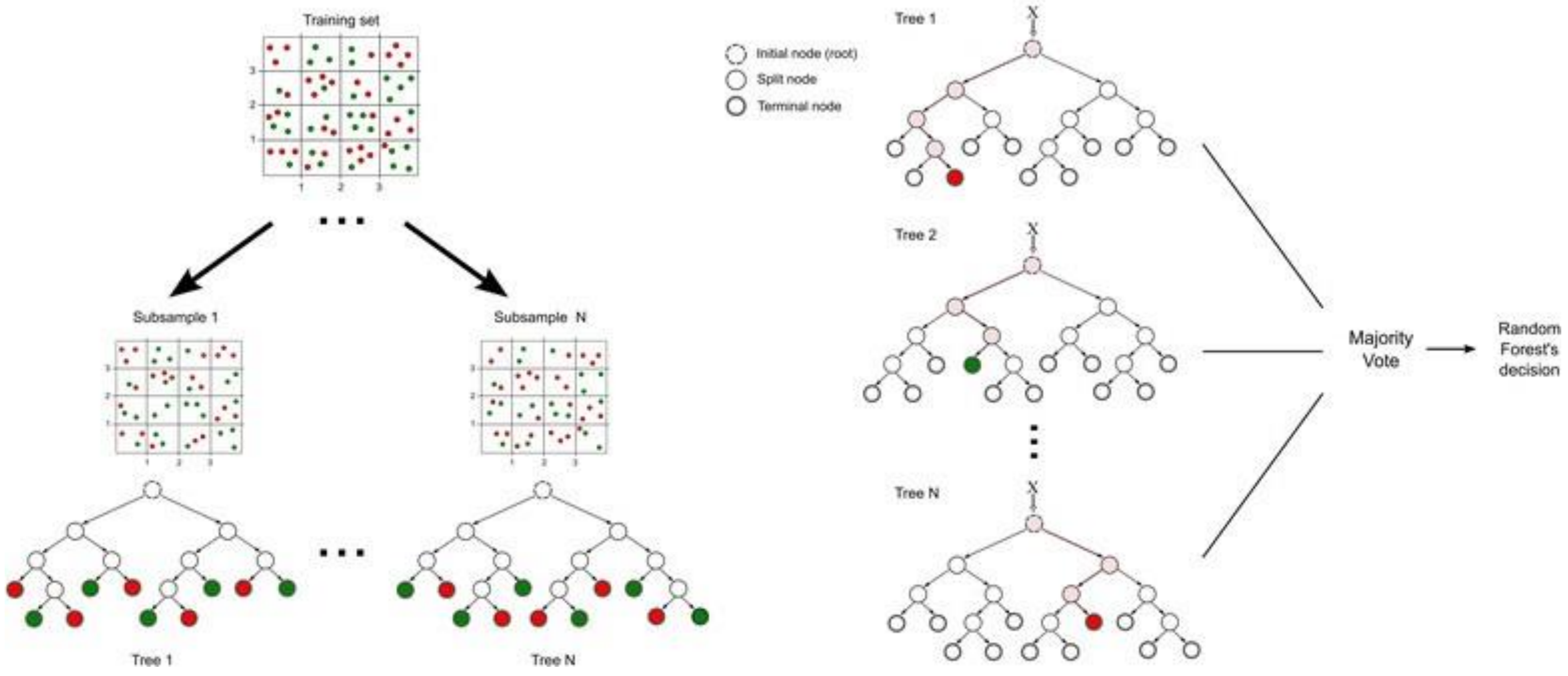


Random/Decision Forest

- Random/Decision forests (regression, two-class, and multiclass) are all based on decision trees.
- Because a feature space can be subdivided into arbitrarily small regions, it's easy to imagine dividing it finely enough to have one data point per region. This is an extreme example of overfitting.
- In order to avoid this, a **large set of trees** are constructed with special mathematical care taken that the trees are not correlated.
- The average of this "**decision forest**" is a tree that avoids overfitting. Decision forests can use a lot of memory.

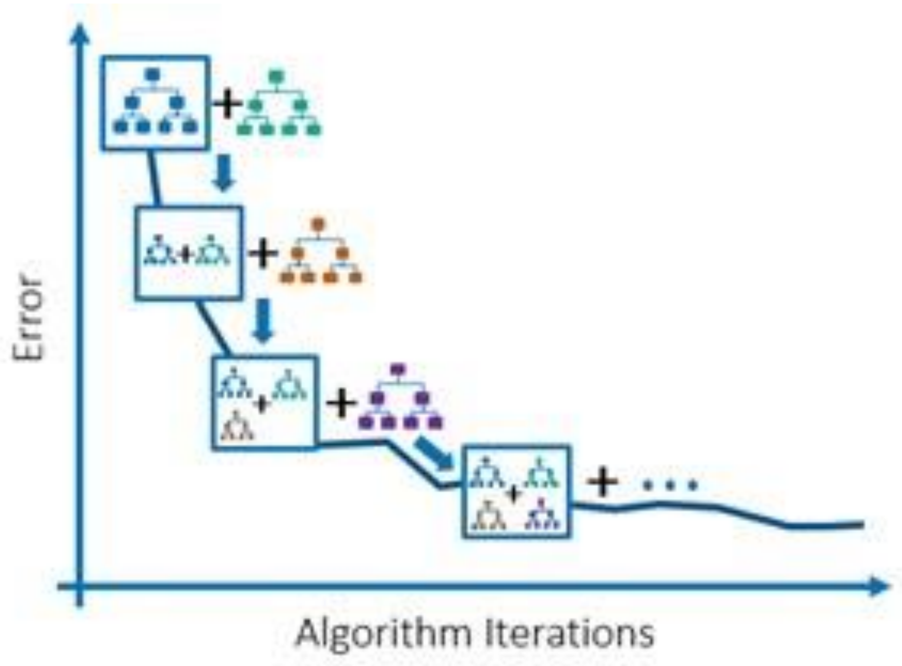


Random Forest (Visualization)



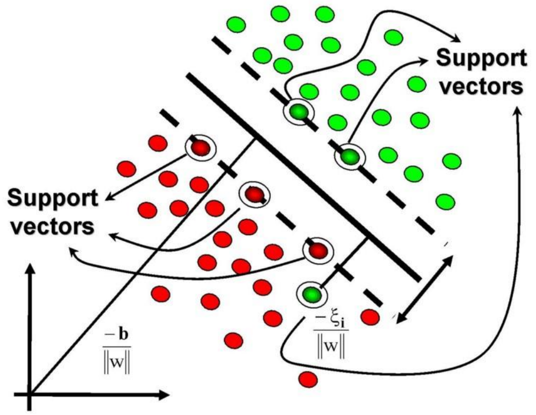
Gradient Boosting

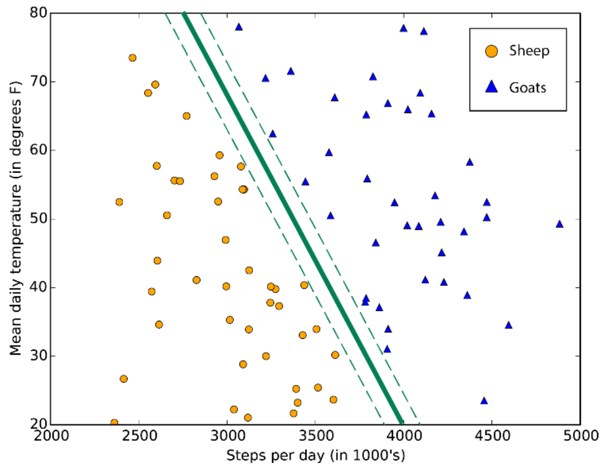
- A boosted decision tree is an ensemble learning method in which the second tree corrects for the errors of the first tree, the third tree corrects for the errors of the first and second trees, and so forth.
- Generally, when properly configured, boosted decision trees are the easiest methods with which to get top performance on a wide variety of machine learning tasks.
- However, they are also one of the more memory-intensive learners.



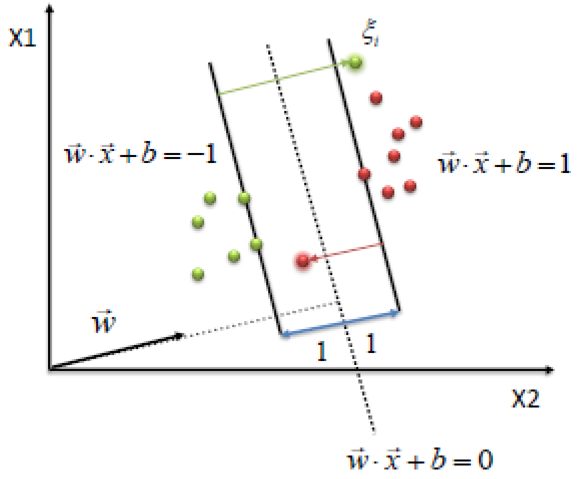
Support Vector Machines

- Support vector machines (SVMs) find the boundary that separates classes by as wide a margin as possible.
- When the two classes can't be clearly separated, the algorithms find the best boundary they can.
- **When most dependent variables are numeric**, logistic regression and SVM should be the first try for classification. These models are easy to implement, their parameters easy to tune, and the performances are also pretty good.





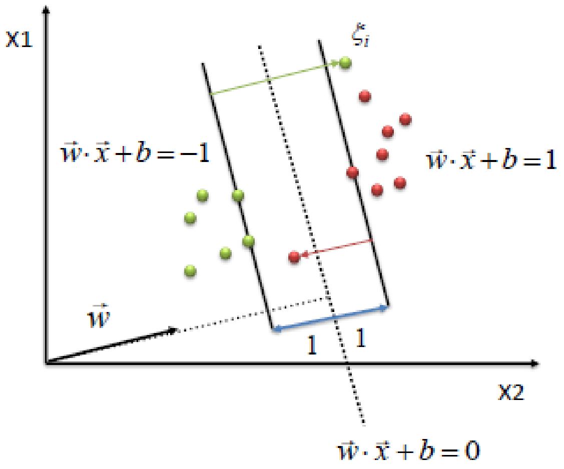
Support Vector Classifier


X1
X2
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}$
1
1

slack variable:

ξ_i

Allow some instances to fall off the margin, but penalize them


X1
X2
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}$
1
1

Constraint becomes :

$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$

$\xi_i \geq 0$

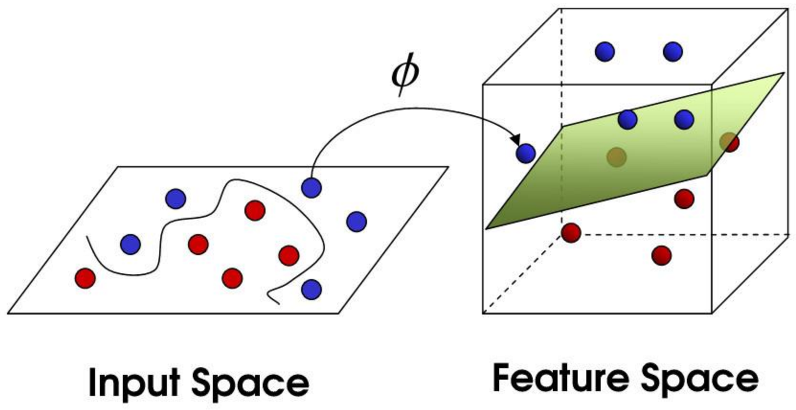
Objective function penalizes for misclassified instances and those within the margin

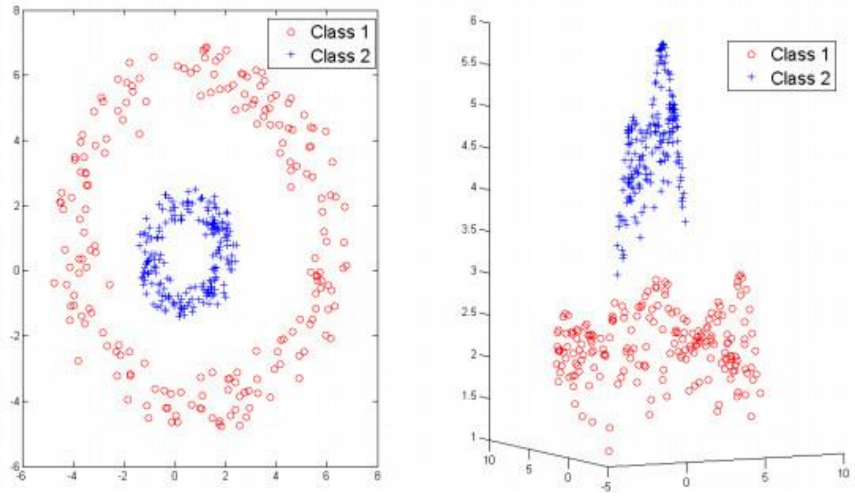
$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

C trades-off margin width and misclassifications

SVM Kernel Trick

What if samples cannot be linear separable. We will use the **kernel trick**. There are many types of kernel that can be used like linear, radial bases function. These kernels map samples from lower dimension to higher dimension, so that they can be separated in higher dimension.



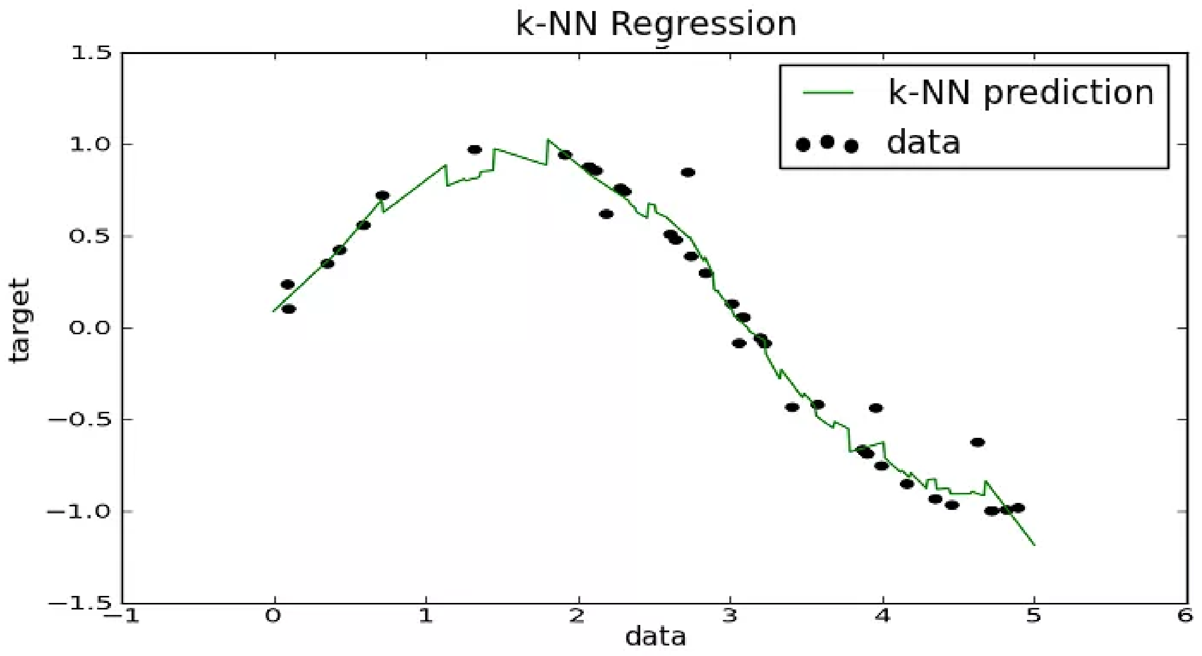


Regression

PREDICTING AN OUTPUT REAL VALUE

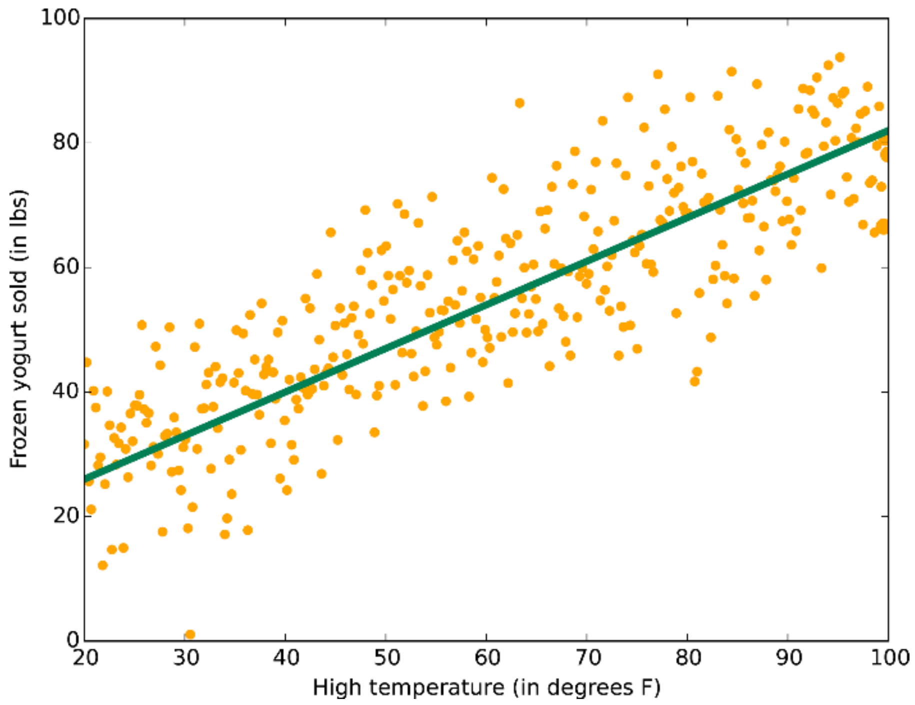
k-Nearest Neighbours

- A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbours.

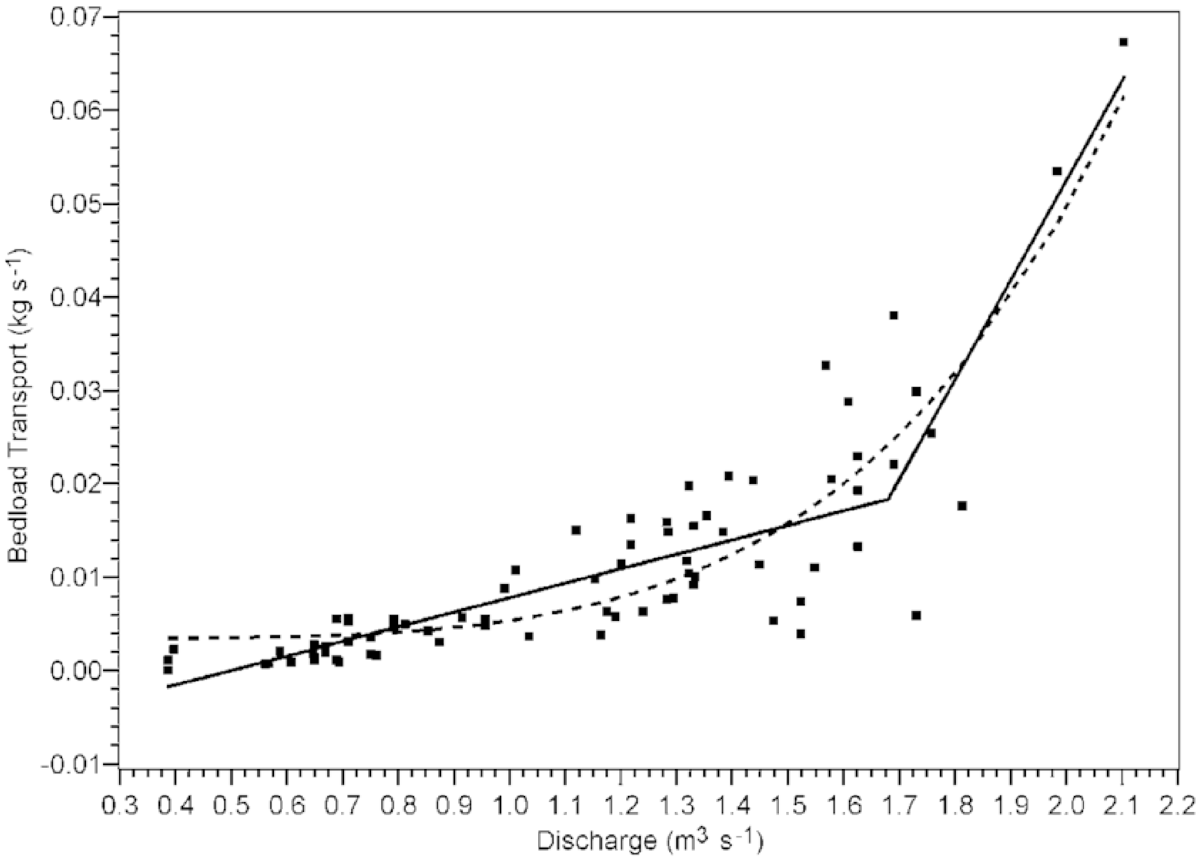


Linear Models

- Linear regression fits a line (or hyper-dimensional line) to the data set
- The linear model is simple but is surprisingly applicable to a wide variety of problems
- Almost any curve can be divided into segments to form a piecewise linear model

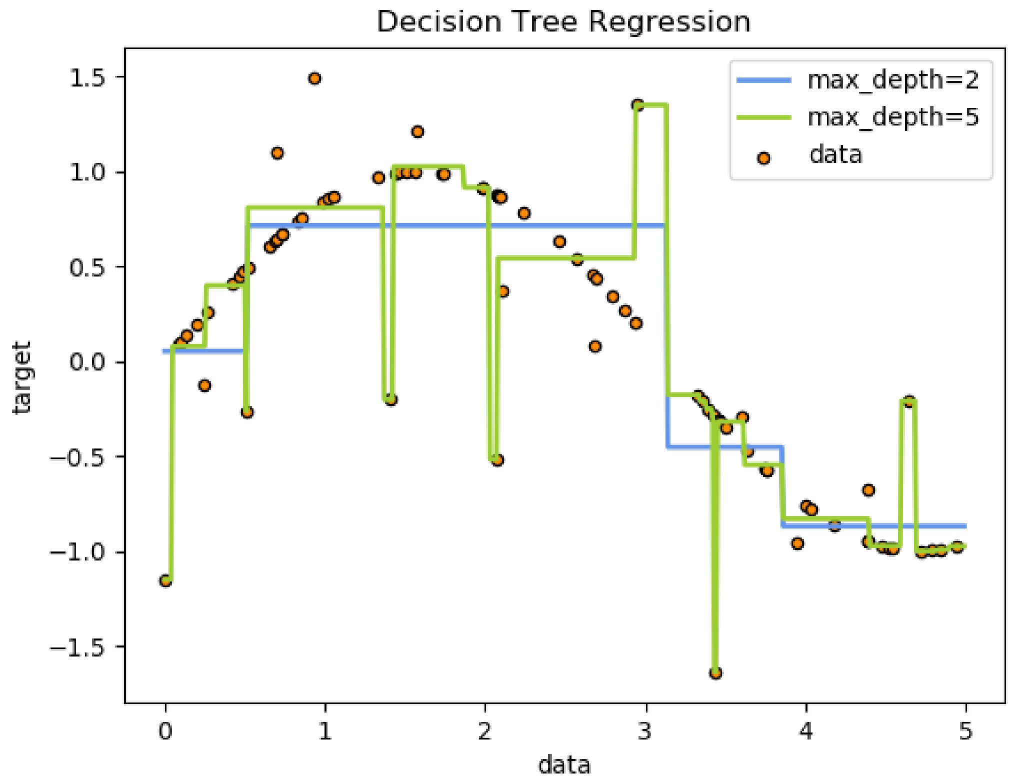


Piecewise Linear Model: Using two linear models to approximate curve

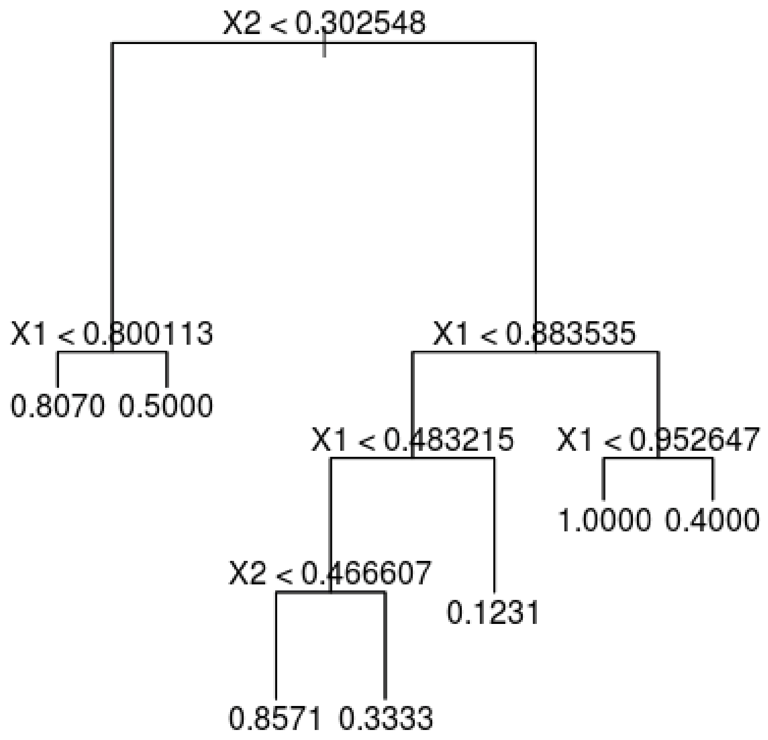


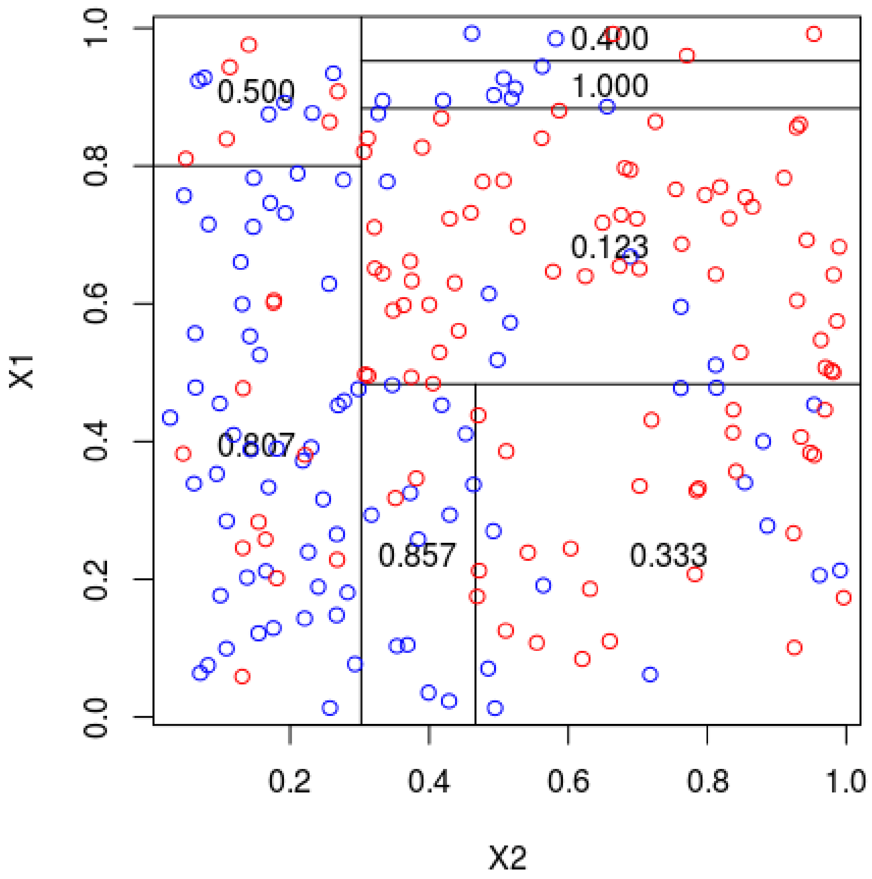
Decision Tree

- Decision tree regressors subdivide the feature space into regions of constant value

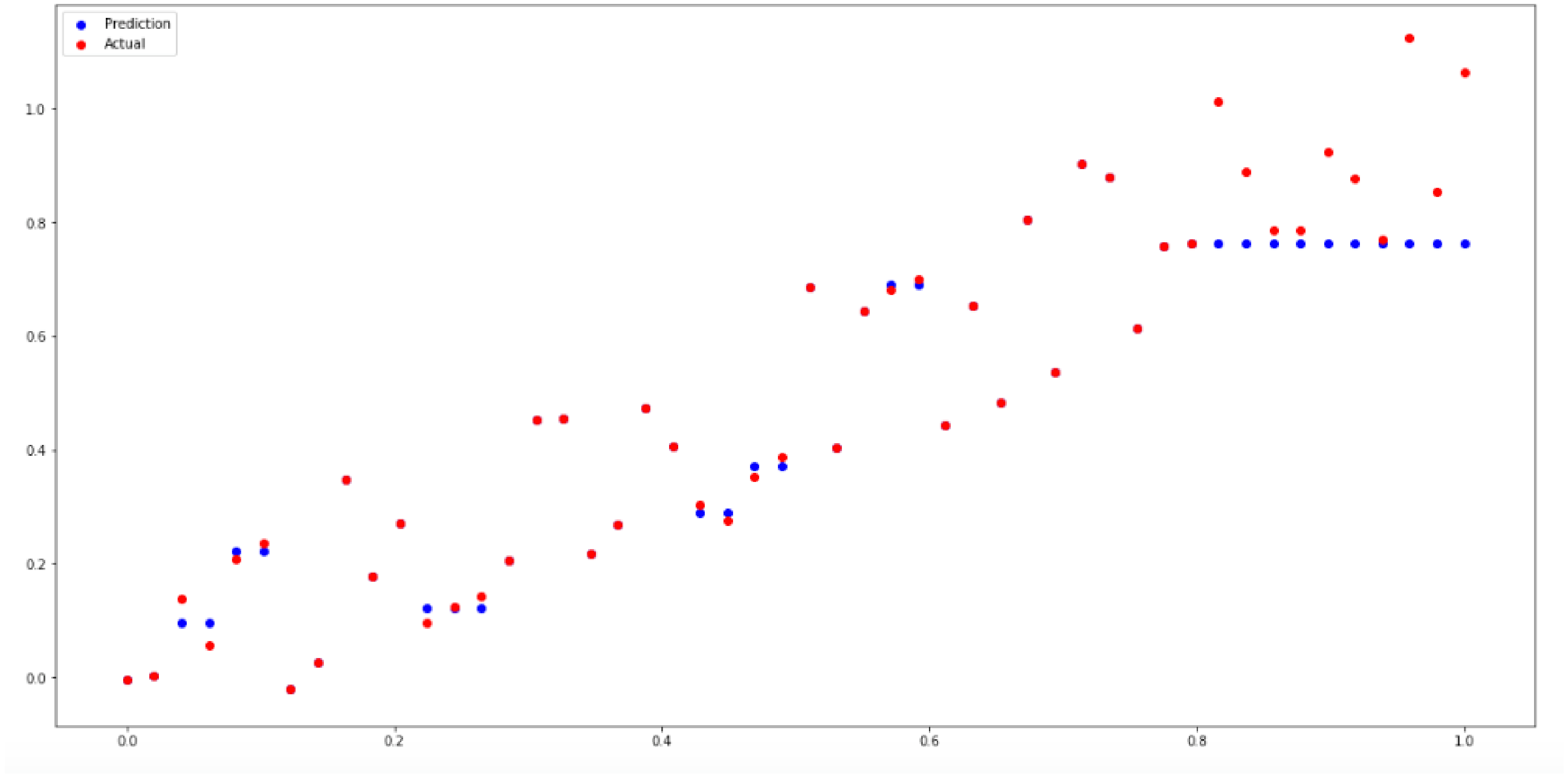


Decision Tree Regressor



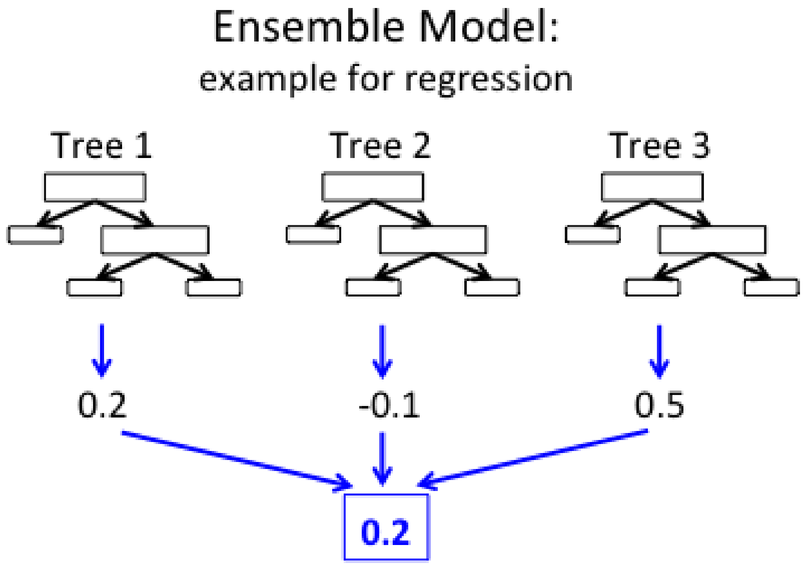


Decision Tree Regressor

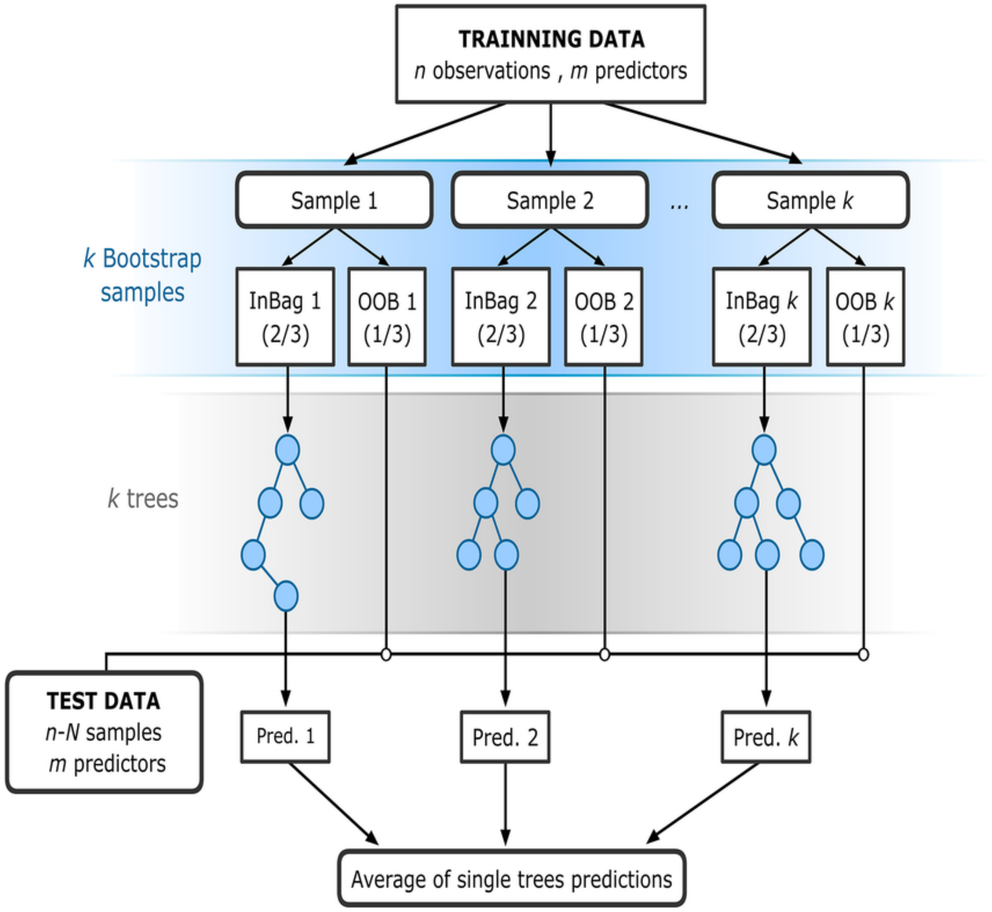


Random Forest Regressor

- Random Decision Regressor are all based on decision trees.
- In order to avoid this, a **large set of trees** are constructed with special mathematical care taken that the trees are not correlated.
- The average of this "**decision forest**" is a tree that avoids overfitting. Decision forests can use a lot of memory.

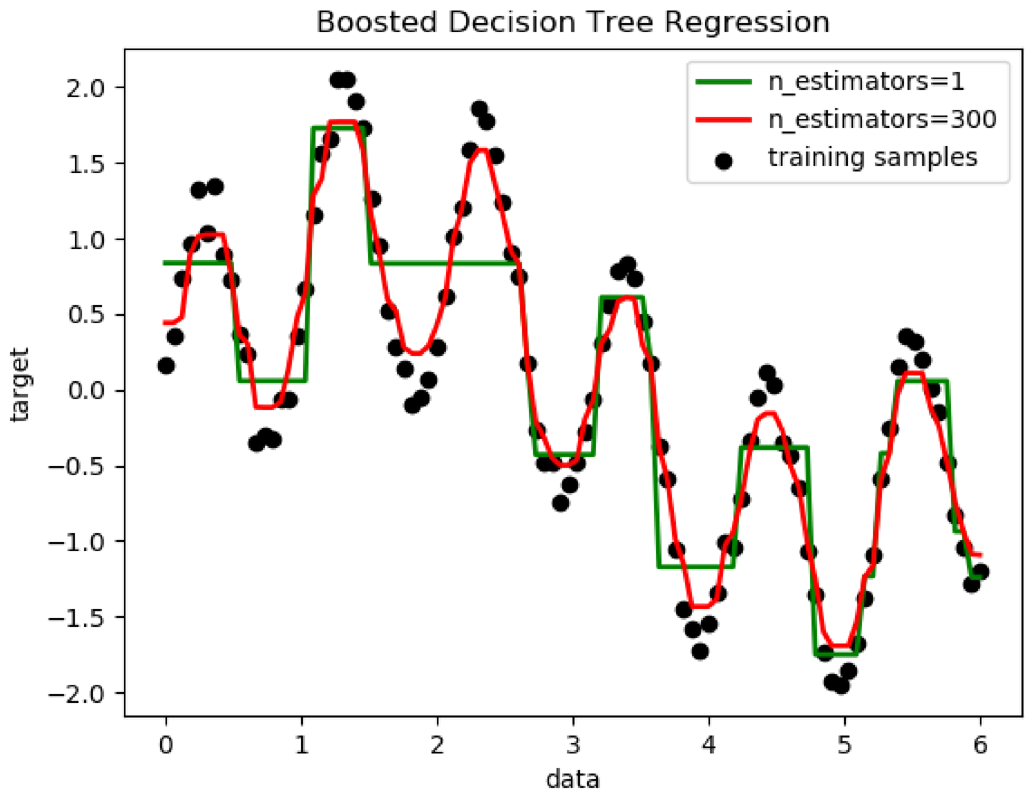


Random Forest Regressor



Boosted Regression Tree (Gradient Boosting)

- Boosted Decision Tree Regression algorithm creates an ensemble of regression trees using boosting.
- Boosting means that each tree is dependent on prior trees.
- The algorithm learns by fitting the residual of the trees that preceded it.



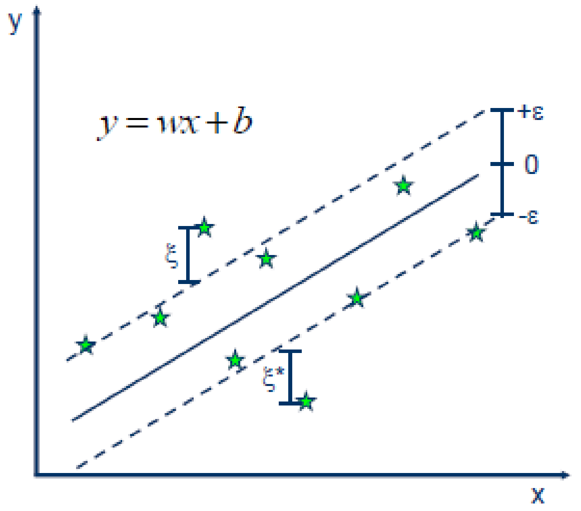
Support Vector Regressor (SVR)

Kernel: The function used to map a lower dimensional data into a higher dimensional data.

Hyper Plane: In SVM this is basically the separation line between the data classes. Although in SVR we are going to define it as the line that will help us predict the continuous value or target value

Boundary line: In SVM there are two lines other than Hyper Plane which creates a margin . The support vectors can be on the Boundary lines or outside it. This boundary line separates the two classes. In SVR the concept is same.

Support vectors: This are the data points which are closest to the boundary. The distance of the points is minimum or least.



- Minimize:

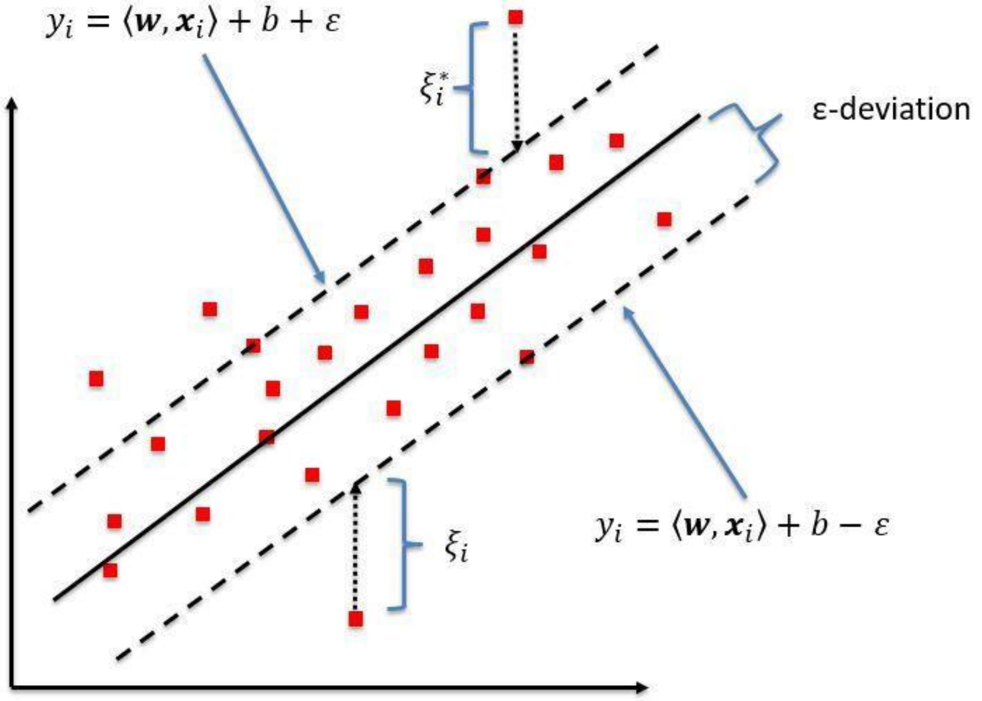
$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

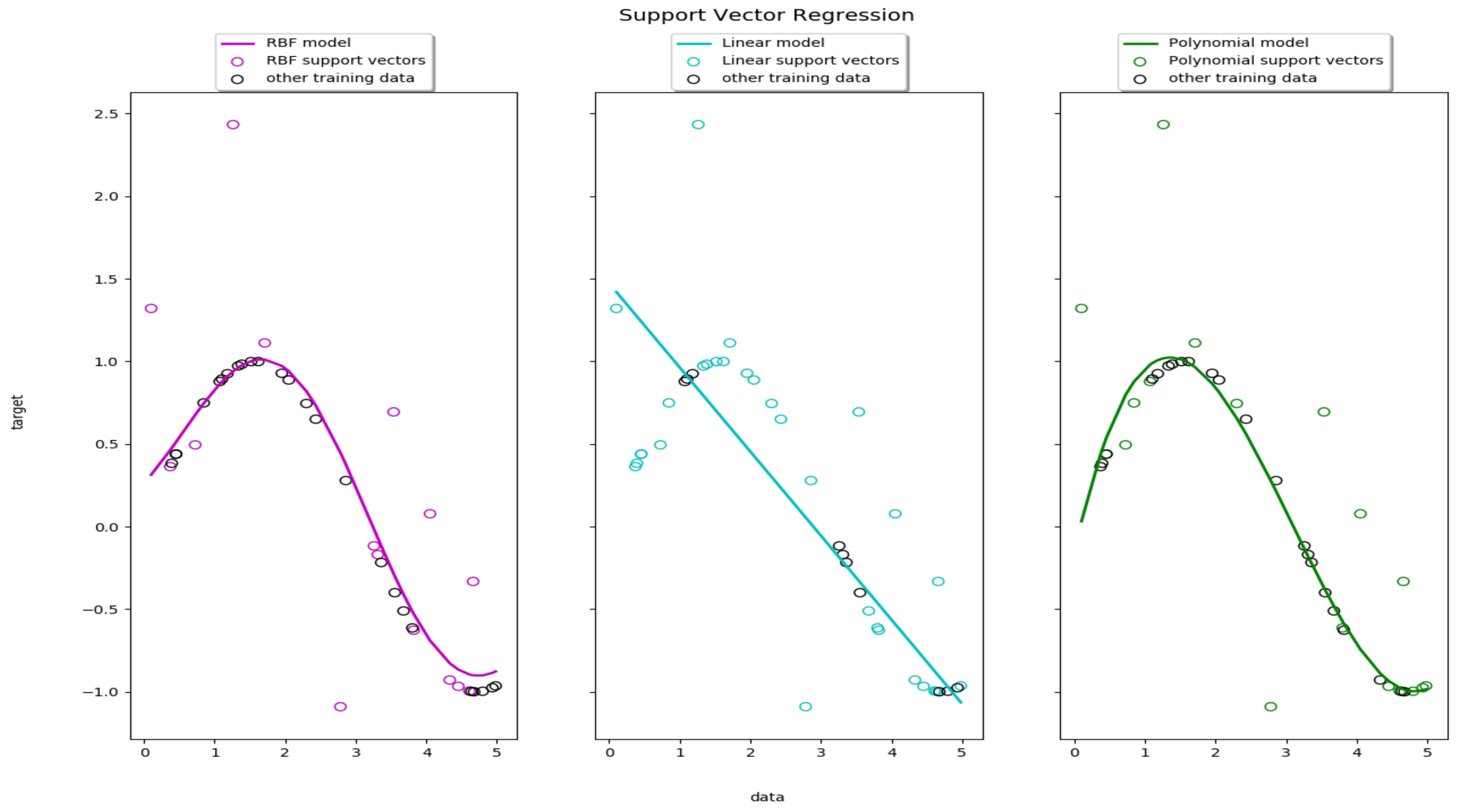
$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

SVR

What we are trying to do here is basically trying to decide a decision boundary at ‘e’ distance from the original hyper plane such that data points closest to the hyper plane or the support vectors are within that boundary line


$y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b - \varepsilon$

Support Vector Regression (SVR) using linear and non-linear kernels



Summary

We have learnt that:

- Supervised learning algorithms can be divided into 2 types
  - Classification – desired target output is predicting a label/class
  - Regression – desired target output is predicting a real value
- There is no magic bullet algorithm (no-free lunch theorem) that will be best algorithm to use with all data sets
- Goldilocks problem — we desire just the right fit; neither overfitting nor underfitting.
- Bias-variance trade-off guides how we would tune the models