Topic 4
Python Supervised Machine Learning

IT8302 APPLIED MACHINE LEARNING

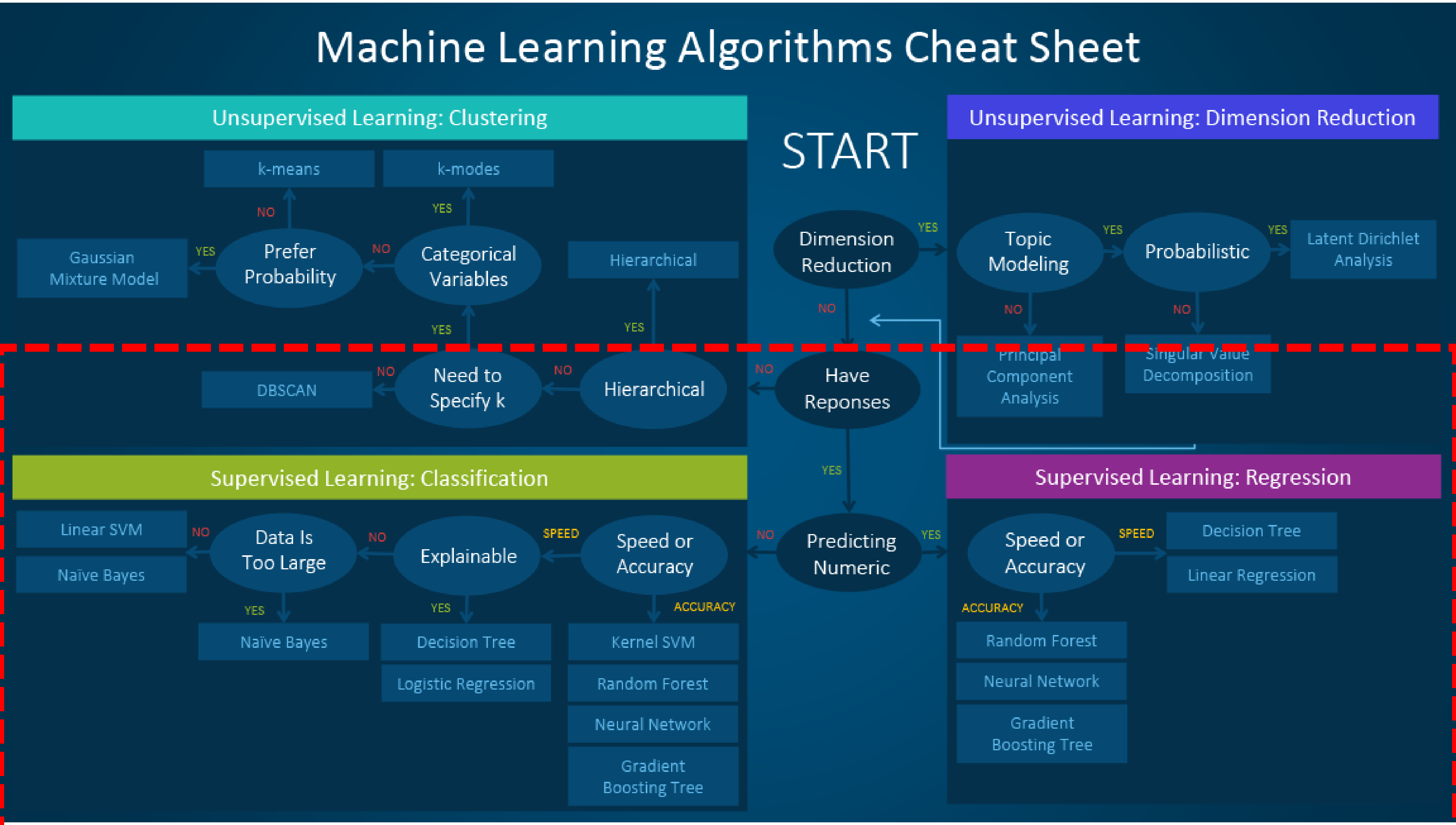
Learning Outcomes

- Apply Classification
  - Explain use cases for classification
  - Explain generalization, overfitting and underfitting for classification
  - Classification Algorithms
    - Understand k-Nearest Neighbours (KNN classifier)
    - Understand Linear Models (Logistic regression classifier)
    - Understand Naïve Bayes Classifiers
    - Understand Decision Trees Classifier
    - Understand Support Vector Machine Classifier (SVC)
  - Model evaluation for classifier

Learning Outcomes

- Apply Regression
  - Explain use cases for regression
  - Explain generalization, overfitting and underfitting for regression
  - Regression Algorithms
    - Understand k-Nearest Neighbours (KNN Regressor)
    - Understand Linear Models (Linear Regression)
    - Understand Decision Tree Regressor
    - Understand Support Vector Regressor
  - Model evaluation for regressor

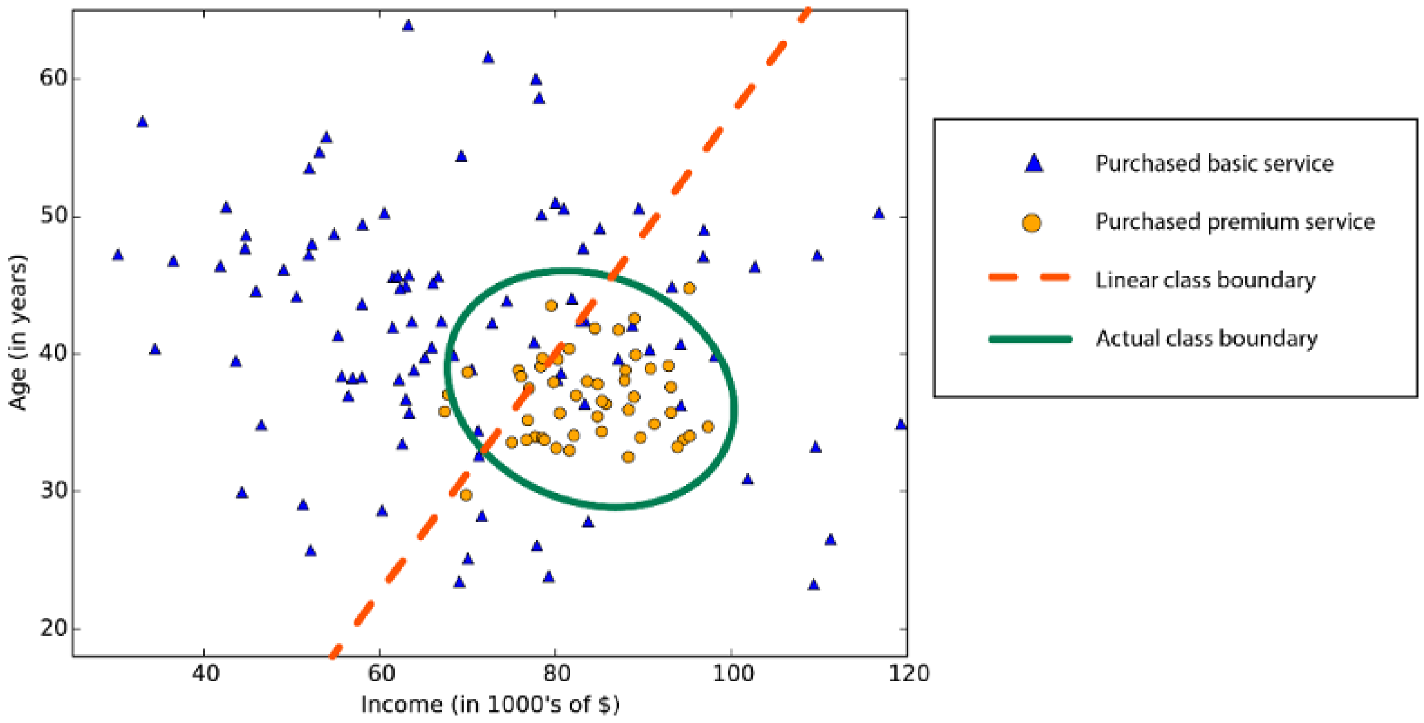
Supervised Machine Learning Algorithms



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

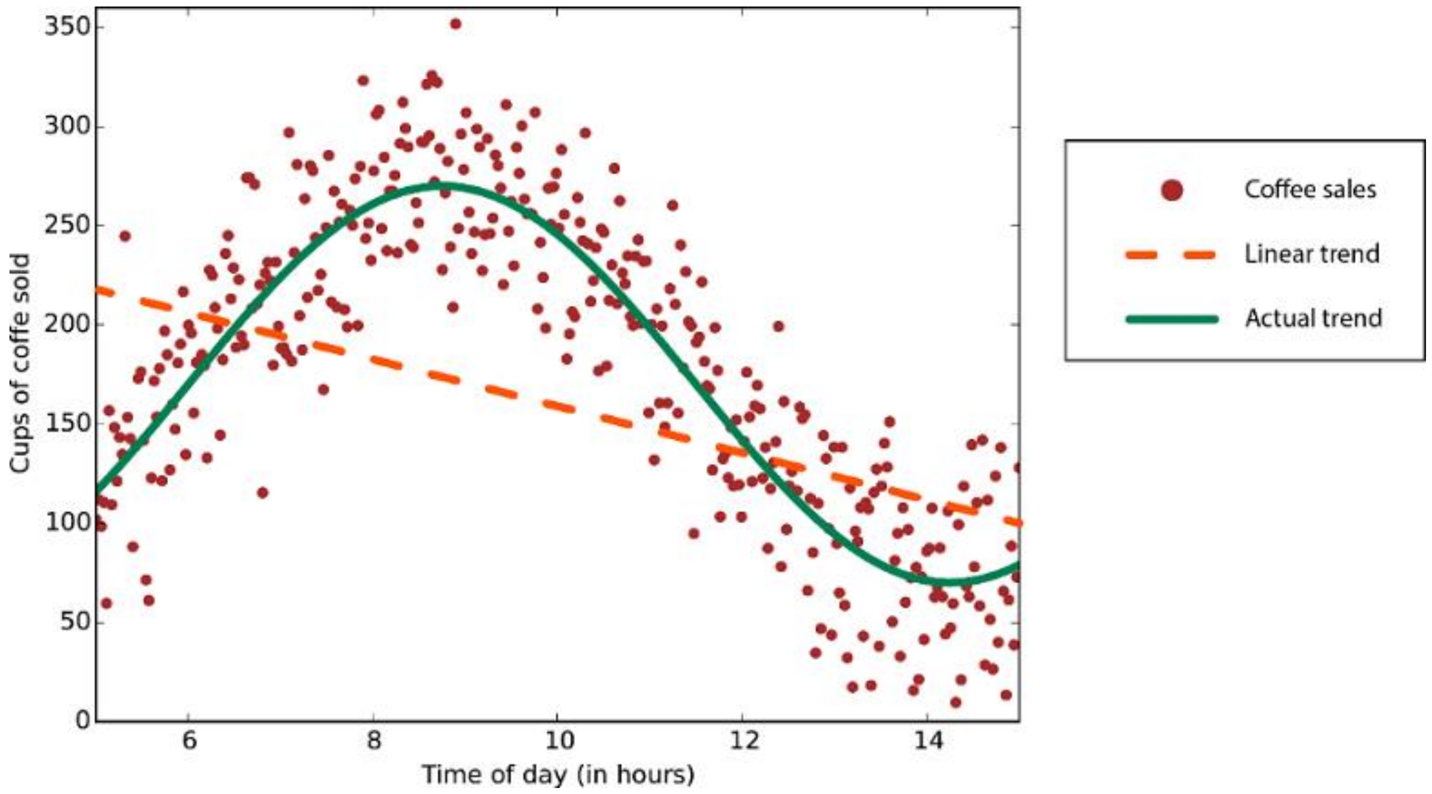
Classification

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



Regression

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



Classification

PREDICTION AN OUTPUT LABEL/CLASS

Classification

USE CASES

Classification use cases

MARKETING AND SALES

Digital marketing and online-driven sales are the first application fields that you may think of for machine learning adoption. People interact with the web and leave a detailed footprint to be analyzed

Churn. A customers who churns is one who cease to complete target actions (e.g. add to cart, leave a comment, checkout, etc.) during a given period.

Response. We want to target ad or marketing campaign to customers who are likely to respond to our campaign than to those who would treat it as spam.

Classification use cases

EDUCATION & LEARNING

In modern education, a lot of data is collected about the student's performance from the Learning Management System (LMS) and from other tools. This data could be analyzed to predict student performance

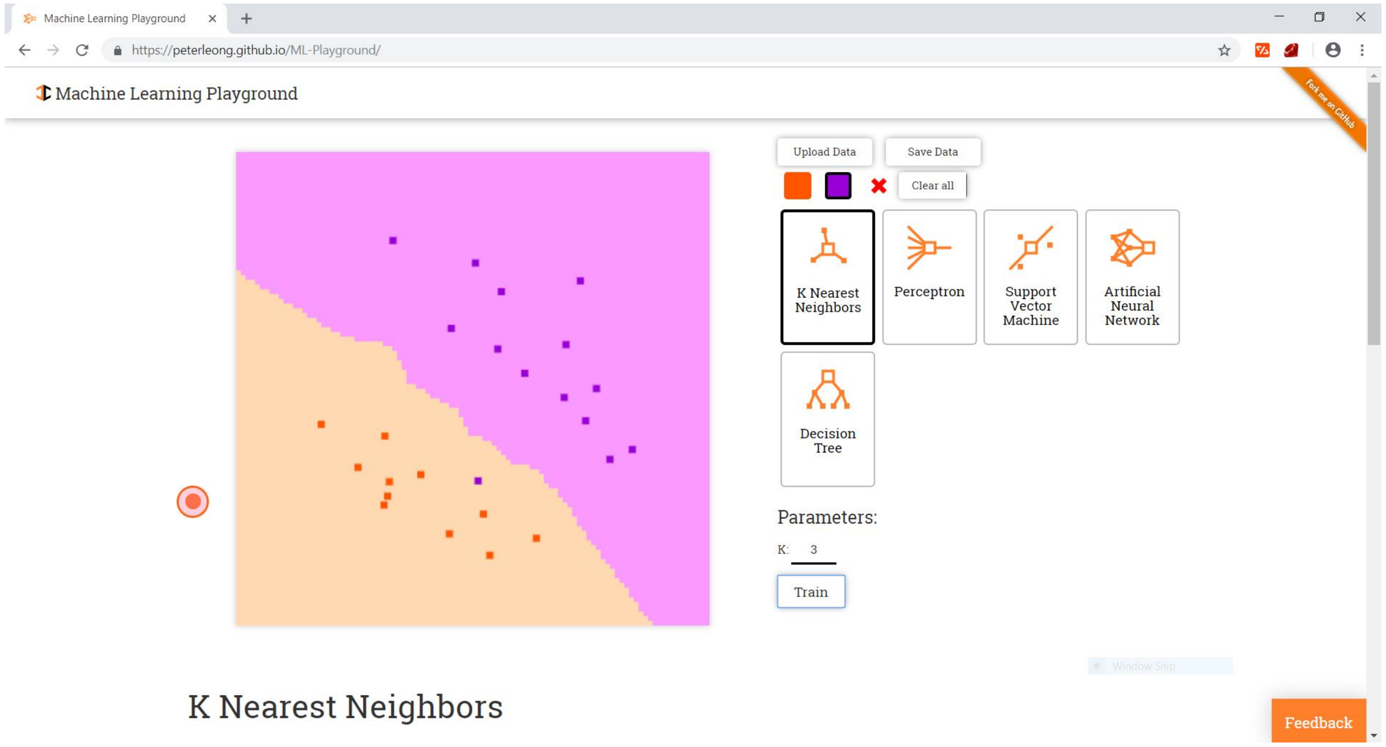
Performance prediction. We could use the data collected to spot an emergent trend for poor performance and provide early intervention.

Personalized learning. We could analyze the data to predict student performance and provide personalized learning path to plug gaps in learning.

Activity

Try out the different classification algorithms in the ML Playground.

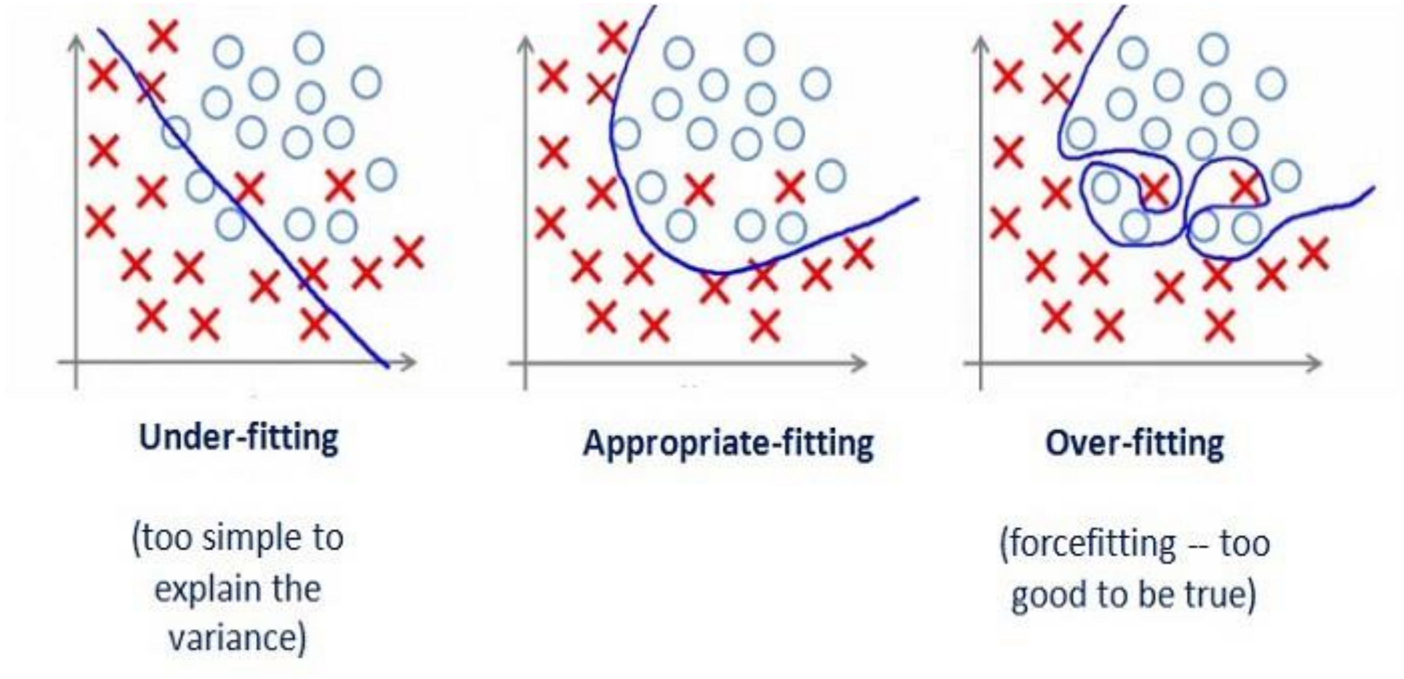
https://peterleong.github.io/ML-Playground/



Classification

GENERALIZATION, UNDERFITTING, OVERFITTING

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)

Bias-Variance Trade-off

OR HOW DO WE DETECT UNDERFITTING AND OVERFITTING

https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

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.

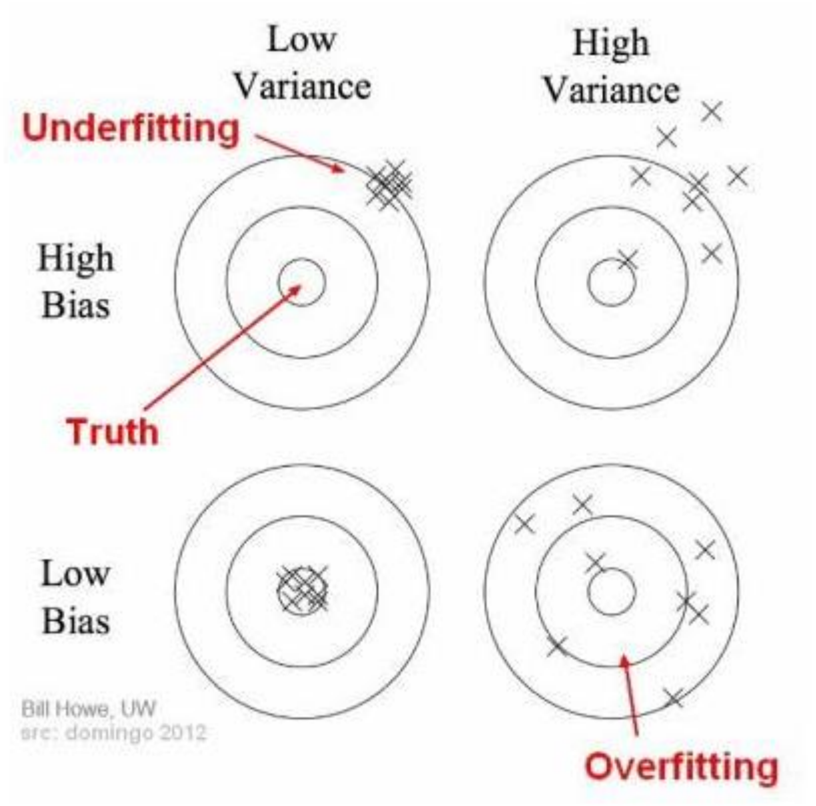
VARIANCE

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

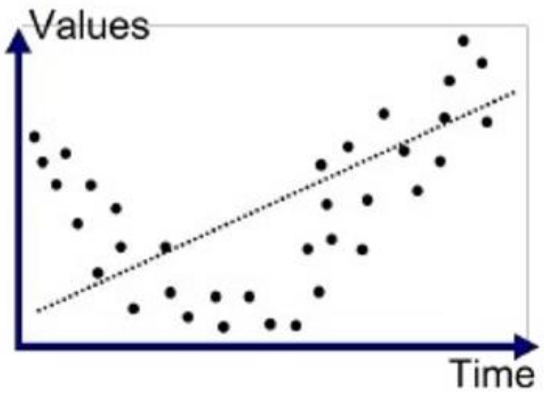
Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before.

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

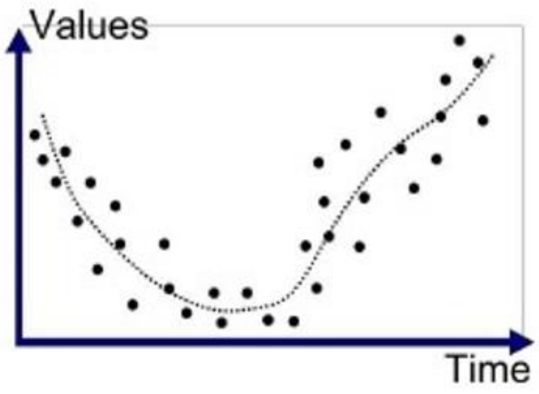
Bias-variance trade-off



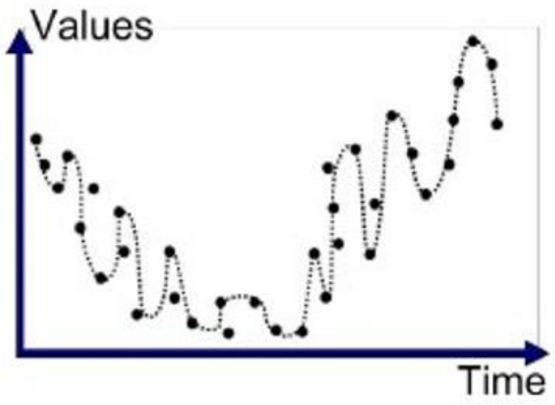
Underfit and Overfit



Underfitted

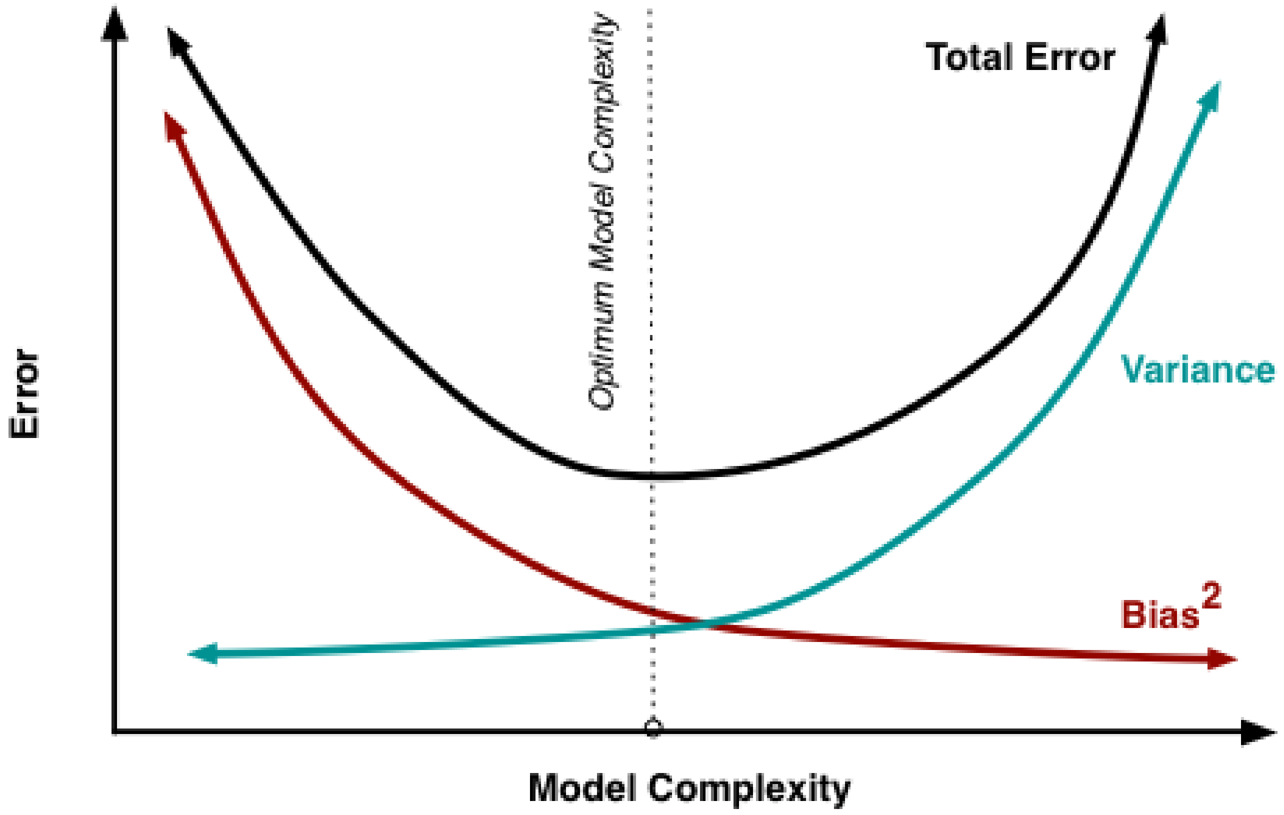


Good Fit/Robust

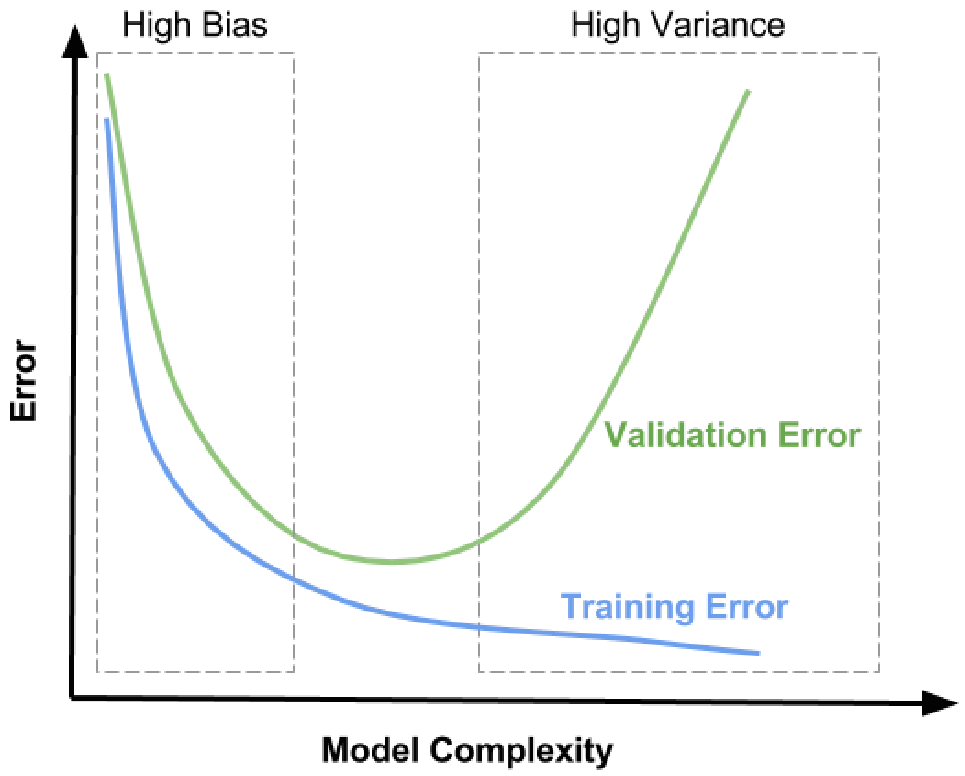


Overfitted

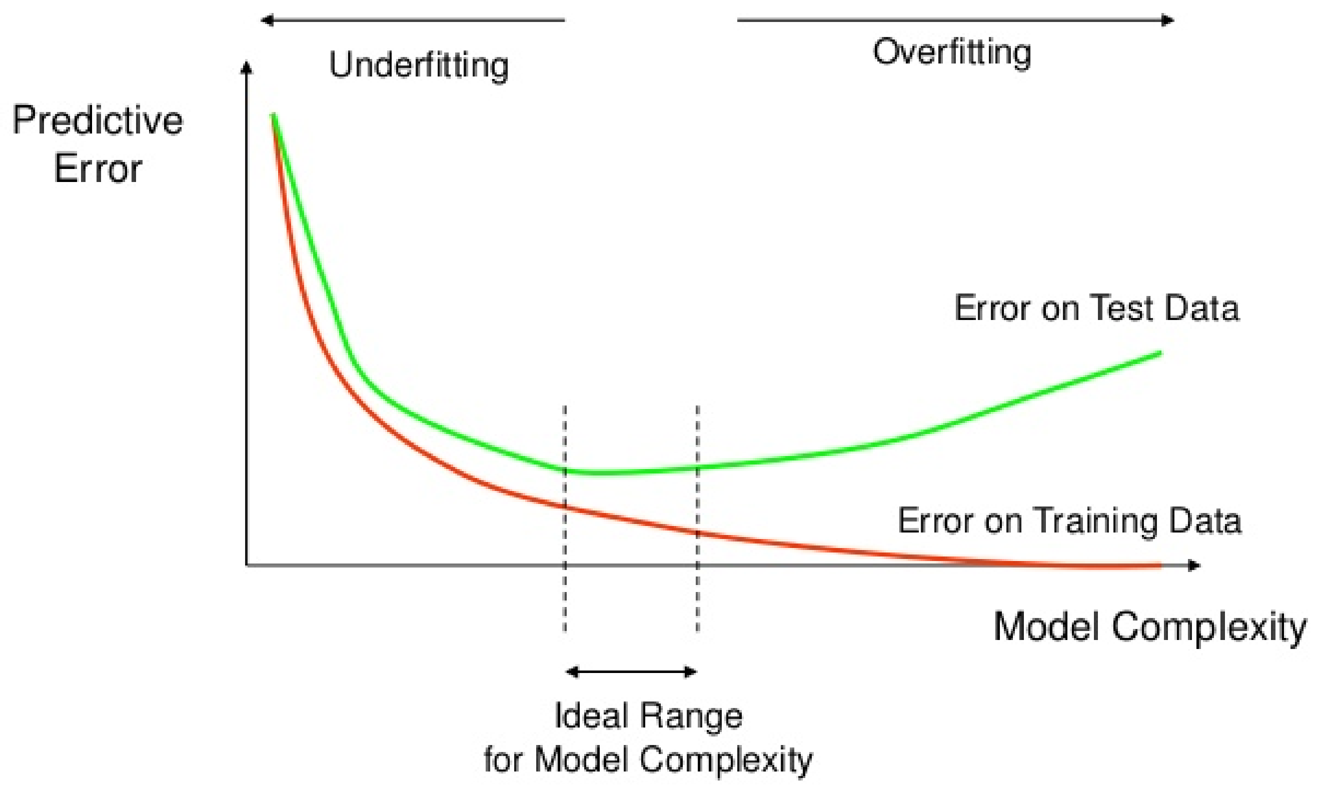
Bias-variance trade-off



Bias-variance trade-off



The model complexity must be correct to get the right fit

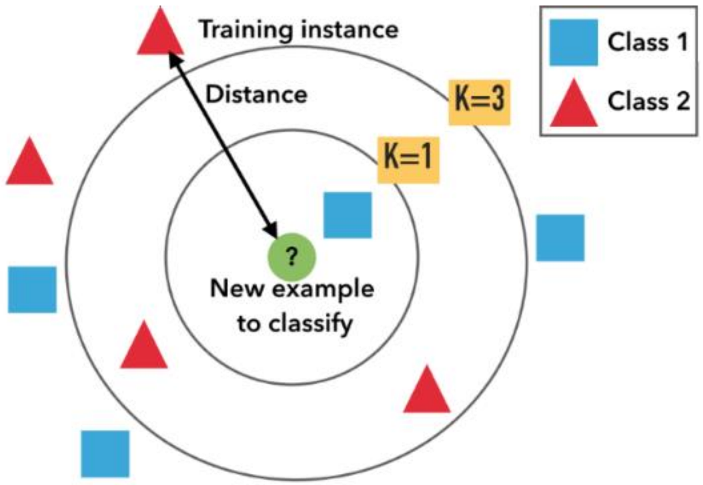


Classification

ALGORITHMS

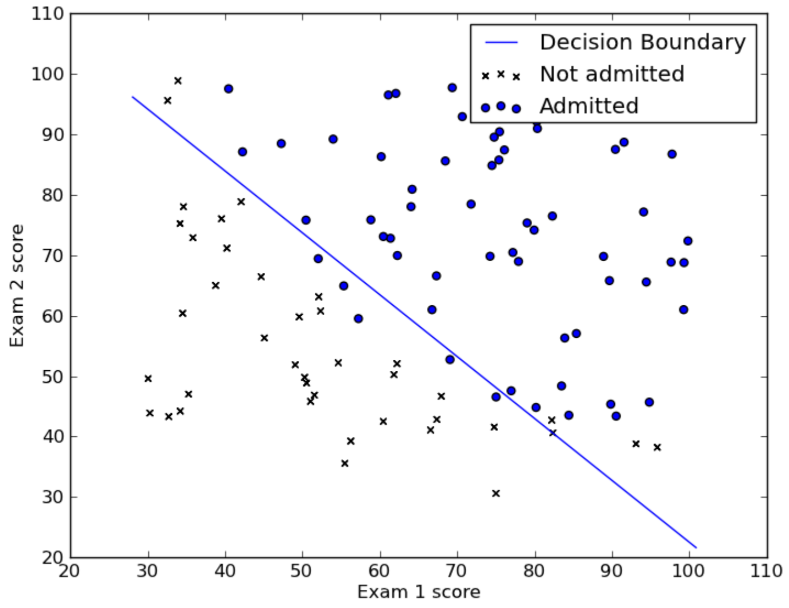
k-Nearest Neighbours

- 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

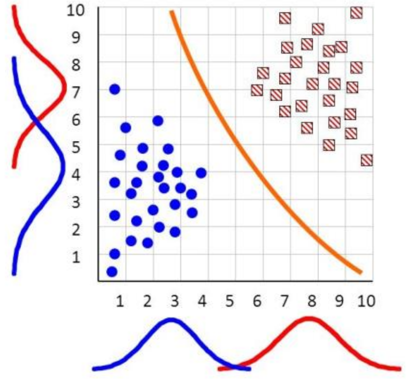
- All naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

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

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

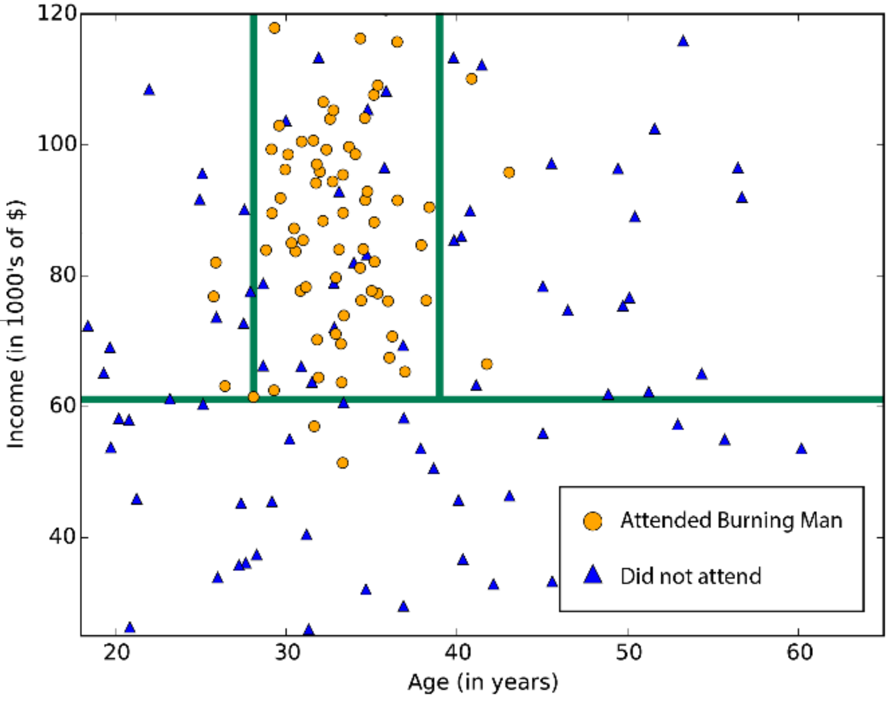
$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

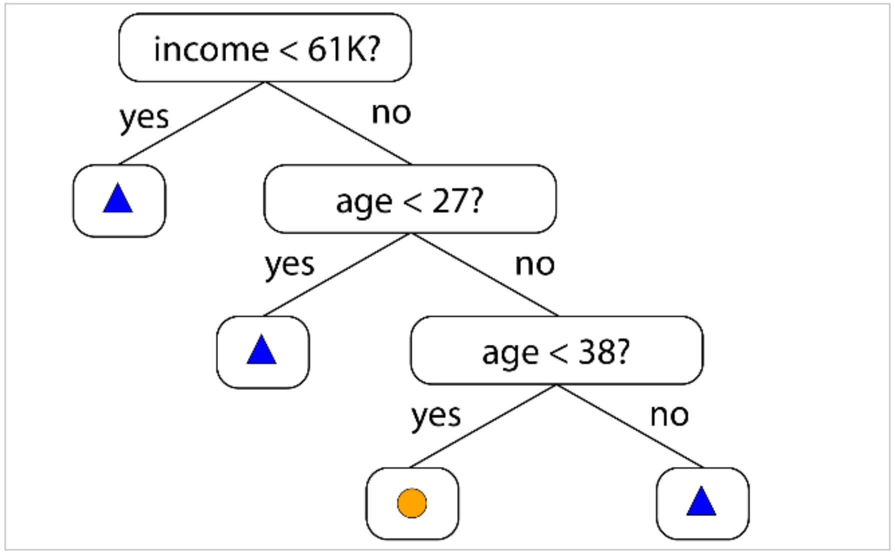
The Naïve Bayesian Classifier has a quadratic decision boundary



Decision Trees

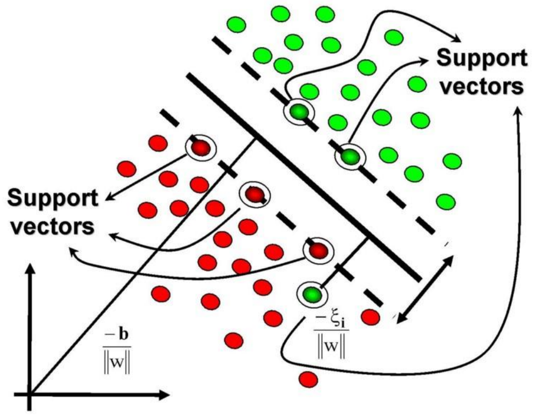
- There are many variants of decision trees, but they all do the same thing—subdivide the feature space into regions with mostly the same label.
- These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.

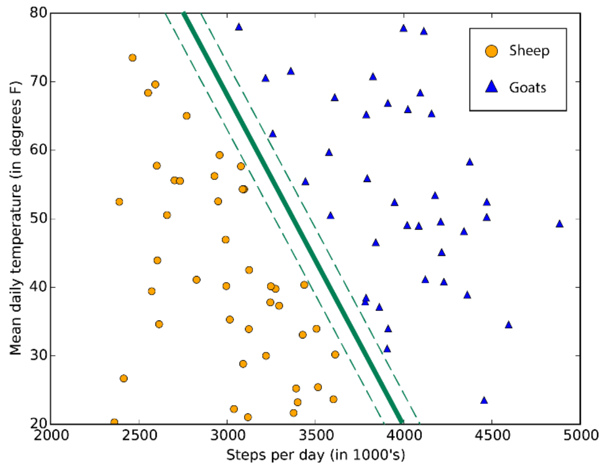




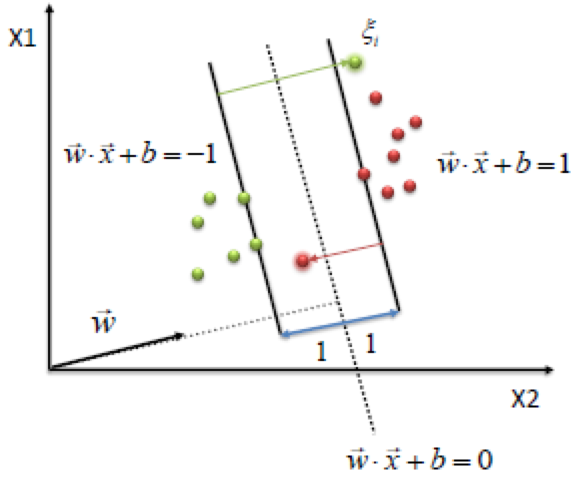
Support Vector Machines

- 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 features 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.





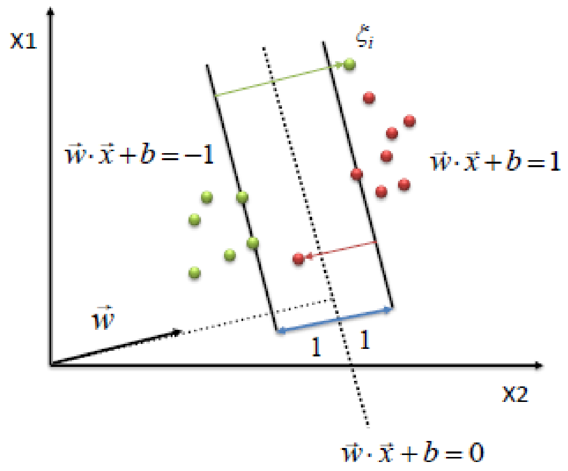
Support Vector Classifier


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

slack variable:

ξ_i

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


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

Constraint becomes :

$$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$$
$$\xi_i\geq 0$$

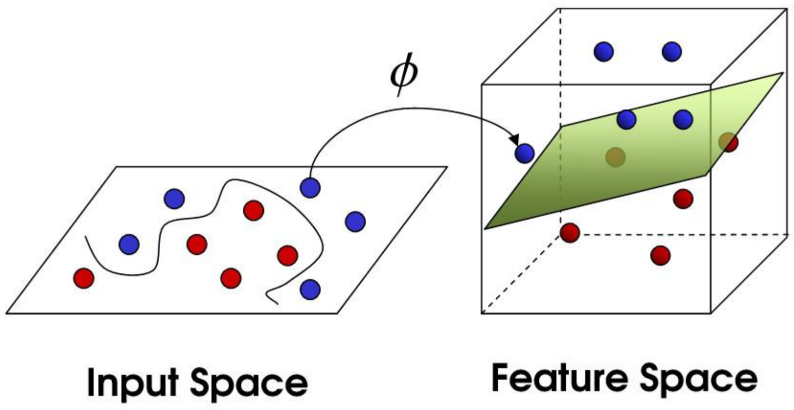
Objective function penalizes for misclassified instances and those within the margin

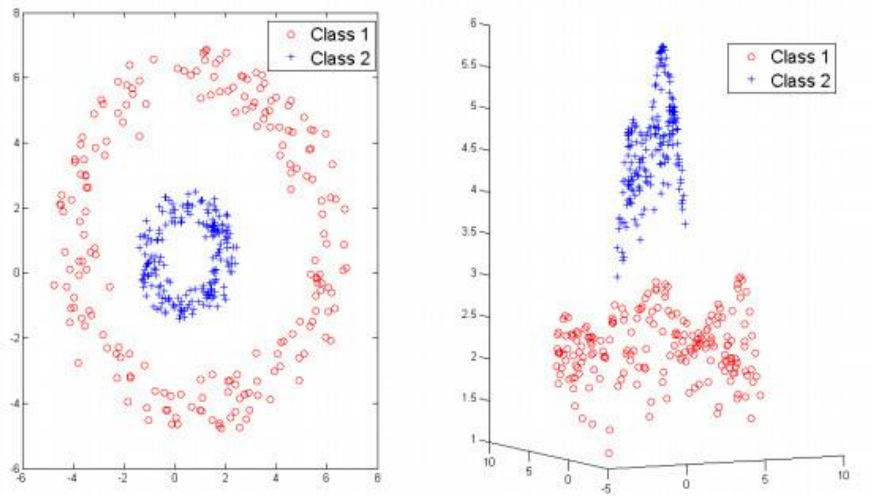
$$\min\frac{1}{2}\|w\|^2+C\sum_i\xi_i$$

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.





Classification

MODEL EVALUATION

(RECAP)

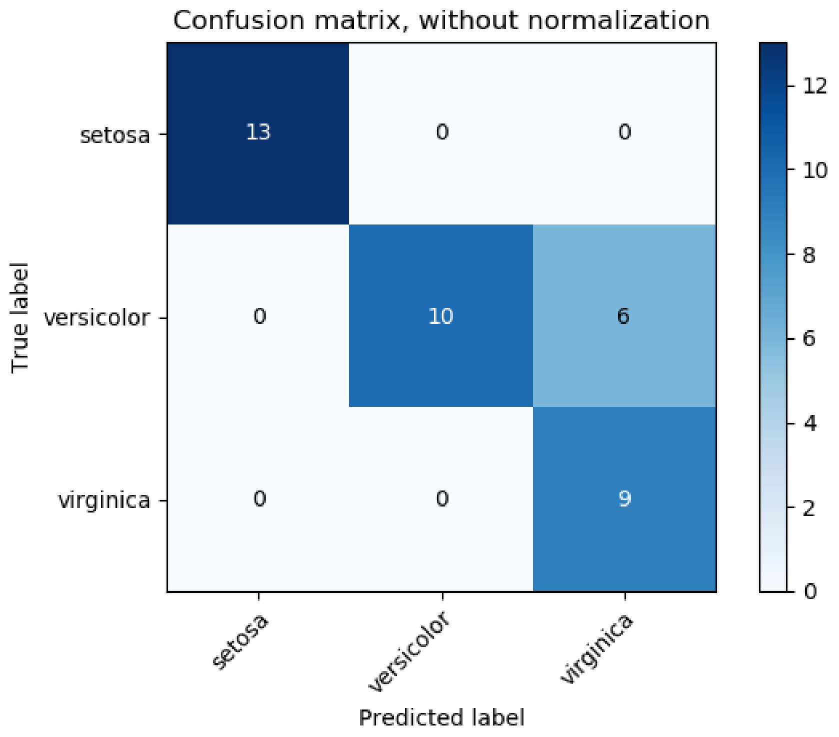
Model evaluation for classifiers

- The metrics used for classification are totally different from those used regression.
- The metrics for classifiers measure that amount of misclassification (i.e. assigning the wrong label to the output) that happens.
- In scikit-learn, the metrics for regression are found at https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
- The accuracy_score function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.
- The best possible score is 1.0 (100% accurate). The worse is 0.5 which corresponds to random guessing. Score of 0.0 mean there is mislabelling of the correct answer since being 100% wrong is not possible without a certain degree of correct scoring/prediction.

Metrics for Classification: Confusion Matrix

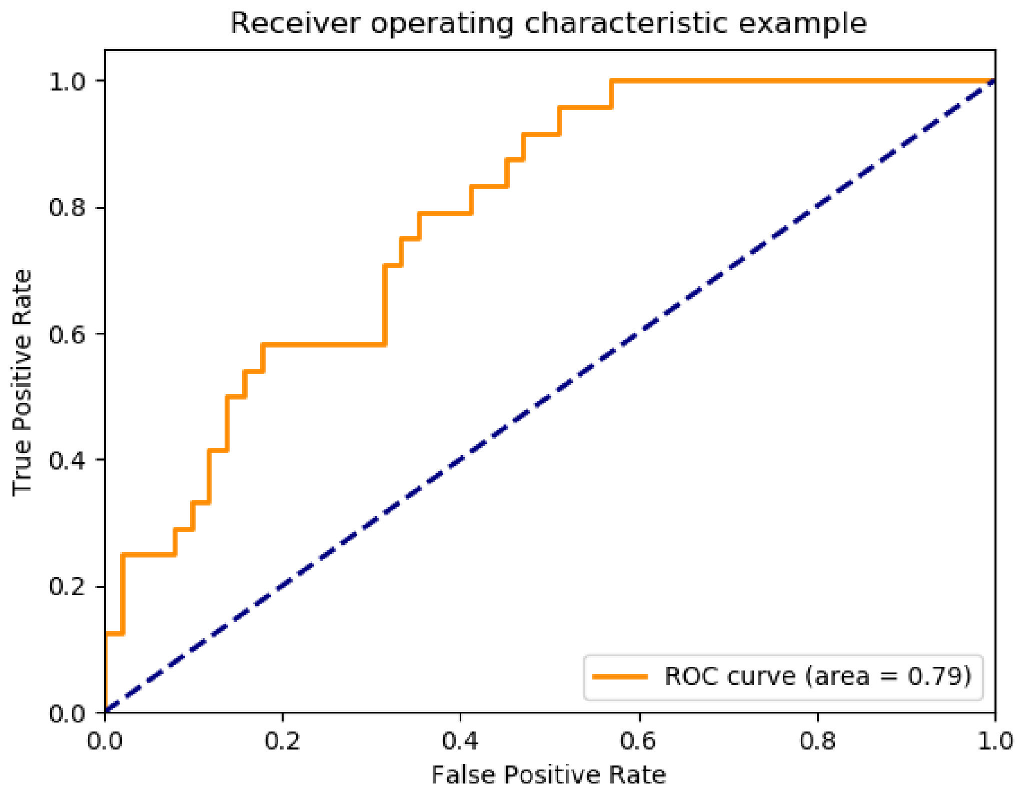
https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix

The confusion_matrix function evaluates classification accuracy by computing the confusion matrix with each row corresponding to the true class



Metrics for two-class/binary classification: ROC curve

https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics



Metrics for two-class/binary classification: ROC curve

A receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives (TPR = true positive rate) vs. the fraction of false positives out of the negatives (FPR = false positive rate), at various threshold settings. TPR is also known as sensitivity, and FPR is one minus the specificity or true negative rate.

Metrics for two-class/binary classifier: Area Under ROC Curve (AUC)

The roc_auc_score function computes the area under the receiver operating characteristic (ROC) curve, which is also denoted by AUC or AUROC. By computing the area under the roc curve, the curve information is summarized in one number.

Best possible AUC is 1.0.

Worst AUC is 0.5 (random guessing)

Regression

PREDICTING AN OUTPUT REAL VALUE

Regression

USE CASES

Regression use cases

MARKETING AND SALES

Digital marketing and online-driven sales are the first application fields that you may think of for machine learning adoption. People interact with the web and leave a detailed footprint to be analyzed

Dynamic Pricing. For fast moving goods, dynamic adjustment of the pricing according to predicted demand (for e.g., Grab) helps to optimize profits.

Inventory Level. Predict the correct level of stocks to hold for product to avoid unnecessary inventory costs.

Regression use cases

FINANCE

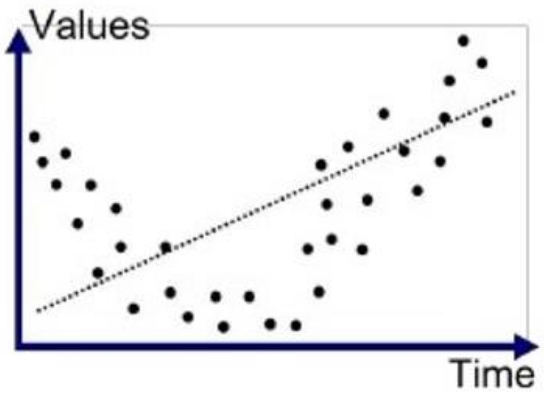
Banks and other financial institutions collect a lot of data about their customer transactions. Trading houses also rely heavily on real-time market information.

Financial Trading. Many people are eager to be able to predict what the stock markets will do on any given day — for obvious reasons. But machine learning algorithms are getting closer all the time. Many prestigious trading firms use proprietary systems to predict and execute trades at high speeds and high volume.

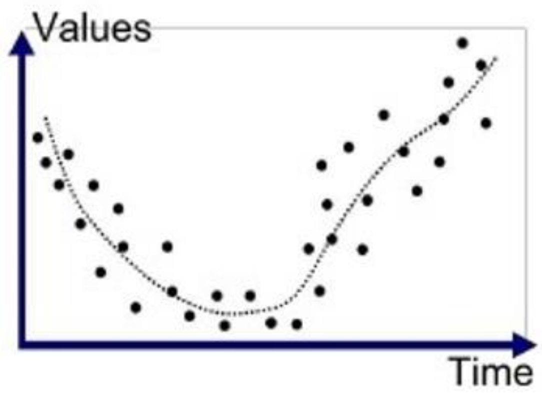
Regression

GENERALIZATION, UNDERFITTING, OVERFITTING

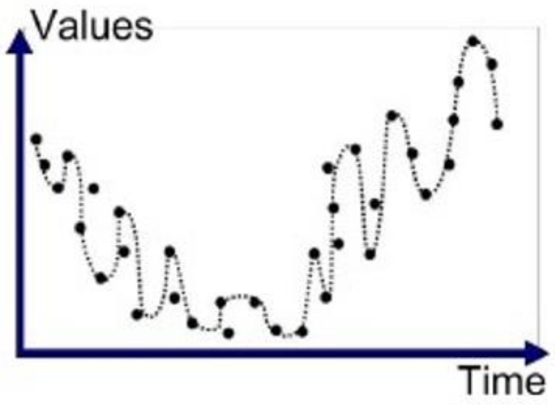
Right-fit for Regression: one that generalizes the trend



Underfitted



Good Fit/Robust



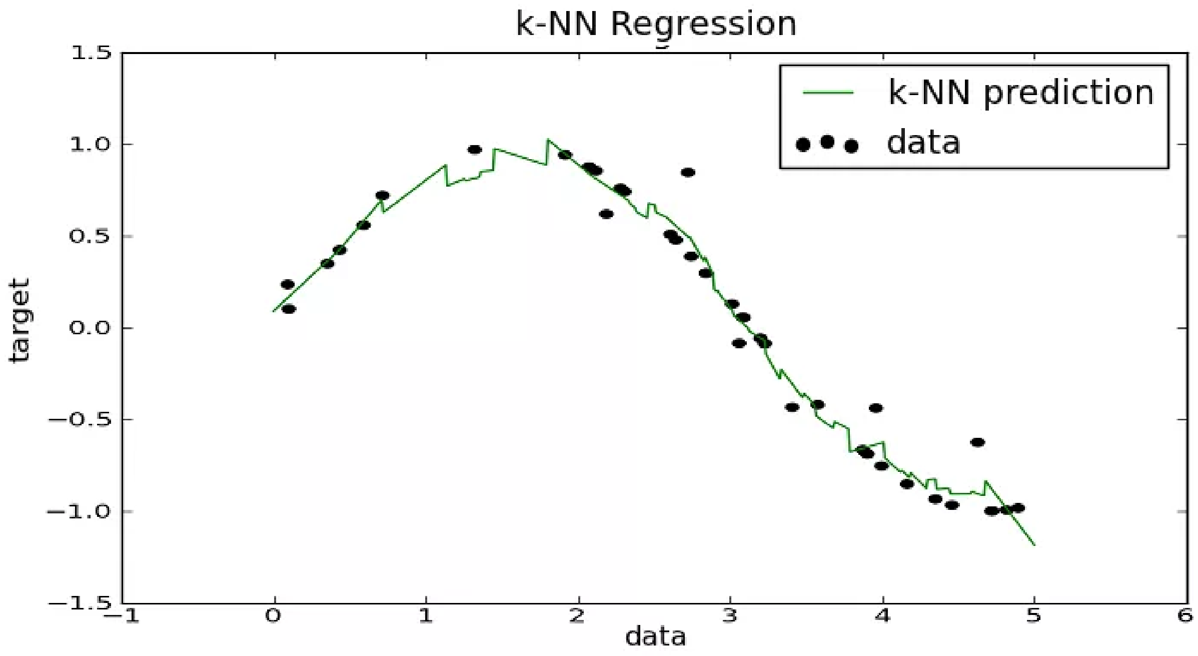
Overfitted

Regression

ALGORITHMS

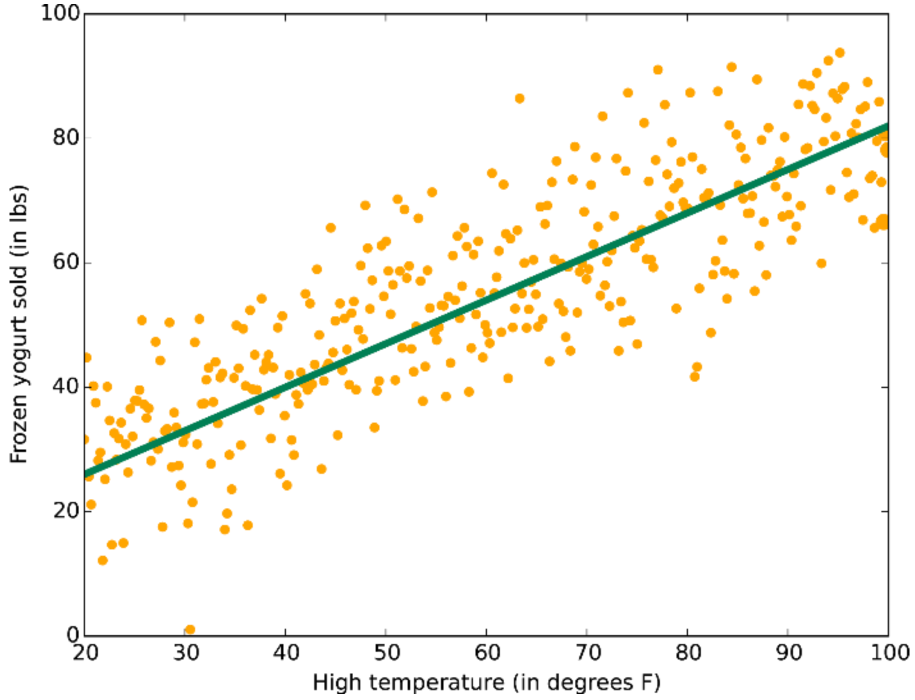
k-Nearest Neighbours

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

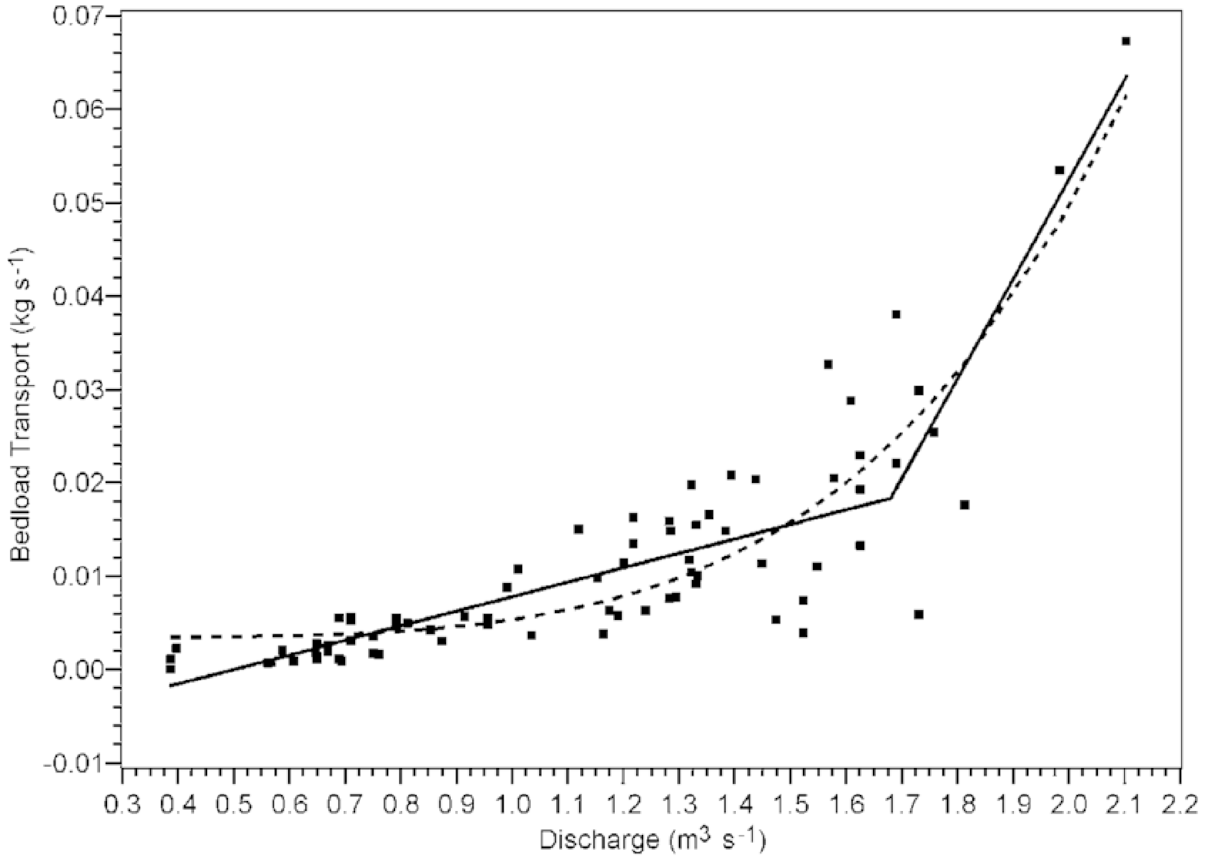


Linear Models

- 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

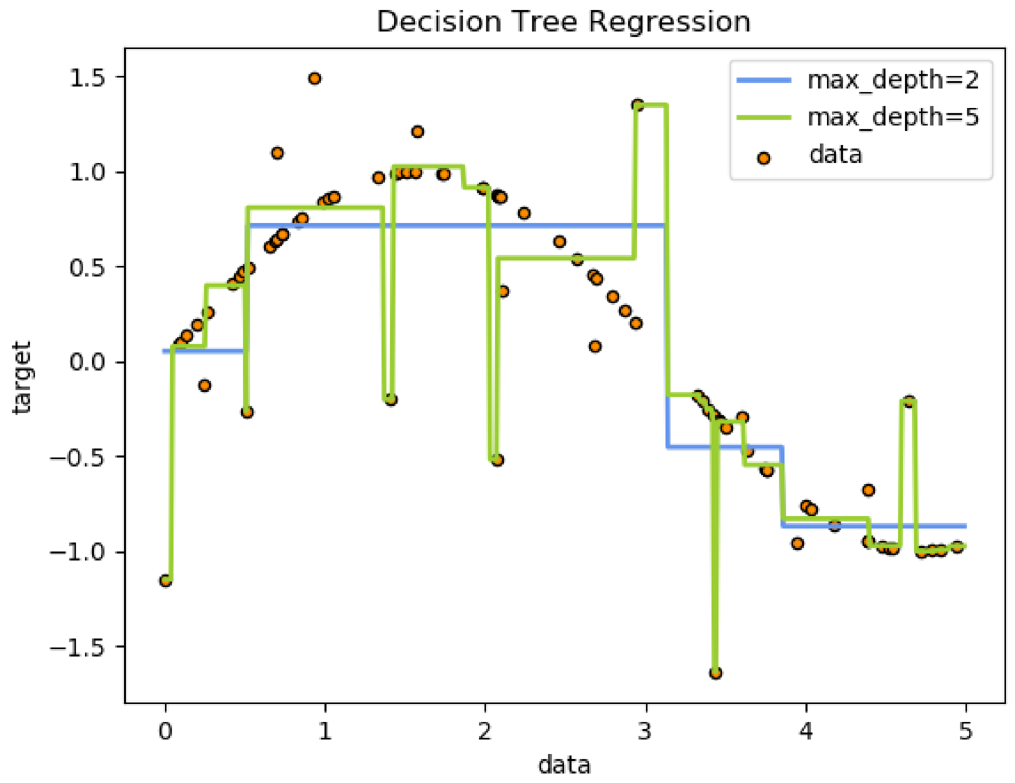


Piecewise Linear Model: Using two linear models to approximate curve

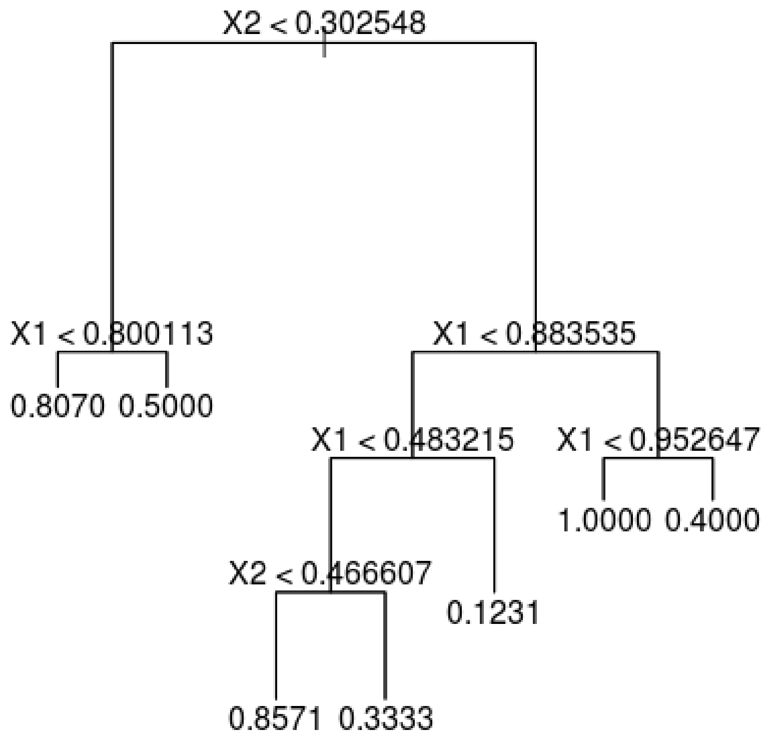


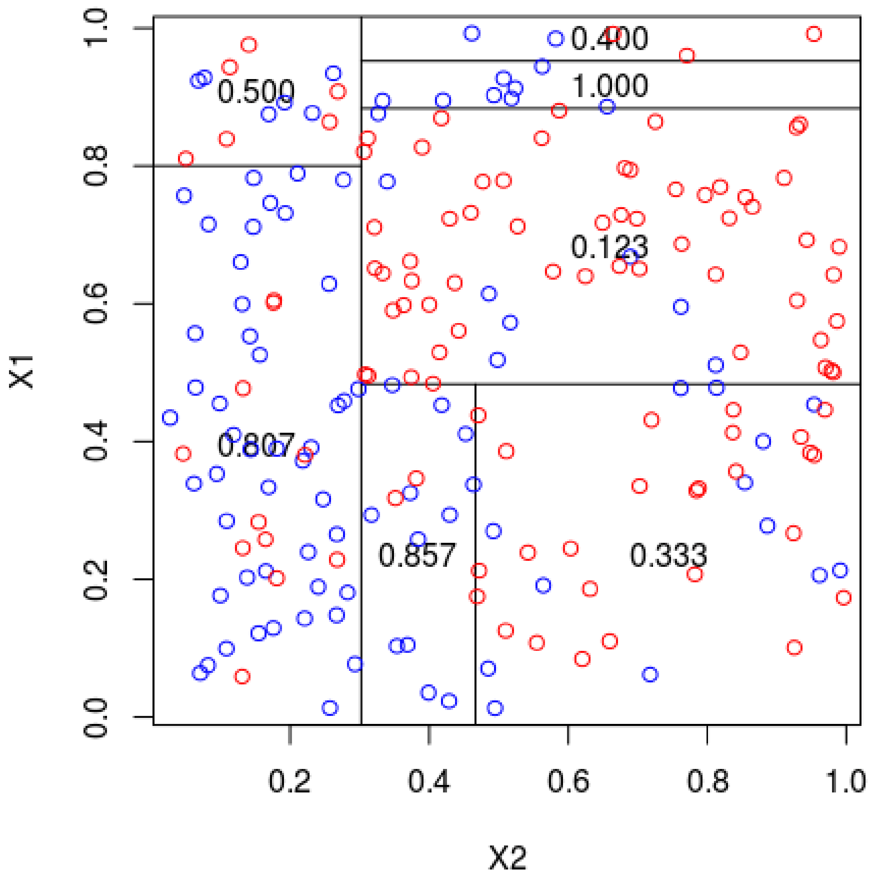
Decision Tree

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

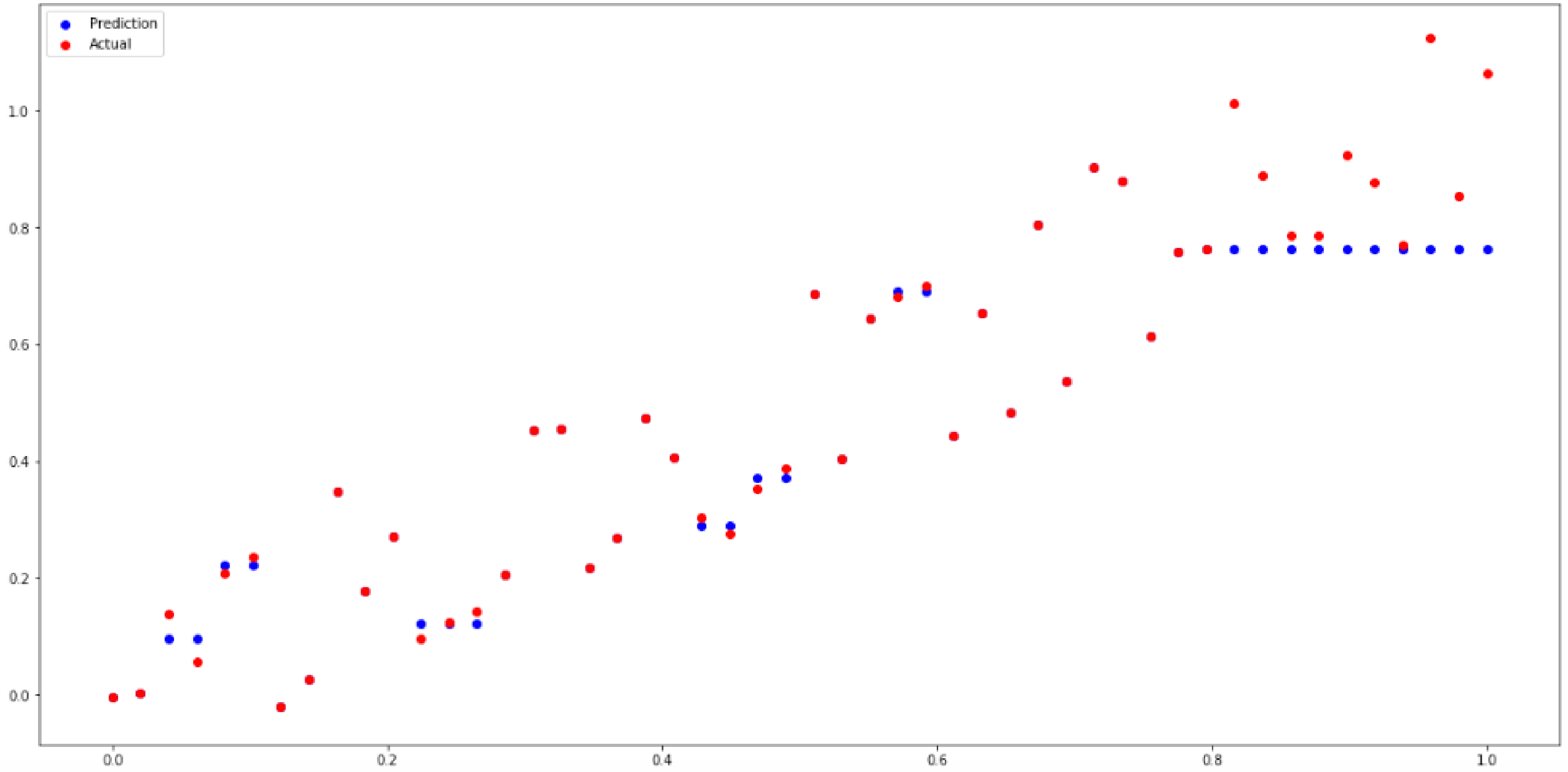


Decision Tree Regressor



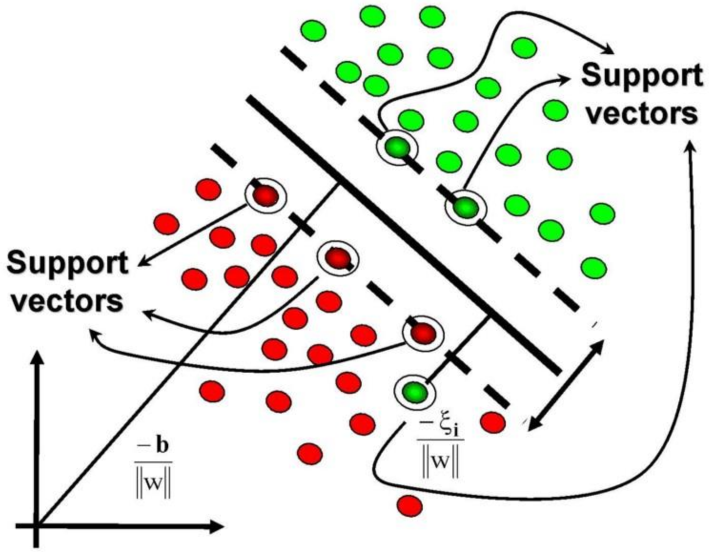


Decision Tree Regressor

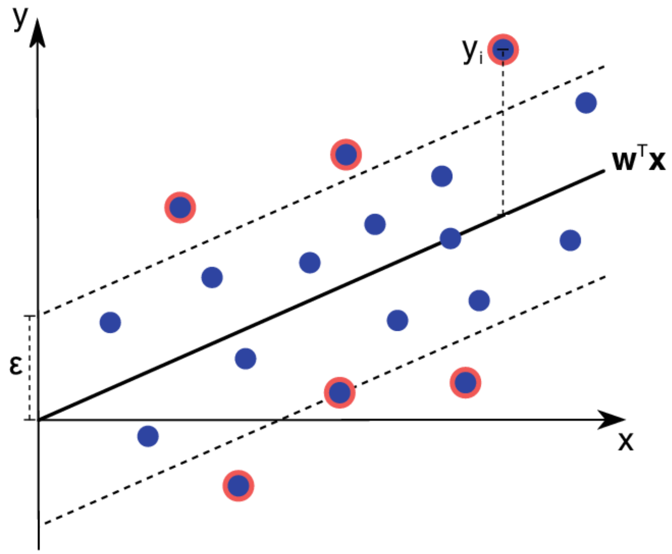


Support Vector Machine for Regression

SVM for classification



SVM for regression



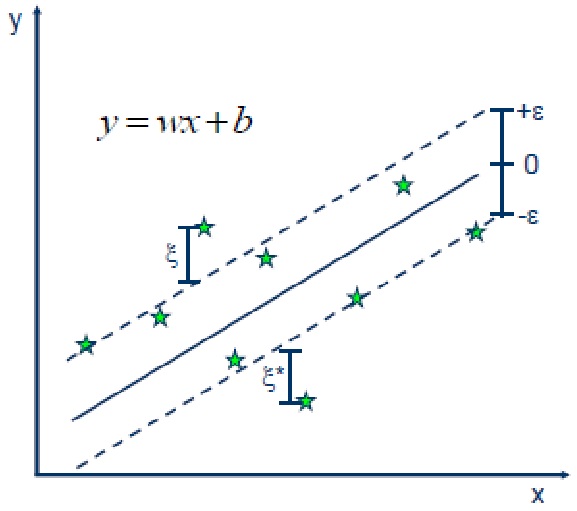
Support Vector Regressor (SVR)

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:

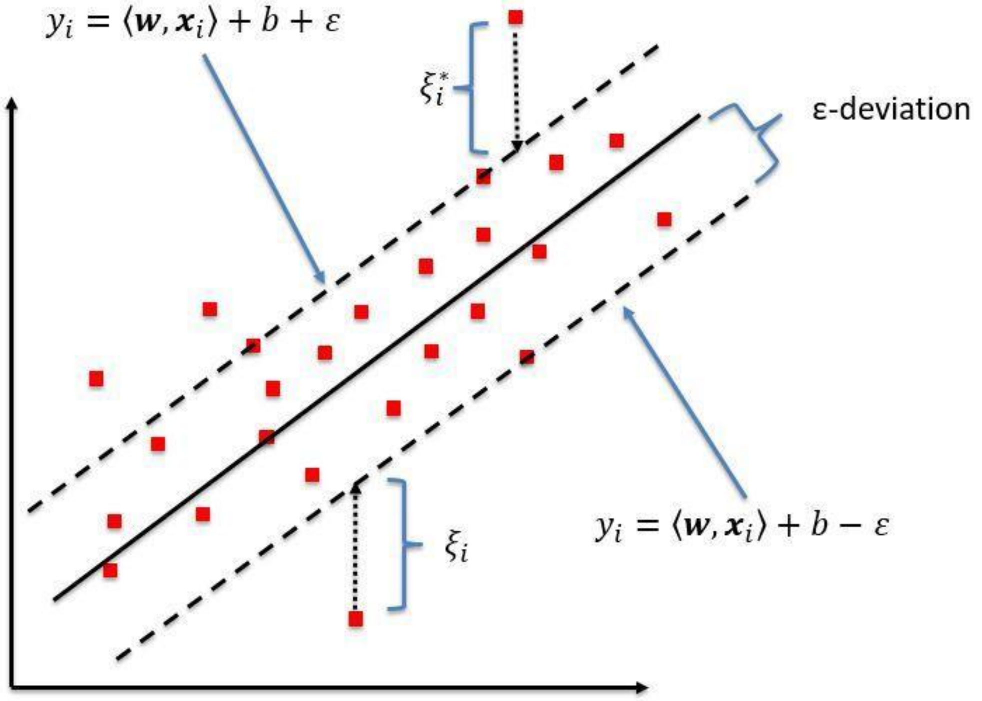
$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

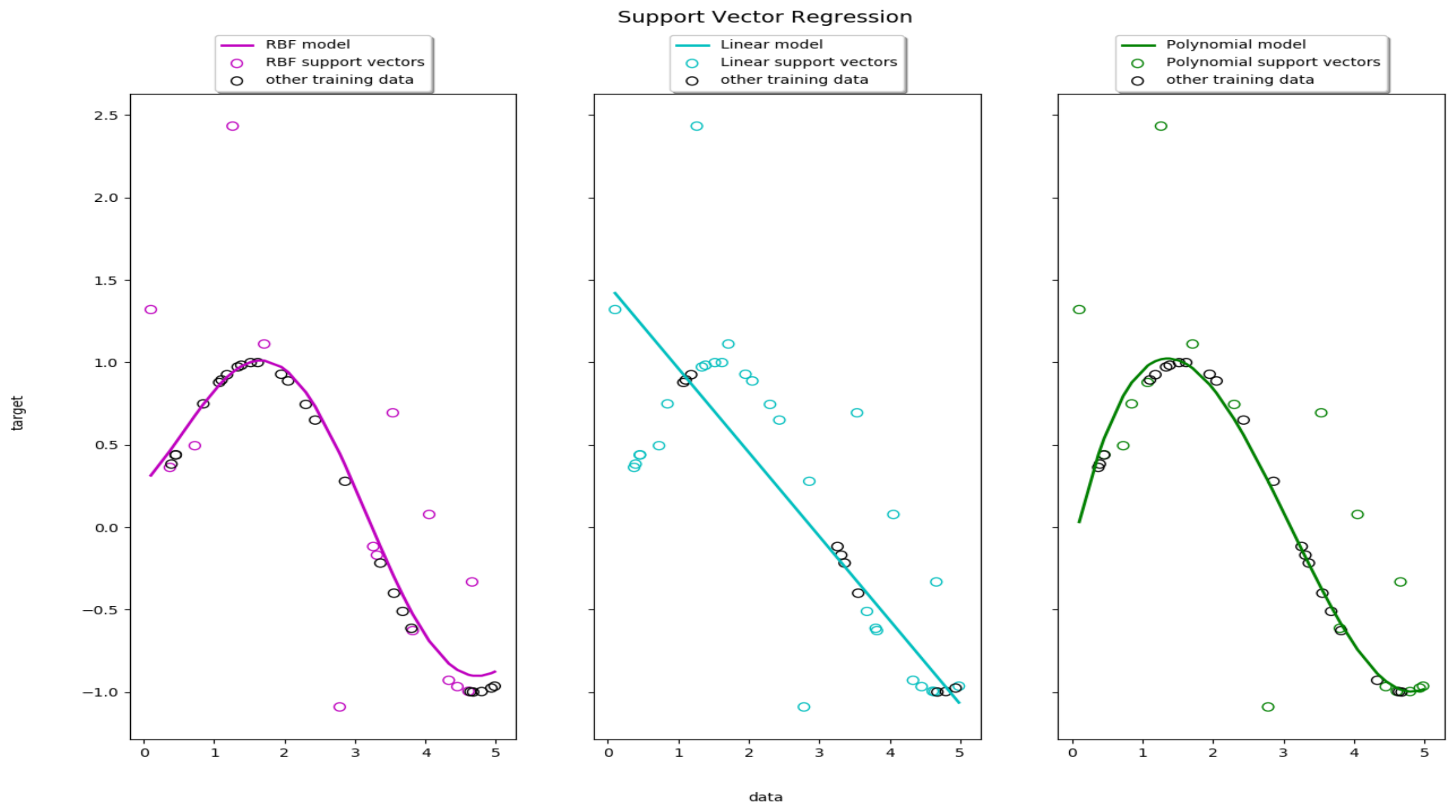
$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

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


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

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



Regression

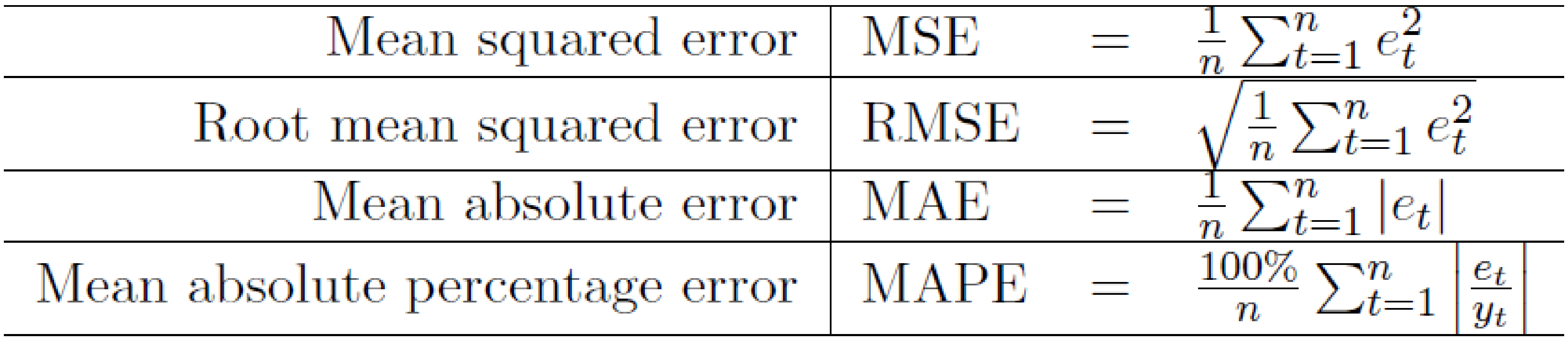
MODEL EVALUATION

(RECAP)

Model evaluation for regressors

- The metrics used for regression are totally different from those used classification.
- The metrics used for regression basically measure the amount of error in the predictions by comparing the predicted value/score (yhat) with the actual value of the lable (y).
- The Residuals (error) e = y – yhat
- In scikit-learn, the metrics for regression are found at https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics
- The r2_score function computes the coefficient of determination, usually denoted as R^2. The best possible r2_score (perfect) is 1.0.
- r2_score can be negative (because the model can be arbitrarily worse). A constant model, disregarding the input features, would get a R^2 score of 0.0.

Metrics for Regression



Mean squared error	MSE	$=$	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	RMSE	$=$	$\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	MAE	$=$	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	MAPE	$=$	$\frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

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