HIPERWIND
PhD Summer School

28-31 August 2023

PhD school on uncertainty quantification and reliability assessment of offshore wind turbines

Mild introduction to regression models and ML regression techniques

Nikolay Dimitrov, DTU Wind and Energy Systems

A model – mathematical definition

A mathematical model $g(\mathbf{x})$ is a function which provides mapping between two variable sets:

- The inputs, $\mathbf{x}$
- The outputs, $\mathbf{y}$

$$\mathbf{y} + \epsilon_y = g(\mathbf{x} + \epsilon_x, \boldsymbol{\theta}) + \epsilon_g$$

where $\boldsymbol{\theta}$ are the model parameters which modify the function $g(\mathbf{x})$

A model – mathematical definition

A mathematical model $g(\mathbf{x})$ is a function which provides mapping between two variable sets:

- The inputs, $\mathbf{x}$
- The outputs, $\mathbf{y}$

$$\mathbf{y} + \epsilon_y = g(\mathbf{x} + \epsilon_x, \boldsymbol{\theta}) + \epsilon_g$$

where $\boldsymbol{\theta}$ are the model parameters which modify the function $g(\mathbf{x})$

All models are wrong – but some of them are useful.

George P. Box

Data model types

- Deterministic model:

$$\mathbf{y} = g(\mathbf{x}, \boldsymbol{\theta}) + \epsilon$$

  - The function g is deterministic and provides a unique mapping between $\mathbf{x}$ and $\mathbf{y}$
  - The error term ϵ represents the combined model uncertainty and measurement uncertainty in $\mathbf{y}$
  - The inputs $\mathbf{x}$ can be random
  - Model training amounts to finding the values of $\boldsymbol{\theta}$ which minimize the error ϵ

- Probabilistic model:
  - The output is not unique: for a given set of inputs $\mathbf{x}$, the output is given in terms of a probability distribution
  - The random output can be due to either probabilistic parameters $\boldsymbol{\theta}$, or due to the function $g(\mathbf{x})$ being intrinsically probabilistic
  - Model training usually carried out by maximizing a likelihood function

Data model types

- Physics-based model: when both $g(\mathbf{x})$ and $\boldsymbol{\theta}$ are derived through relationships explained by physical laws (including numerical models)

- Data-driven model (a.k.a. black-box model): the function $g(\mathbf{x})$ uses abstract relationships tailored to fit complex shapes but not providing any physical explanation. The parameters $\boldsymbol{\theta}$ are determined from data

- Hybrid (greybox) model: $g(\mathbf{x})$ uses physics-based formulations, and $\boldsymbol{\theta}$ are adjusted with data

- (Bayesian) model updating: an initial version of a probabilistic model is established as a physics-based model or with data available a priori. The model parameters $\boldsymbol{\theta}$ are then updated with a new data set.

Model types

If the output y is:

- Discrete (e.g. category): → classification model
- Continuous: → regression model
- Continuous, correlated (e.g. time series) → sequence model

The concept of Machine (Statistical) Learning

Supervised learning

We **map the relation** between a set of input (descriptive) variables $\mathbf{x}$, and an output (dependent) variable y:

$$y = g(\mathbf{x})$$

learning from **observations of both $\mathbf{x}$ and y**

Unsupervised learning

We don't have a dependent variable:

$$y = ?$$

We could look for **patterns and associations between input variables $\mathbf{x}$** (data mining)

Supervised learning

Classification problem:

- y is **discrete** (belongs to a **category**)
- Mapping between $\mathbf{x}$ and y is not unique

Regression problem:

- y is **continuous** (e.g. $y \in \mathbb{R}$)
- Mapping between $\mathbf{x}$ and y is continuous

Model training:

For **simultaneous** observations of pairs of variables, $\hat{\mathbf{x}}$ and $\hat{y}$, find model parameters $\boldsymbol{\theta}$ which **minimize the difference** between observations $\hat{y}$ and model predictions $y = g(\hat{\mathbf{x}})$

The "black box" aspect:

In such a data-driven model, the function $g(\mathbf{x})$ uses abstract relationships tailored to represent complex shapes but not providing any physical explanation.

Regression models

- Typical approaches:
  - Polynomial fits (e.g., linear, quadratic…)
  - Expansions based on decomposition (Taylor series, Polynomial Chaos, Karhunen-Loeve, Principle Component Analysis)
  - Interpolation
  - Feedforward Neural Networks
  - Kernel-based methods
  - Gaussian model (a.k.a. Kriging)

Linear models

- General linear model definition:

$$E[Y] = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

(note that x_i are not necessarily 1st-order variables)

- Question: if all x_i are Gaussian, what is the distribution of Y?

- If Y is Gaussian, its expected value is a linear combination of the expected values of the input variables

- Generalized linear models: replace $E[Y]$ with $g(E[Y])$, where $g(.)$ is a smooth continuous function transforming the response to Gaussian. Used for handling problems where the response is non-Gaussian.

Building a regression model

- Let us consider a model represented by a system of linear equations:

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y}$$

where $\mathbf{y}$ is a vector or realizations of the dependent variable, $\mathbf{X}$ is a matrix of known factors, and $\boldsymbol{\beta}$ is a vector of coefficients to be determined.

- We have a set of m realizations, $\hat{\mathbf{y}}$, and want to find an optimal set of coefficients $\widehat{\boldsymbol{\beta}}$ which minimize the squared sum of the residuals:

$$\Sigma\varepsilon(\beta) = \sum_{i=1}^{m} \left| \hat{y}_i - \sum_{j=1}^{n} X_{ij}\beta_j \right|^2 = \left\| \hat{\mathbf{y}} - \mathbf{X}\widehat{\boldsymbol{\beta}} \right\|^2$$

- Minimizing the squared sum of the residuals and making sure the mean of the residuals is zero means that the result has the lowest possible variance of the model error (unexplained variance)

Building a regression model

- Multiplying the linear system with $\mathbf{X}^T$:

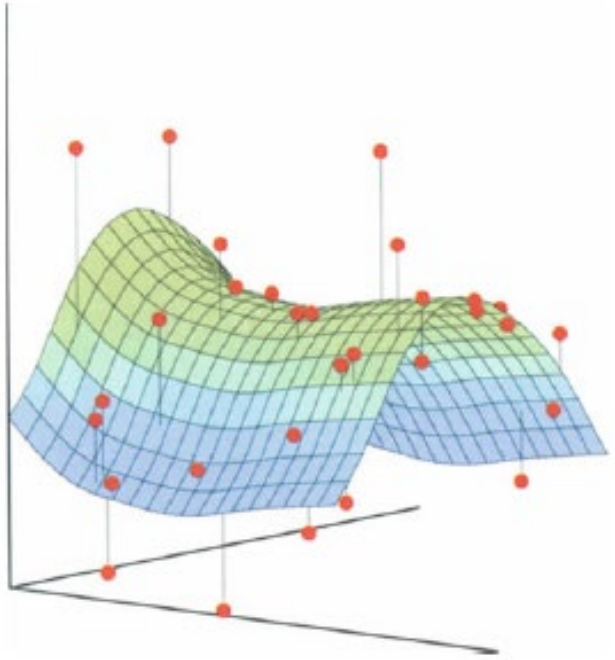
$$\left(\mathbf{X}^T\mathbf{X}\right)\hat{\boldsymbol{\beta}} = \mathbf{X}^T\hat{\mathbf{y}}$$

- Matrix $\left(\mathbf{X}^T\mathbf{X}\right)$ is square and invertible:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\hat{\mathbf{y}}$$

- The matrix $\mathbf{X}$ is called the design matrix. It contains expressions with all the input factors. For example, for a model of the type $y = x^2 + x + z + 1$ the design matrix will have the following columns:

$$X = \begin{bmatrix} 1 & x_1 & z_1 & x_1^2 \\ 1 & x_2 & z_2 & x_2^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & z_m & x_m^2 \end{bmatrix}$$



Hastie, The elements of statistical learning, 2009, Springer

Regression model error

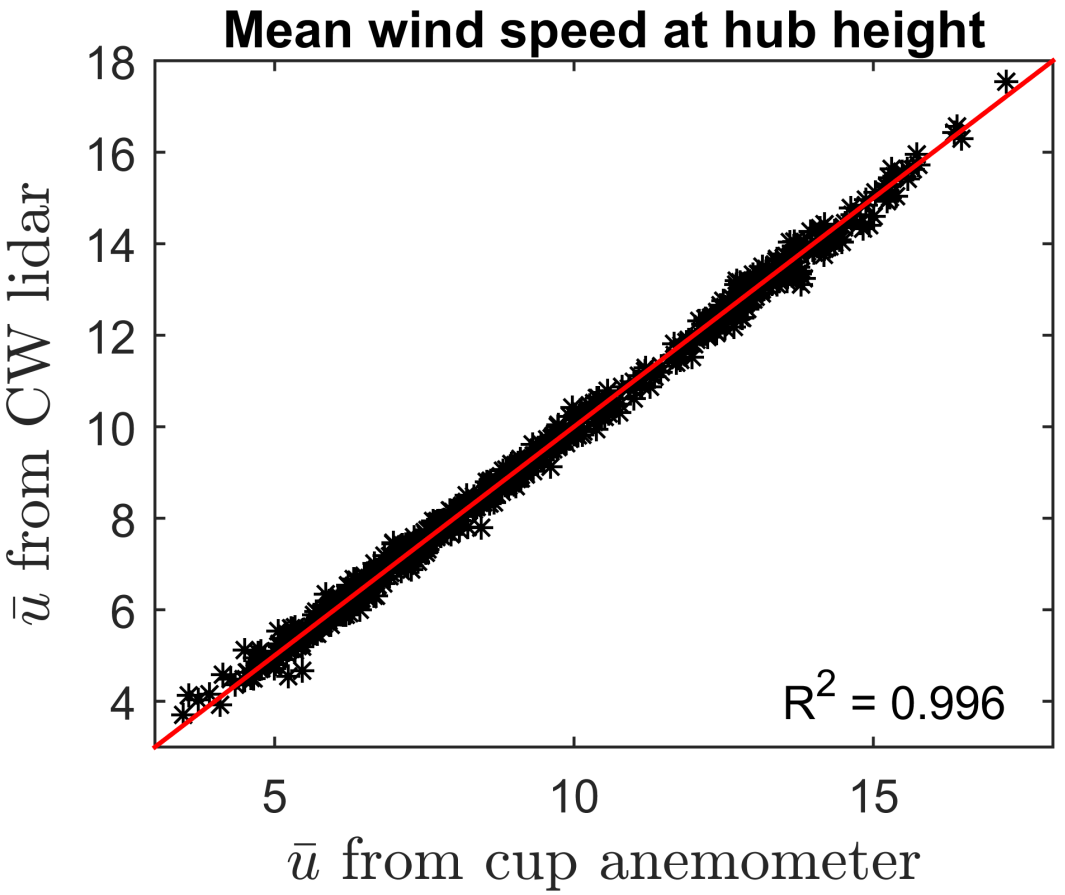
Let us recall that:

- We are aiming at obtaining a mapping of the type

$$y = g(\mathbf{x}, \boldsymbol{\theta}) + \epsilon$$

- y is a variable with certain statistics as mean and variance.

- What can the statistics of the error (residual) term ϵ tell us about our model?

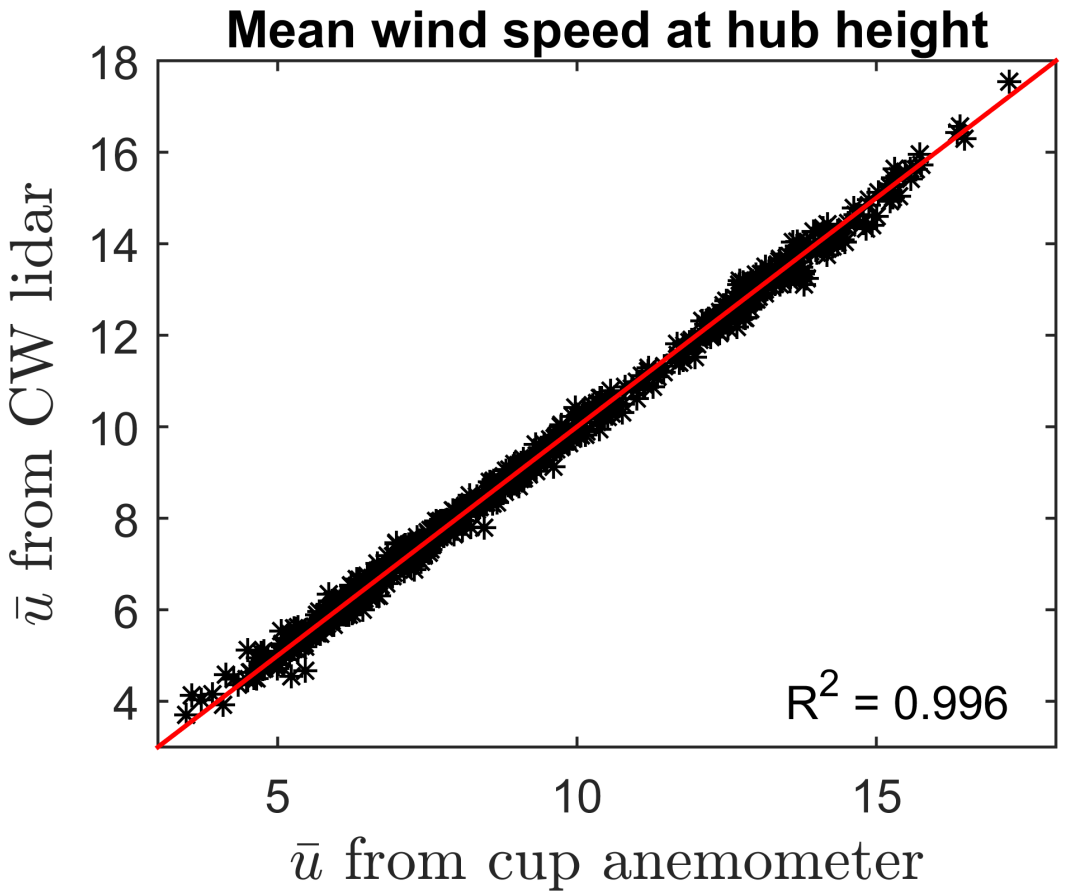


Regression model error

What can the error statistics tell us about our model?

- The mean of ϵ is the model **bias**
- The variance relation is:

$$Var[y] = Var[g(\mathbf{x}, \boldsymbol{\theta})] + Var[\epsilon]$$



Regression model error

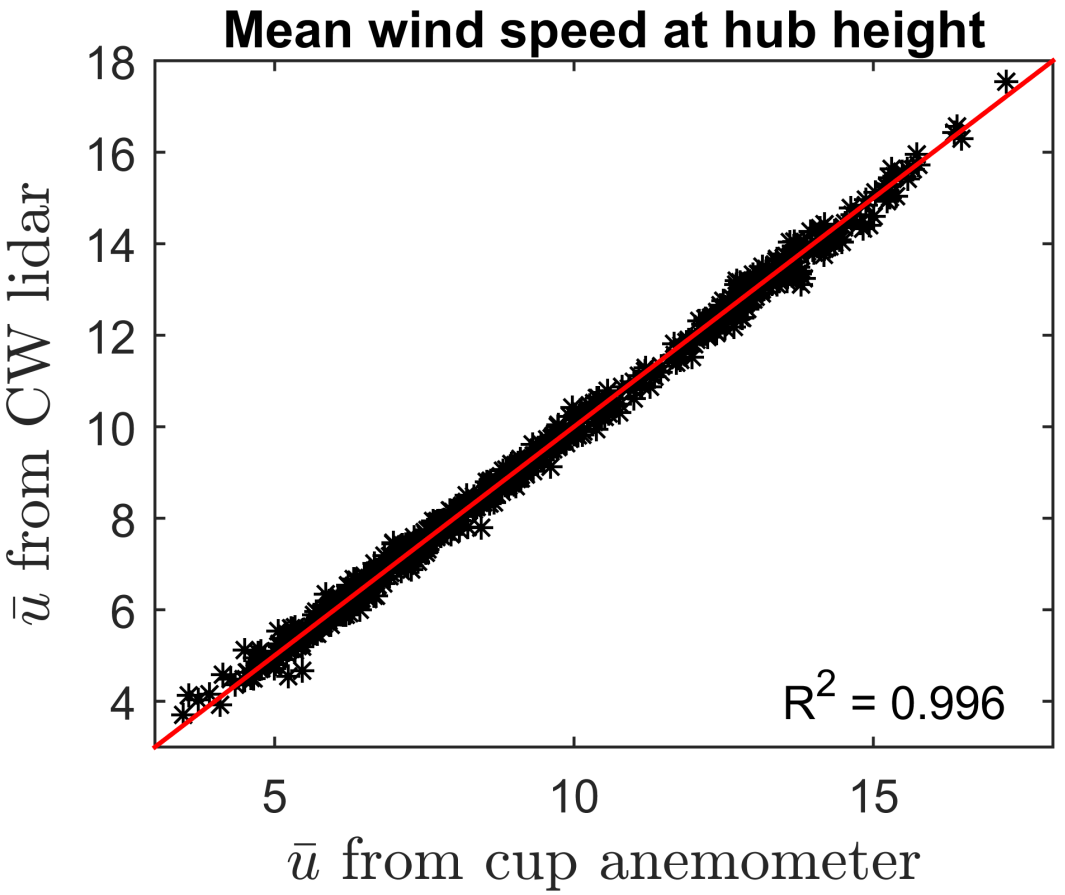
What can the error statistics tell us about our model?

- The mean of ϵ is the model **bias**
- The variance relation is:

$$Var[y] = Var[g(\mathbf{x}, \boldsymbol{\theta})] + Var[\epsilon]$$

"Explained variance"

"Unexplained variance"



Regression model error

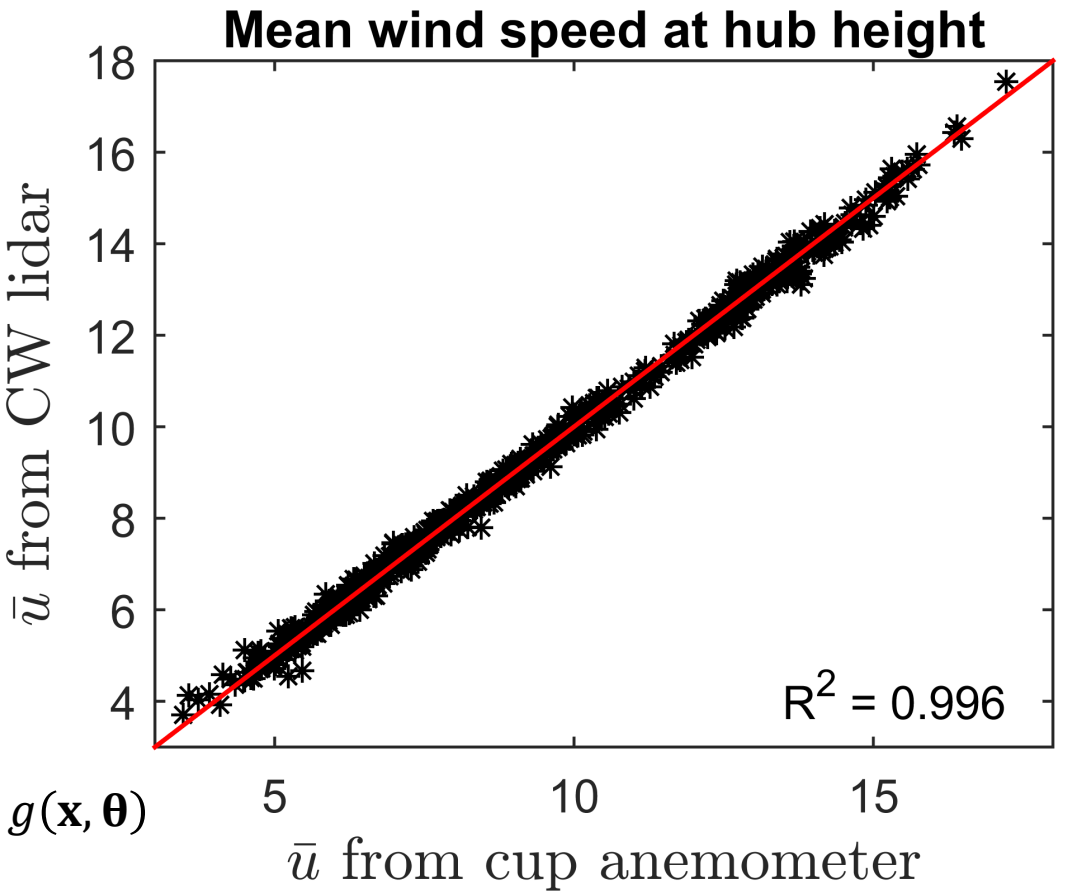
What can the error statistics tell us about our model?

- The mean of ϵ is the model **bias**
- The variance relation is:

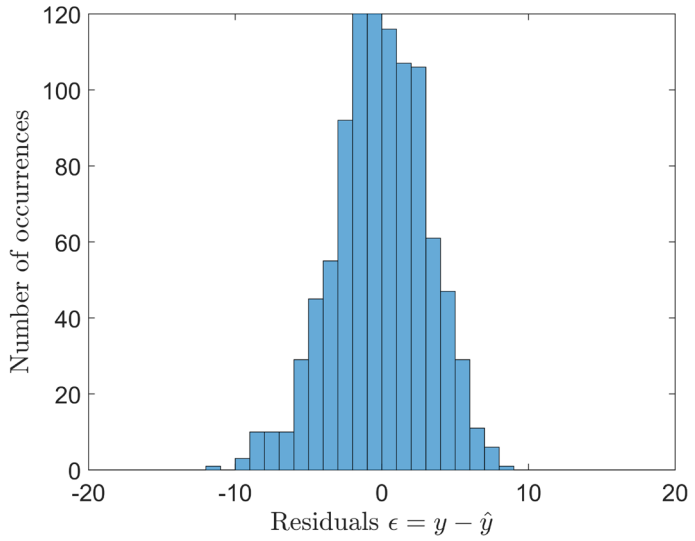
$$Var[y] = Var[g(\mathbf{x}, \boldsymbol{\theta})] + Var[\epsilon]$$

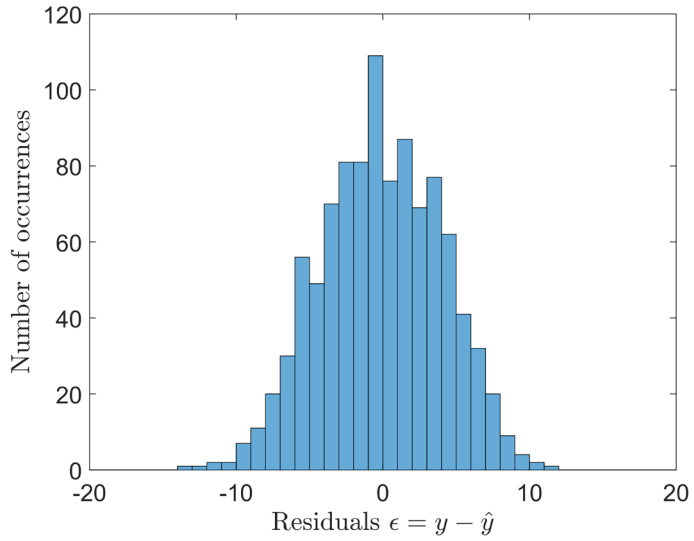
"Explained variance" "Unexplained variance"

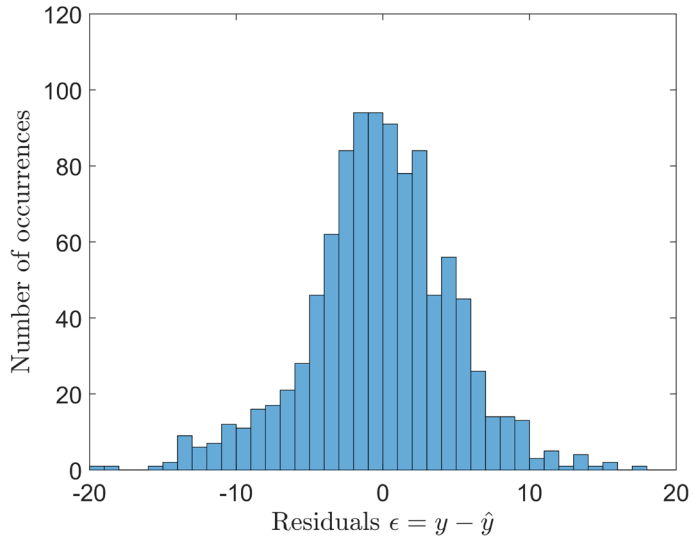
- The r-square (R^2 or r^2):
  - Approximately equals the square of the correlation between y and $g(\mathbf{x}, \boldsymbol{\theta})$
  - Equals the ratio of explained variance to total variance!
  - Is also called "coefficient of determination"



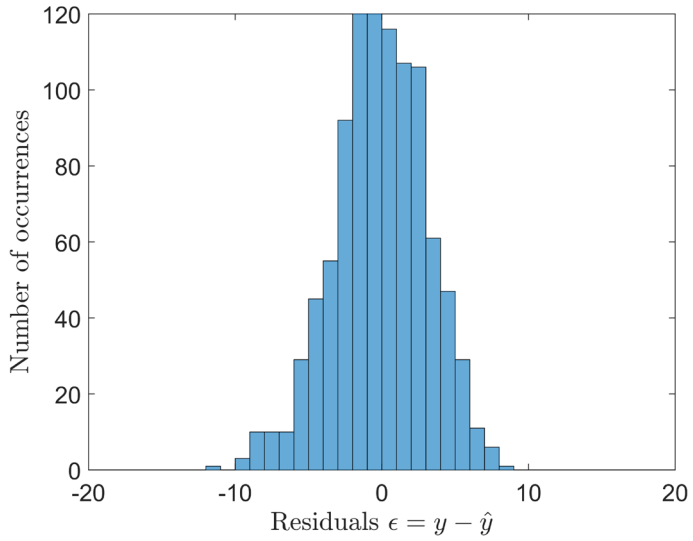
Residual plots – what can we see from them?

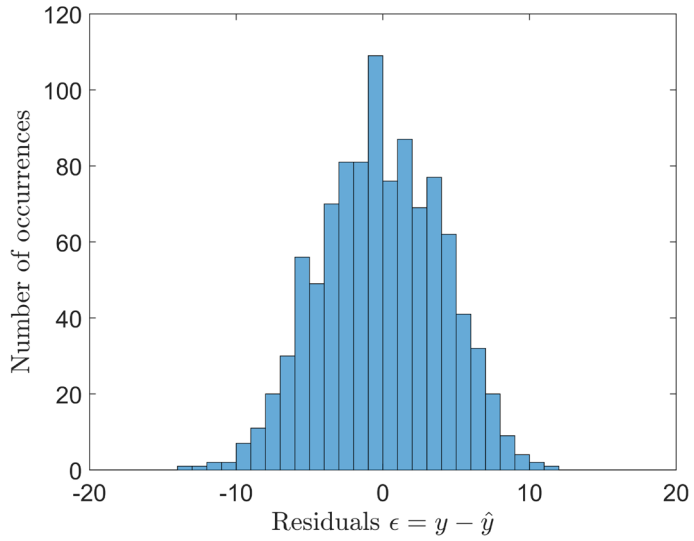


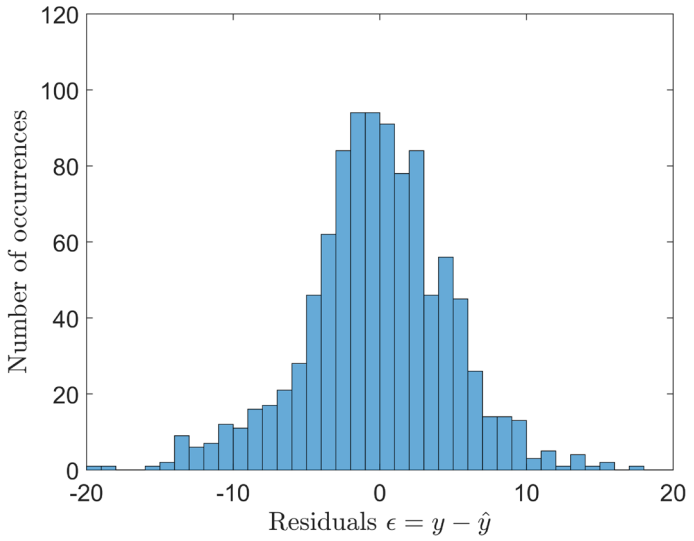


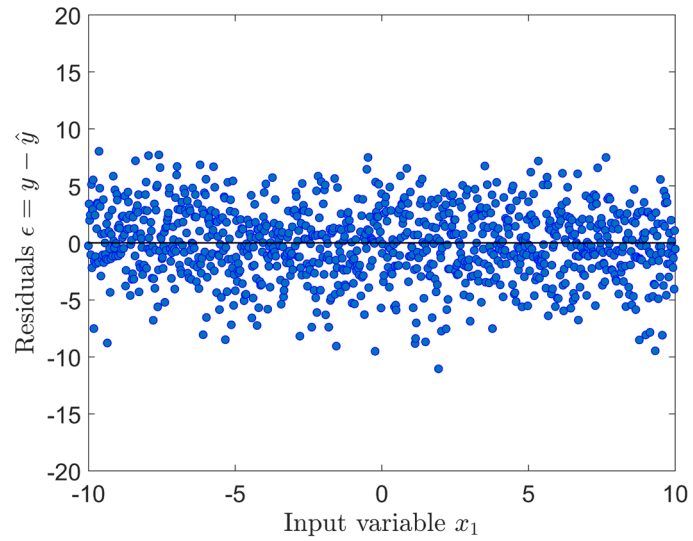


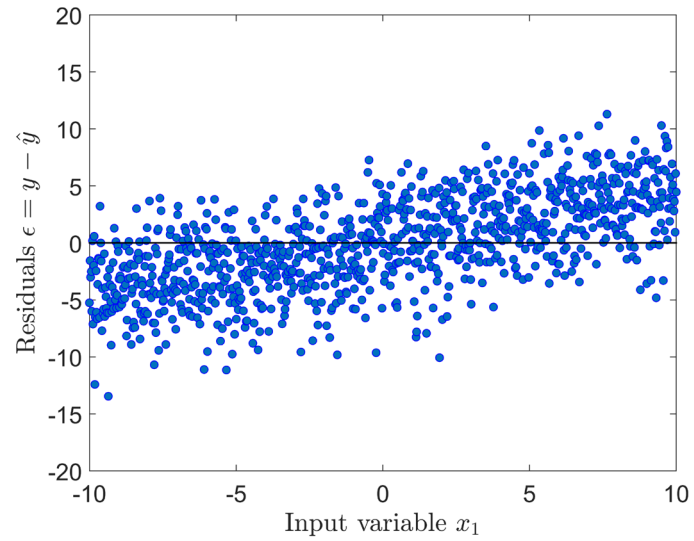
Residual plots – what can we see from them?

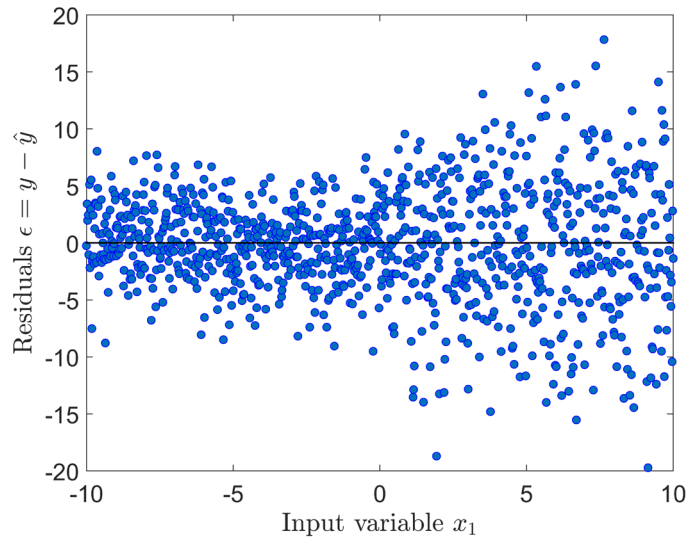




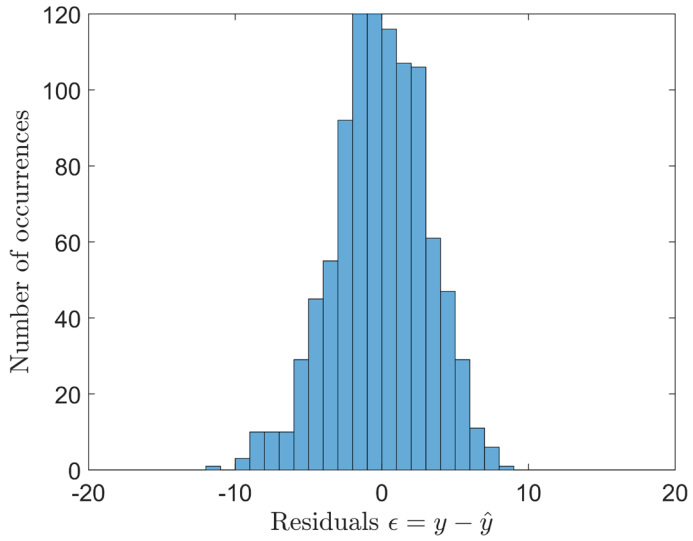


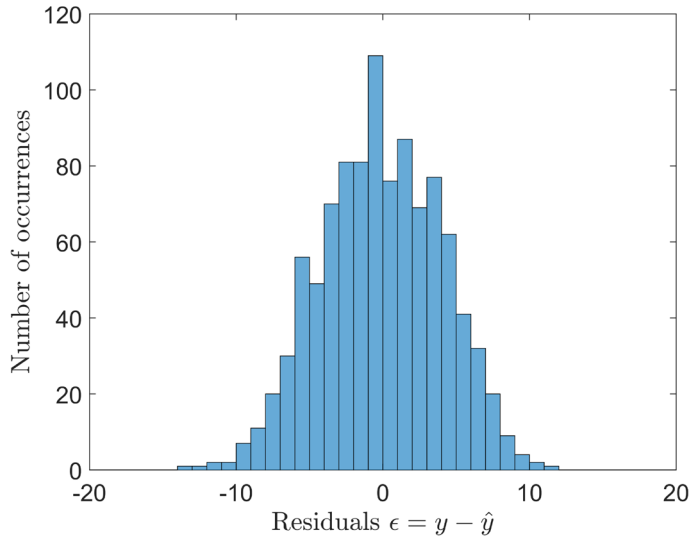


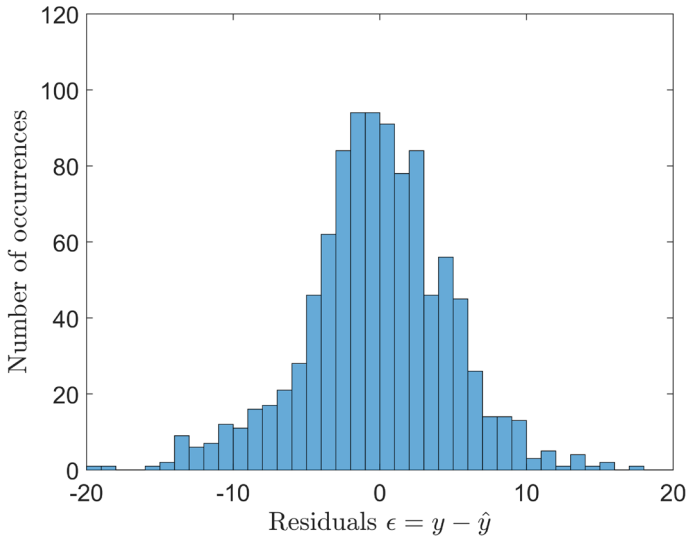


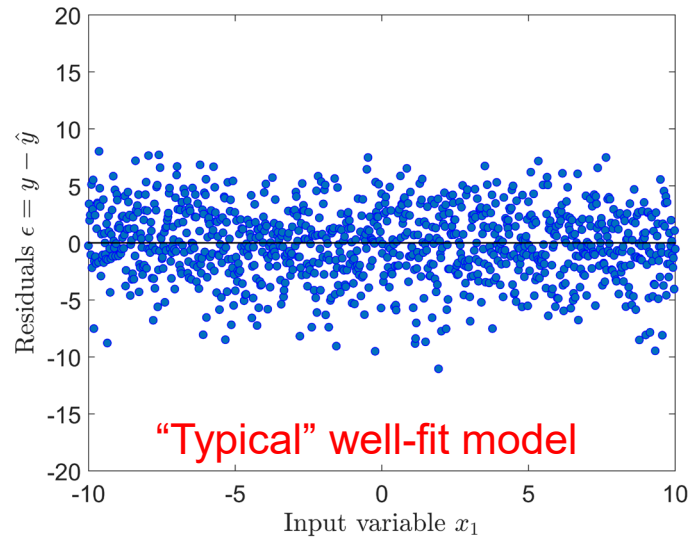


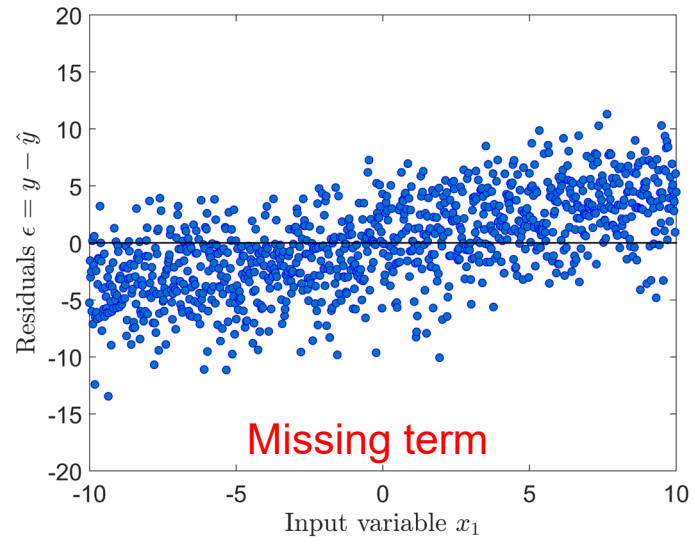
Residual plots – what can we see from them?

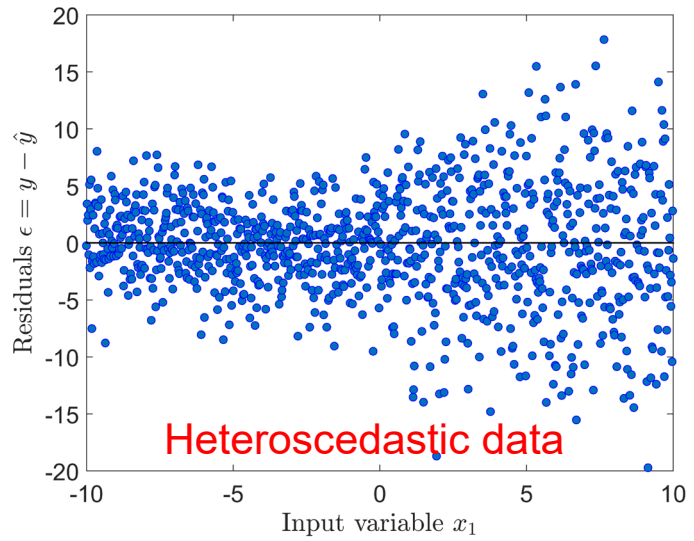












Regression model performance metrics

Metric	Formula	Description
R-squared (r^2)	$r^2 = 1 - \left(\frac{\sum_i (y_i - g(x_i))^2}{\sum_i (y_i - \bar{y})^2} \right)$	Defines the fraction of variance in the data which is explained by the model. Since it is normalized, can be used across data sets.
Root mean squared error (RMSE) and mean squared error (MSE)	$RMSE = \sqrt{MSE} = \sqrt{E\left((y - g(x))^2 \right)}$ $= \sqrt{\frac{\sum_i (y_i - g(x_i))^2}{T}}$	Quantifies the variance which is NOT explained by the model.
Mean absolute error (MAE)	$MAE = \sum_{i=1}^{N} \frac{\lvert y_i - g(x_i) \rvert}{N}$	Indicator for both model bias and uncertainty. More easily interpretable but not directly related to error variance

Common ML methods

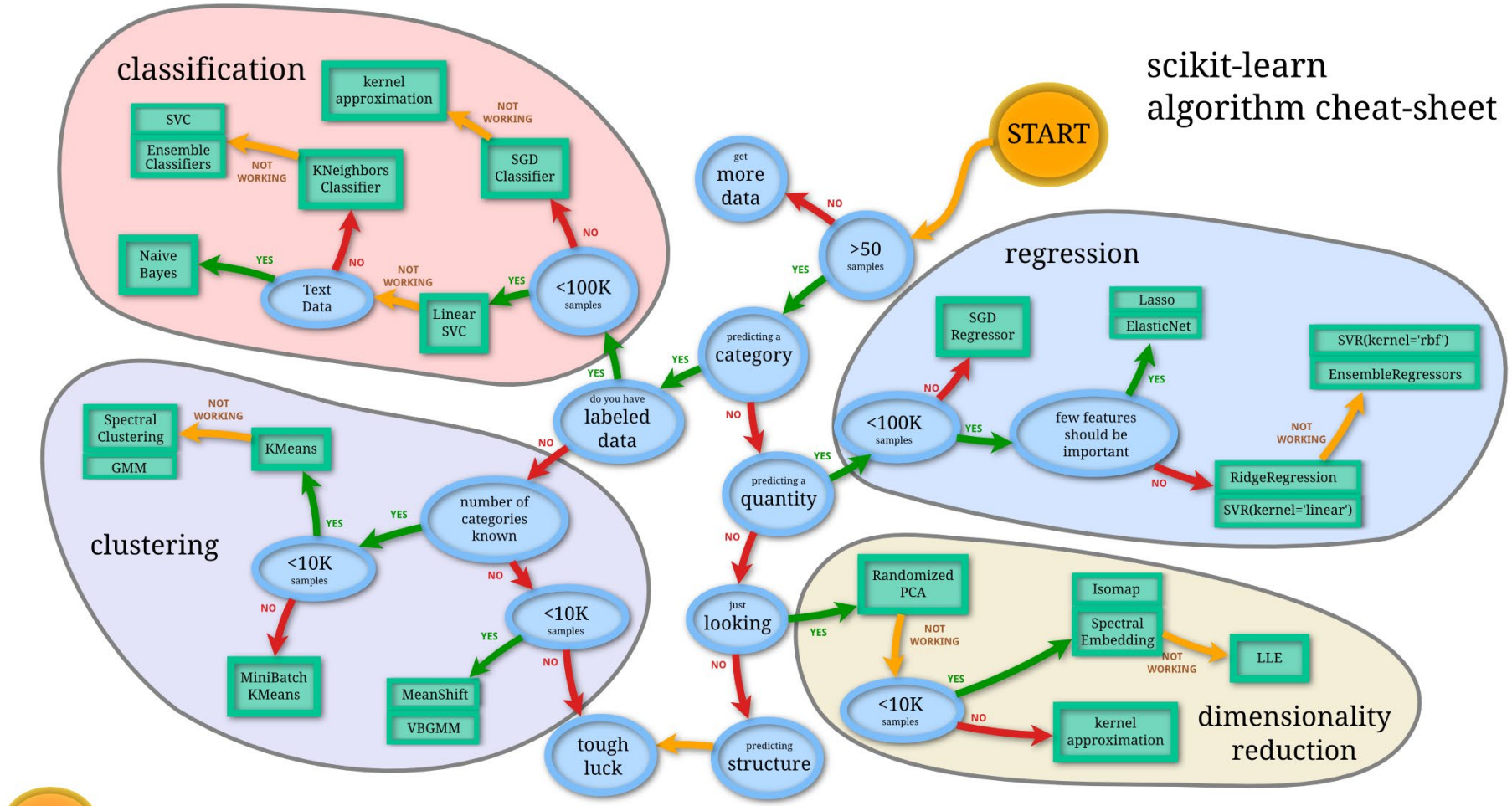
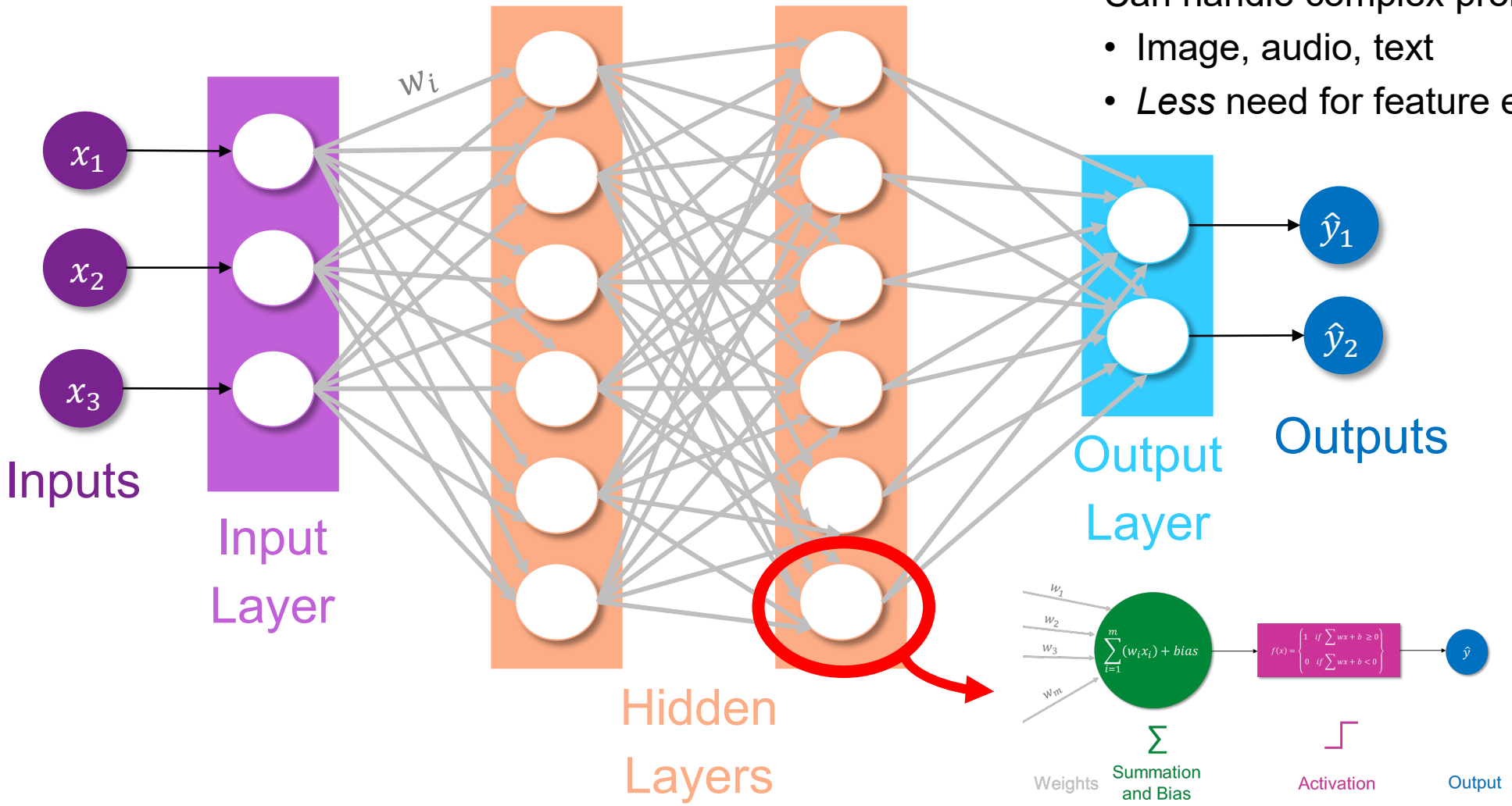


Image source: Scikit-learn cheat sheet: choosing the right estimator

Neural networks

- Requires large dataset to work well
- *Very* parametric → requires experience
- Can handle complex problems
  - Image, audio, text
  - *Less* need for feature engineering



ML model training

- Recalling:

Supervised learning

We **map the relation** between a set of input (descriptive) variables $\mathbf{x}$, and output (dependent) variable(s) $\mathbf{y}$:

$$\mathbf{y} = g(\mathbf{x})$$

learning from **observations of both $\mathbf{x}$ and $\mathbf{y}$**

Model training (supervised learning):

For **simultaneous** observations of pairs of variables, $\hat{\mathbf{x}}$ and $\hat{y}$, find model parameters $\boldsymbol{\theta}$ which **minimize the difference** between observations $\hat{y}$ and model predictions $y = g(\hat{\mathbf{x}})$

- The least-squares approach (universally used):

$$\boldsymbol{\theta}_{opt} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\hat{y}_i - g(\hat{\mathbf{x}}_i, \boldsymbol{\theta})\right)^2$$

The likelihood method

The likelihood function

- For a given set of model parameters $\boldsymbol{\theta}$, the likelihood $\mathcal{L}$ represents a measure of the possibility that a set of observations $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3, \ldots \hat{\mathbf{x}}_n)$ belongs to a population of function values $g(\mathbf{x}, \boldsymbol{\theta})$.

$$\mathcal{L}(\boldsymbol{\theta}|\hat{\mathbf{x}}) \propto P(\hat{\mathbf{x}}|\boldsymbol{\theta})$$

- In the case when $\boldsymbol{\theta}$ are parameters of a statistical distribution, the likelihood is given by

$$\mathcal{L}(\boldsymbol{\theta}|\hat{\mathbf{x}}) = \prod_{i=1}^{n} g_{\mathbf{x}}(\boldsymbol{\theta}, \hat{\mathbf{x}}_i)$$

- A log-transformation is usually applied:

$$\log \mathcal{L}(\boldsymbol{\theta}|\hat{\mathbf{x}}) = \sum_{i=1}^{n} \log\{g_X(\boldsymbol{\theta}, \hat{\mathbf{x}}_i)\}$$

- For model fitting, the likelihood is typically defined in terms of the model error ϵ, i.e., the highest likelihood corresponds to values of θ for which the error follows the intended distribution (e.g. zero mean and smallest variance).
- A maximum-likelihood estimate (MLE) is obtained by minimizing the negative log-likelihood
- For a linear model with normally distributed error, the MLE should be equivalent to the least-squares estimate

ML model training

- In recent ML terminology, the function J which is minimized during training is called the **cost function**. For example, for the least-squares approach:

$$J_{LSQ} = \sum_{i=1}^{n} \left(\hat{y}_i - g(\hat{\mathbf{x}}_i, \boldsymbol{\theta})\right)^2$$

- The contribution to the cost function from a single data pair $(\hat{\mathbf{x}}_i, \hat{y}_i)$ is called a **loss function**:

$$L_{LSQ} = \left(\hat{y}_i - g(\hat{\mathbf{x}}_i, \boldsymbol{\theta})\right)^2$$

- Another example of a widely used loss function is the cross-entropy (useful for binary classifiers):

$$L_{CE} = -\hat{y}_i \log\left(g(\hat{\mathbf{x}}_i, \boldsymbol{\theta})\right) - (1 - \hat{y}_i) \log\left(1 - g(\hat{\mathbf{x}}_i, \boldsymbol{\theta})\right)$$

Note that in the cross-entropy loss function, y and $g(\mathbf{x}, \boldsymbol{\theta})$ are given in terms of the probability of the output belonging to a certain category

The gradient descent algorithm

- The gradient descent algorithm is an optimization technique which iteratively searches for the model parameter values which minimize the cost function.

- The general formula for the parameter values at iteration step $(i + 1)$ is:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_{\mathbf{i}} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_i)$$

where

- $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_i)$ is the gradient of the cost function with respect to the model parameters $\boldsymbol{\theta}$, evaluated at $\boldsymbol{\theta}_i$
- α is the learning rate

- The algorithm is broadly used in ML model training, with multiple variations and additions

Data split

- Attempting to train complex models with limited amounts of data could lead to overfitting
- Leaving some data aside for additional testing can help avoiding overfitting:
  - The model parameters are only trained using the training set
  - The model performance (predictions) on the training set are regularly checked against the performance with the validation set
  - An overfitting model will start showing increasingly bigger difference in performance between training and validation sets
  - Model training can be stopped when overfitting kicks in – - or mitigation measures such as regularization could be introduced
- The test set is only used for final evaluation of the trained model performance

K-fold cross validation

- A method to ensure more uniform model performance throughout the application domain
- Data are split in k folds of (approximately) equal size.
- K different models are trained, where in each training process one fold is taken as a validation set, the remaining data are used for training
- The final model is normally deployed as running all k models as an ensemble.

Example k-fold cross-validation where k = 5:

Regularization

- Regularization techniques seek to reduce model weights (parameters) in terms of the number of active parameters and in terms of parameter values.

- This typically improves model generalizability and reduces overfitting

- A standard approach is to introduce a penalty function $R(\boldsymbol{\theta})$ to the cost function:

$$\tilde{J}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) + \lambda R(\boldsymbol{\theta})$$

- LASSO regularization (least absolute shrinkage and selection operator): limits the sum of all model coefficients. For a linear regression $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$:

$$\tilde{J} = \left(\frac{1}{N} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right)$$

- Neural network regularization:
  - L_1, L_2 regularization with a penalty function based on the L_1 or L_2 norms of the ANN weights
  - Dropout (at each iteration, randomly change the weights of a fraction of the neurons to 0)
  - Early stopping

- Other methods:
  - Bootstrapping / bagging (ensemble models)
  - Introducing noise – or adversarial examples

Hyperparameter tuning

- Various model settings outside the actual parameters can affect the training process as well as the resulting model performance.

- Typical list of hyperparameters include:
  - Model architecture (e.g. number of layers in a neural network, number of trees in a random forest model, kernel functions…)
  - Neural networks in particular have many hyperparameters that can be tuned:
    - Number and type of layers
    - Activation functions
    - Learning rate
    - Regularization (L2, dropout)
    - Weight initialization
    - Batch size (data can be split in minibatches = stochastic gradient descent algorithm)

- Systematic hyperparameter tuning can be done by e.g. a grid search. There are also some automatic model tuning algorithms

Additional slides

The concept of Machine (Statistical) learning

- Recalling from earlier:

Data modelling: the process of identifying and evaluating relationships and patterns in a dataset, through mapping mathematical representations.

Machine learning: the specific task of establishing the mathematical representation

- What does “learning” mean when applied to machines?
  - Algorithms that perform a specific task
  - “Learning” (training) = to perform the task better based on observations

- There is lots of hype regarding ML (machine learning) recently. But is ML a new concept?

- What is the connection between ML and statistics & probability?

The concept of Machine (Statistical) Learning

- In this course, we have a short introduction to the ML concept, how it connects to statistics and probability, and we'll see some examples from Wind Energy

- Some good books with more information:

  – Tibshirani et al. *Introduction to Statistical Learning – with applications in R*, Springer New York, 2013 (available online through DTU Library login)

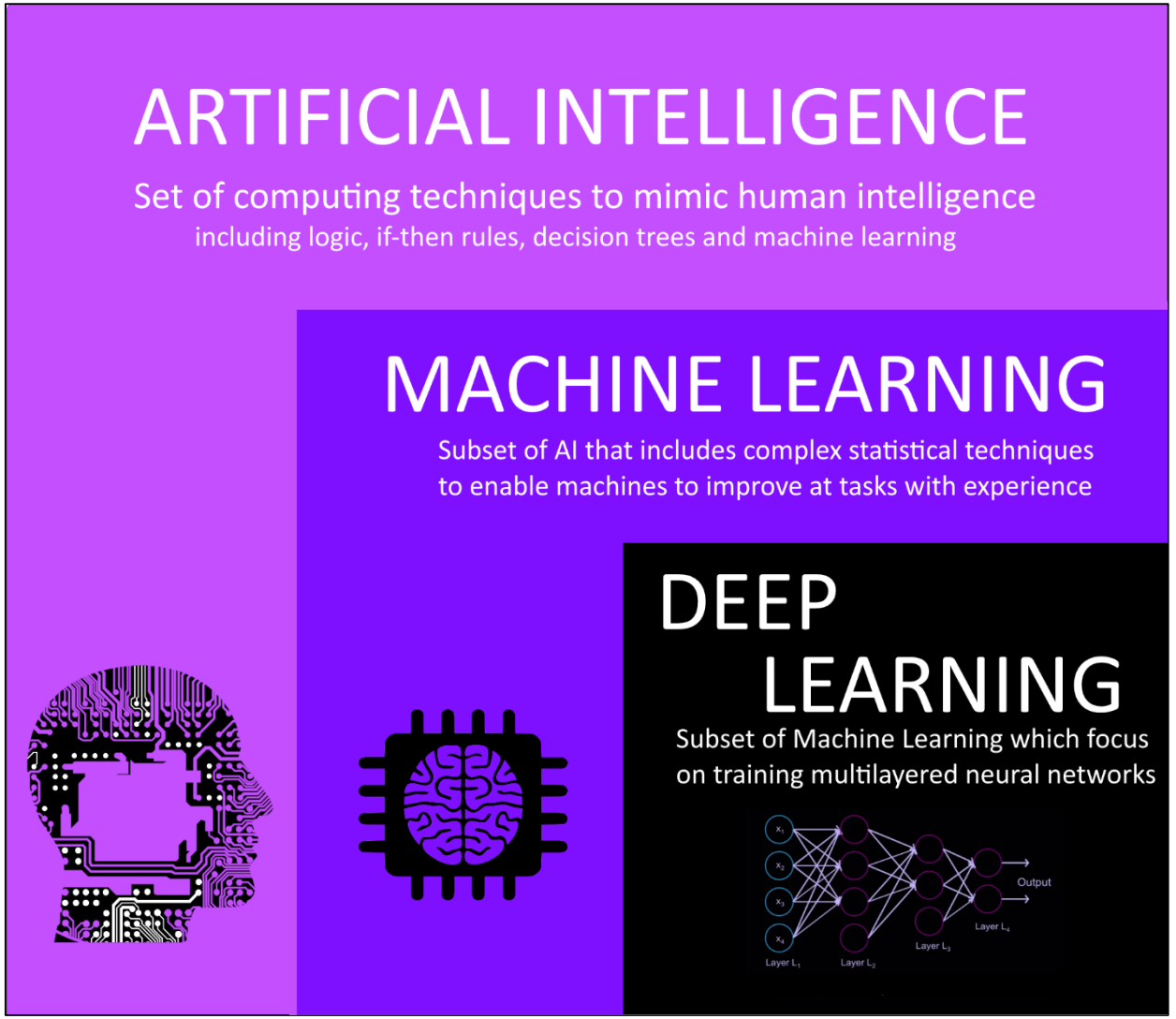
  – Murphy, K. *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012 (you would need to purchase it – ca. 500-700kr.)

  – Goodfellow et al. *Deep learning*, MIT Press, 2016. Freely available online (www.deeplearningbook.org).

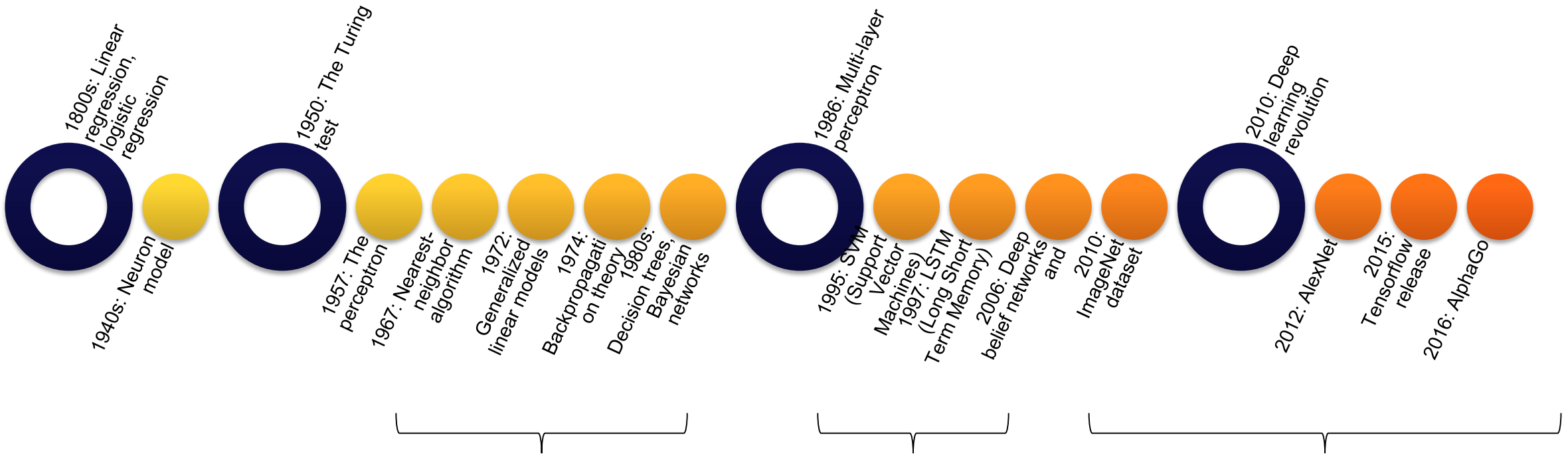
  – Bishop, C. M., *Pattern recognition and Machine learning*, Springer, 2016 (available online through DTU Library login)

  – Ding, Yu., Data Science for Wind Energy

Difference between ML, DL and AI



ML timeline


1800s: Linear regression, logistic regression
1940s: Neuron model
1950: The Turing test
1957: The perceptron
1967: Nearest-neighbor algorithm
1972: Generalized linear models
1974: Backpropagation theory
1980s: Decision trees, Bayesian networks
1986: Multi-layer perceptron
1995: SVM (Support Vector Machines)
1997: LSTM (Long Short Term Memory)
2006: Deep belief networks and
2010: ImageNet dataset
2010: Deep learning revolution
2012: AlexNet
2015: Tensorflow release
2016: AlphaGo
First AI winter
Second AI winter
Deep learning revolution

The concept of Machine (Statistical) Learning

Supervised learning

We **map the relation** between a set of input (descriptive) variables $\mathbf{x}$, and an output (dependent) variable y:

$$y = g(\mathbf{x})$$

learning from **observations of both $\mathbf{x}$ and y**

Unsupervised learning

We don't have a dependent variable:

$$y = ?$$

We could look for **patterns and associations between input variables $\mathbf{x}$** (data mining)

Supervised learning

Classification problem:

- y is **discrete** (belongs to a **category**)
- Mapping between $\mathbf{x}$ and y is not unique

Regression problem:

- y is **continuous** (e.g. $y \in \mathbb{R}$)
- Mapping between $\mathbf{x}$ and y is continuous

Model training:

For **simultaneous** observations of pairs of variables, $\hat{\mathbf{x}}$ and $\hat{y}$, find model parameters $\boldsymbol{\theta}$ which **minimize the difference** between observations $\hat{y}$ and model predictions $y = g(\hat{\mathbf{x}})$

The "black box" aspect:

In such a data-driven model, the function $g(\mathbf{x})$ uses abstract relationships tailored to represent complex shapes but not providing any physical explanation.

Common ML methods

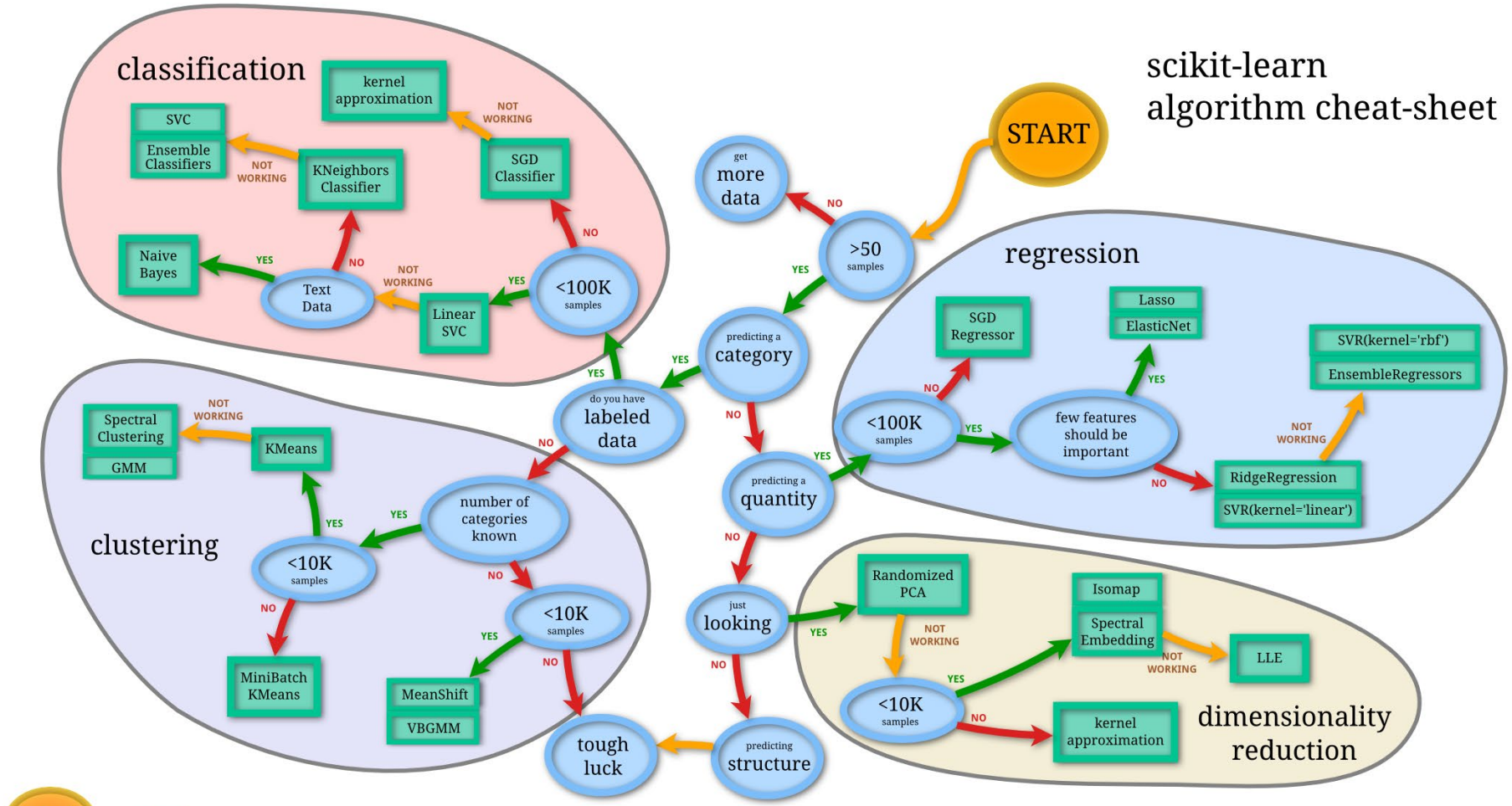


Image source: Scikit-learn cheat sheet: choosing the right estimator

Classification

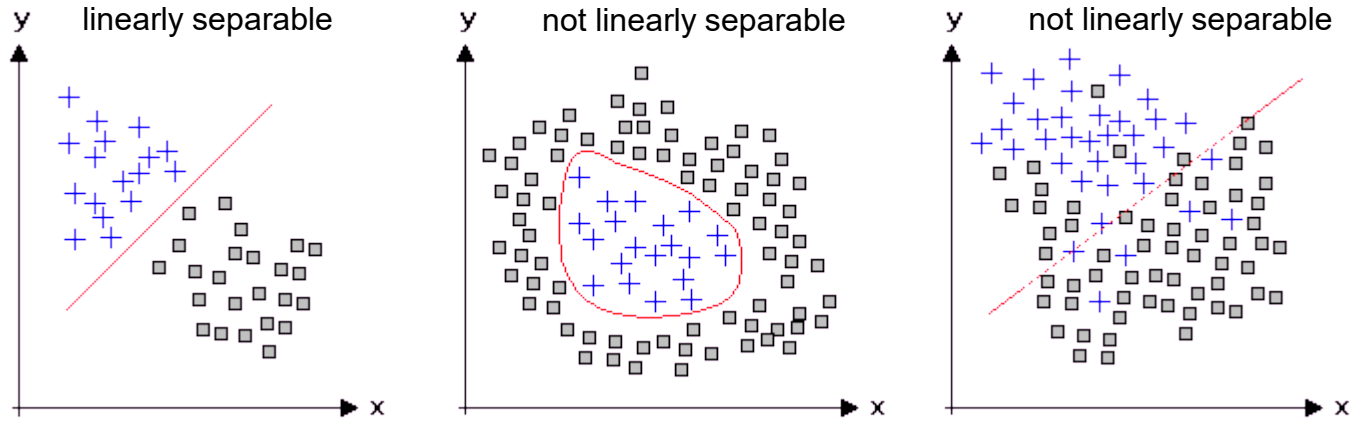
- A **supervised** learning problem where the output variable is a **category** (or categorical class labels)

$$h_\theta(x) = P(y = i|x;\theta) \qquad (i = 0,1,2,3,\dots)$$

- h_θ is the classifier of input x, which is parameterised by θ, to predict the output classes $(y = i = 0,1,2,3,\dots)$

- 2 main types of classification problems
  - Binary classification $(i = 0,1)$:
    - fault in induction generator → 1, non-fault → 0
  - Multi-class classification $(i = 0,1,2,3,\dots)$:
    - Stable atmosphere → 0, Neutral atmosphere → 1, Unstable atmosphere → 2

- Separation of classes depends on the input-output relationship:



http://www.statistics4u.com/fundstat_eng/cc_classif_calib.html

Classification

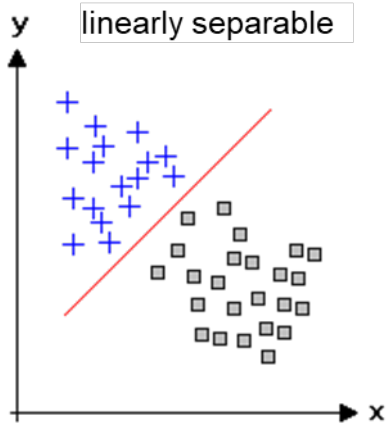
- Depends on the relationship, different classification algorithms exist:
  - Linear / Logistic Regression
    - linearly separable classes
    - mainly for (simpler) binary problems, could be extended to multi-class via One-vs.-rest (OvR) technique like:

$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 0,1,2,3,\dots)$$

  - Support Vector Machines (SVMs)*
    - linearly separable classes & linearly not-separable classes with kernel trick
    - Finds the decision boundary that separates the classes → *hyperplanes*

$$\beta_0 + \beta_n x_n = 0 \qquad (n = 1,2,3,\dots) \text{ for } n \text{ dimensional data}$$

      - Look for separate hyperplanes that classify classes correctly
      - Conduct optimization: pick up the one that maximizes the margin
    - Support vectors are the data points nearest to the hyperplane



*For in depth reading on SVMs and application examples in Python,
see https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

Classification

- Decision Trees
  - Set of **if-then-else decision rules**
  - Splits the data into subsets with *homogenous* values
  - Homogeneity is calculated via **entropy**

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i \qquad (i = 0,1,2,3, \dots, c)$$

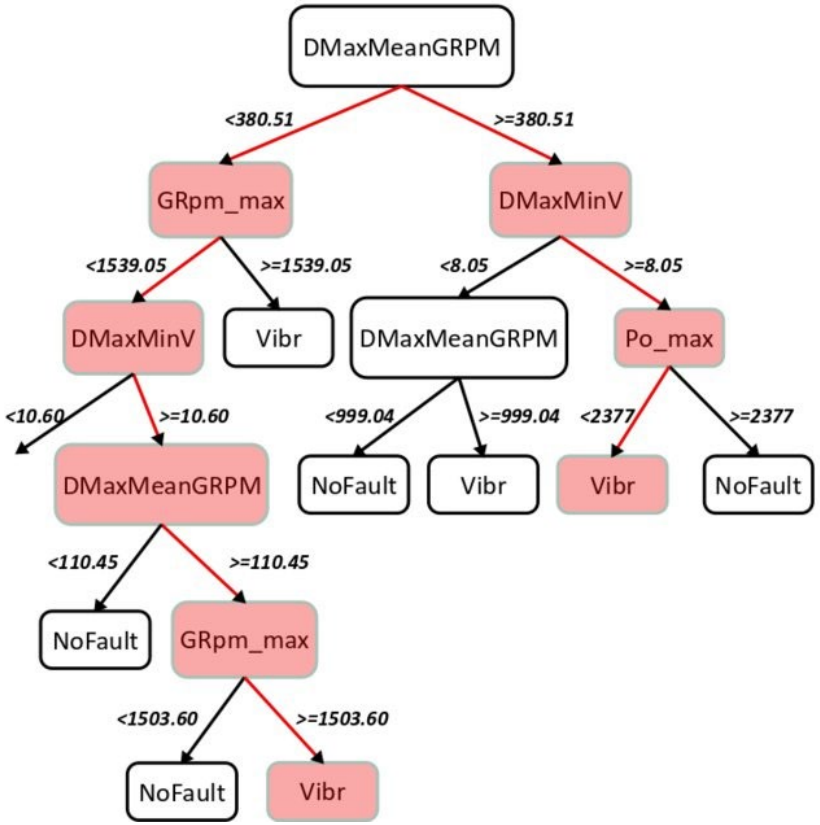
    - p_i is the probability of a class i
  - **Entropy** is a measure of disorder or uncertainty
  - Information Gain is used to see if we can reduce entropy by splitting data based on different attributes

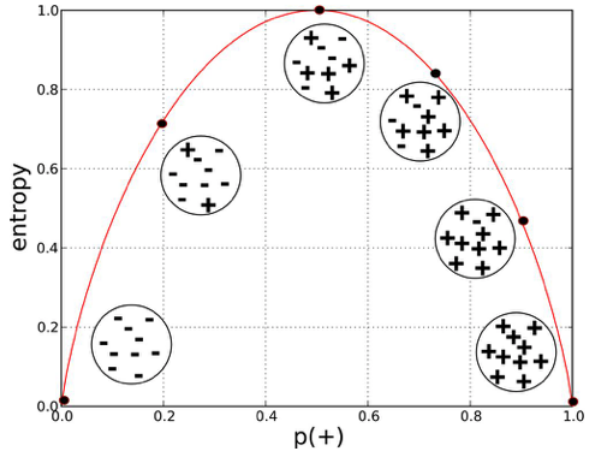
$$IG(Y,S) = E(Y) - E(Y|S)$$

  - We then choose the attribute with largest IG as the decision node

  → create the branch → repeat the process on every branch

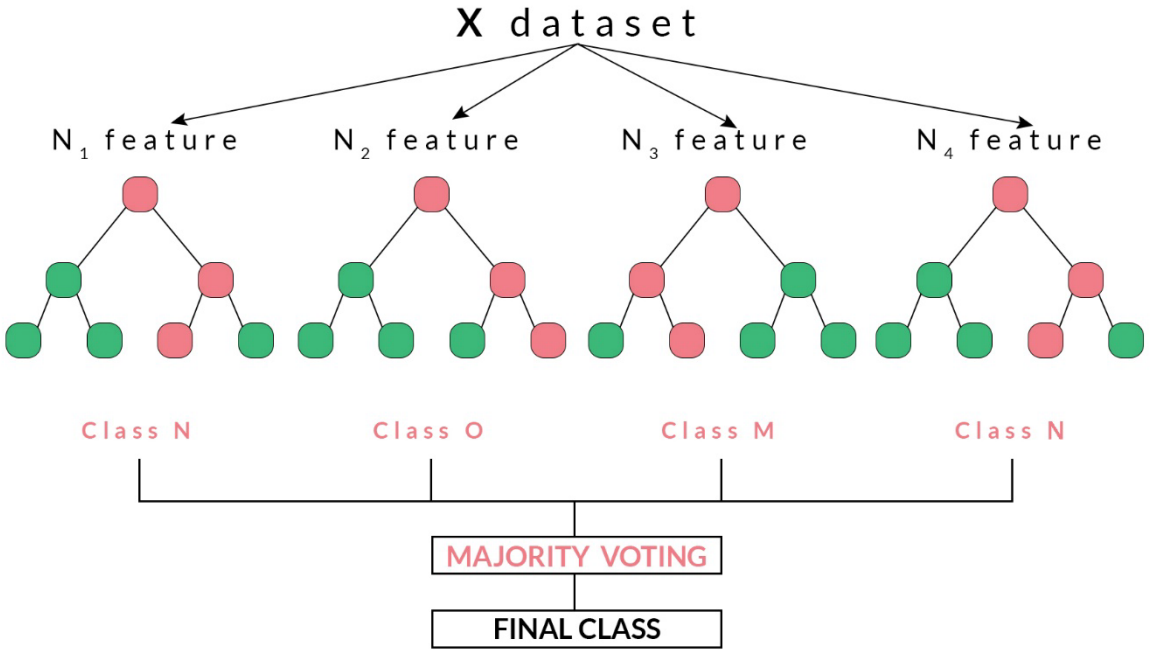
- Easy to use, can deal with irrelevant features → less preparation
- Prone to overfit…





Abdallah, Imad, et al. "Fault diagnosis of wind turbine structures using decision tree learning algorithms with big data." Safety and Reliability–Safe Societies in a Changing World (2018): 3053-3061.

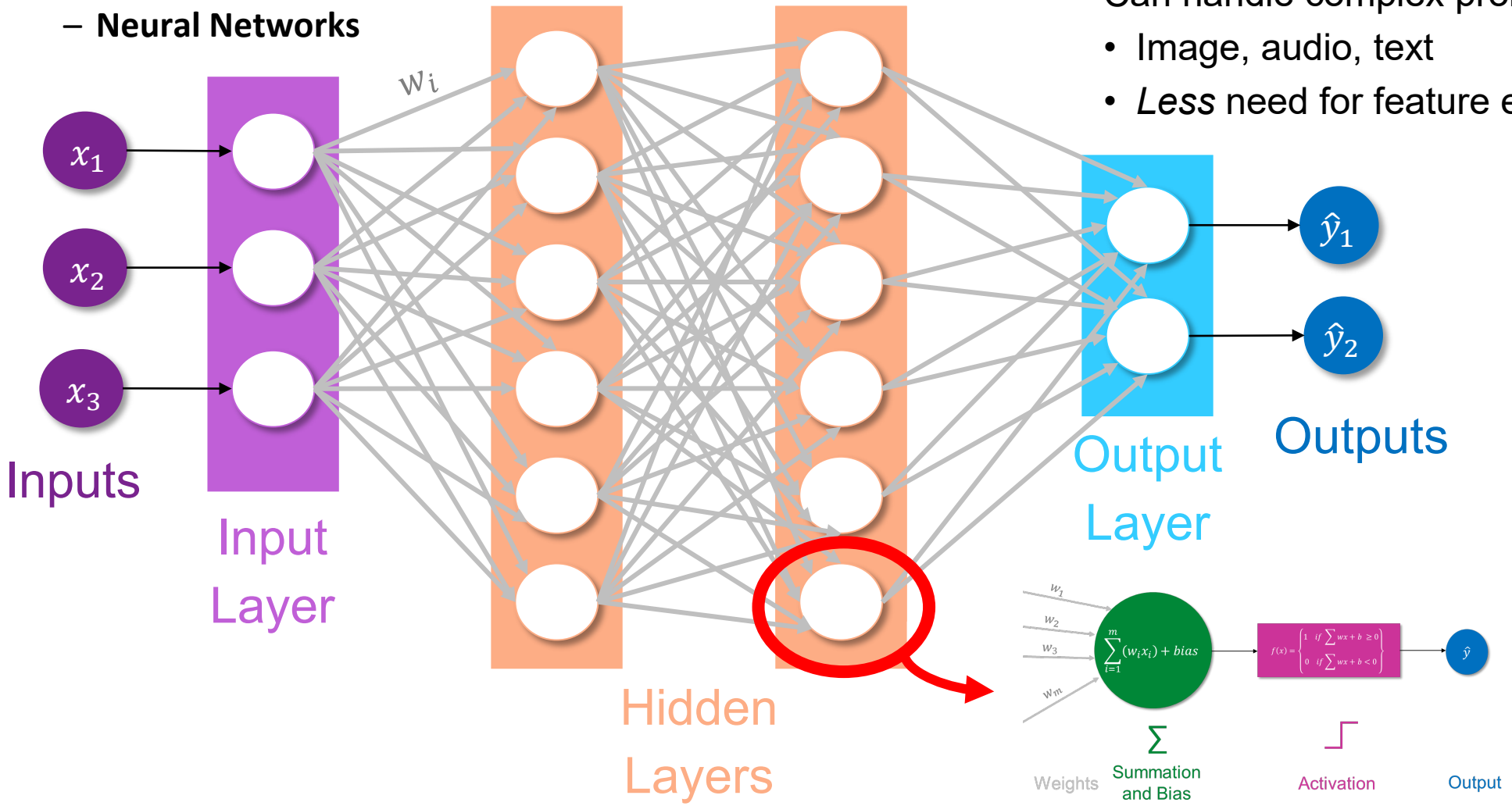
Classification



- **Random Forest**
  - Ensemble learning algorithm to increase robustness
  - Random subset of features, bootstrap sample of size *n*
    - *n* → number of trees
    - *n* tress are splitted in branches, maximizing IG
  - Then we aggregate the predicition done by each tree, assigning the class label by **majority vote**
- Few parameters to tune → *worry-free* approach
- Only works with tabular or categorical data

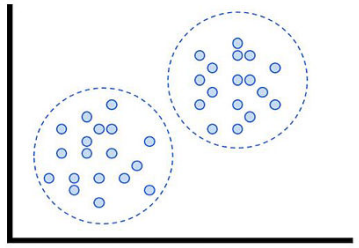
Classification

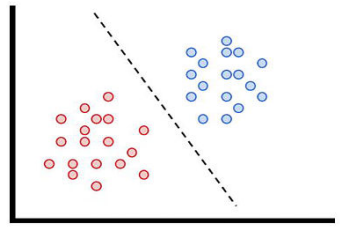
– **Neural Networks**



– Requires large dataset to work well
– *Very* parametric → requires experience
– Can handle complex problems
  • Image, audio, text
  • *Less* need for feature engineering

Clustering

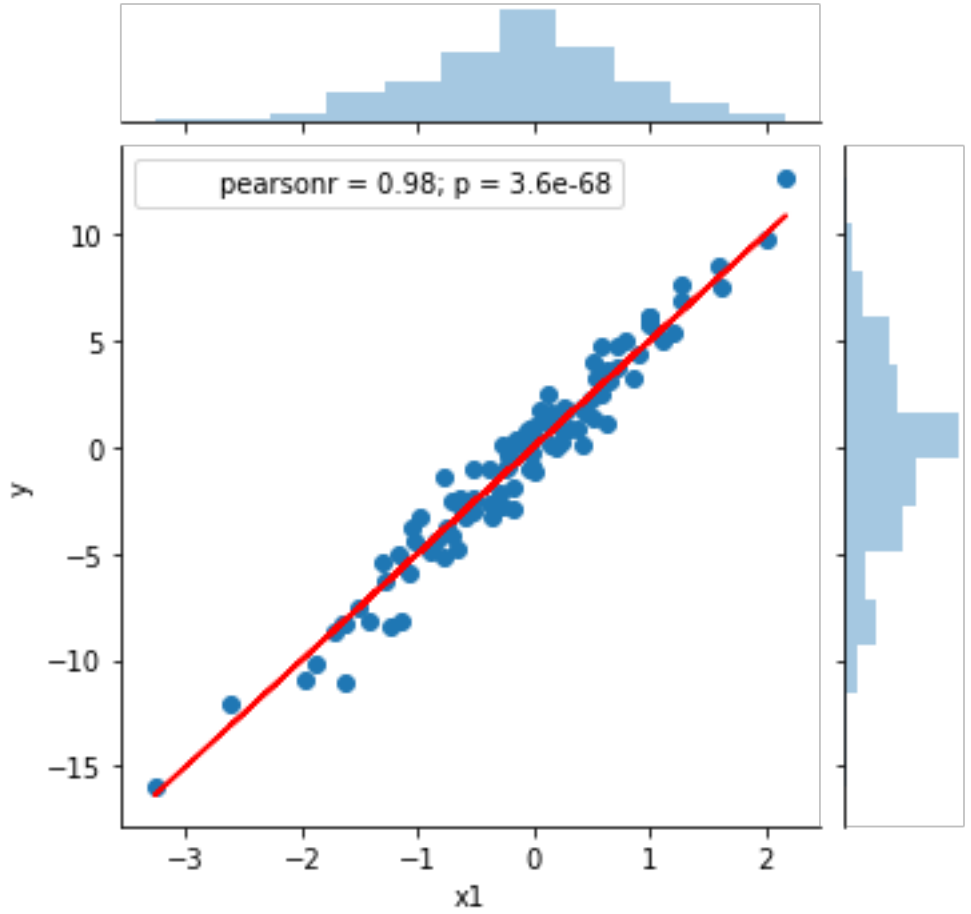




- Unsupervised learning
- Segregate groups with similar attributes and assign them into clusters
- 2 types of clustering
  - Hard assignment to a cluster
    - Either *completely* in or out of a given cluster
  - Soft assignment to a cluster
    - *Probably* or *likely* in a given cluster
- Different types of clustering algorithms, based on how *similarity* is defined
  - Connectivity models: *e.g.* hierarchical clustering algorithm
    - Based on the closeness to each other in data space → depends on the selection of the distance function
  - Centroid models: *e.g.* **k-means clustering algorithm**
    - Based on the distance from the centroid of the clusters, iteratively calculated
    - Number of clusters to be set beforehand → prior knowledge of the dataset
  - Distribution Models: *e.g.* **expectation-maximization (EM) with Gaussian mixture model (GMM)**
    - How probable is it that all data points in the cluster belong to the same distribution?
    - Multiple clusters per data point → final assignment based on the most probable cluster

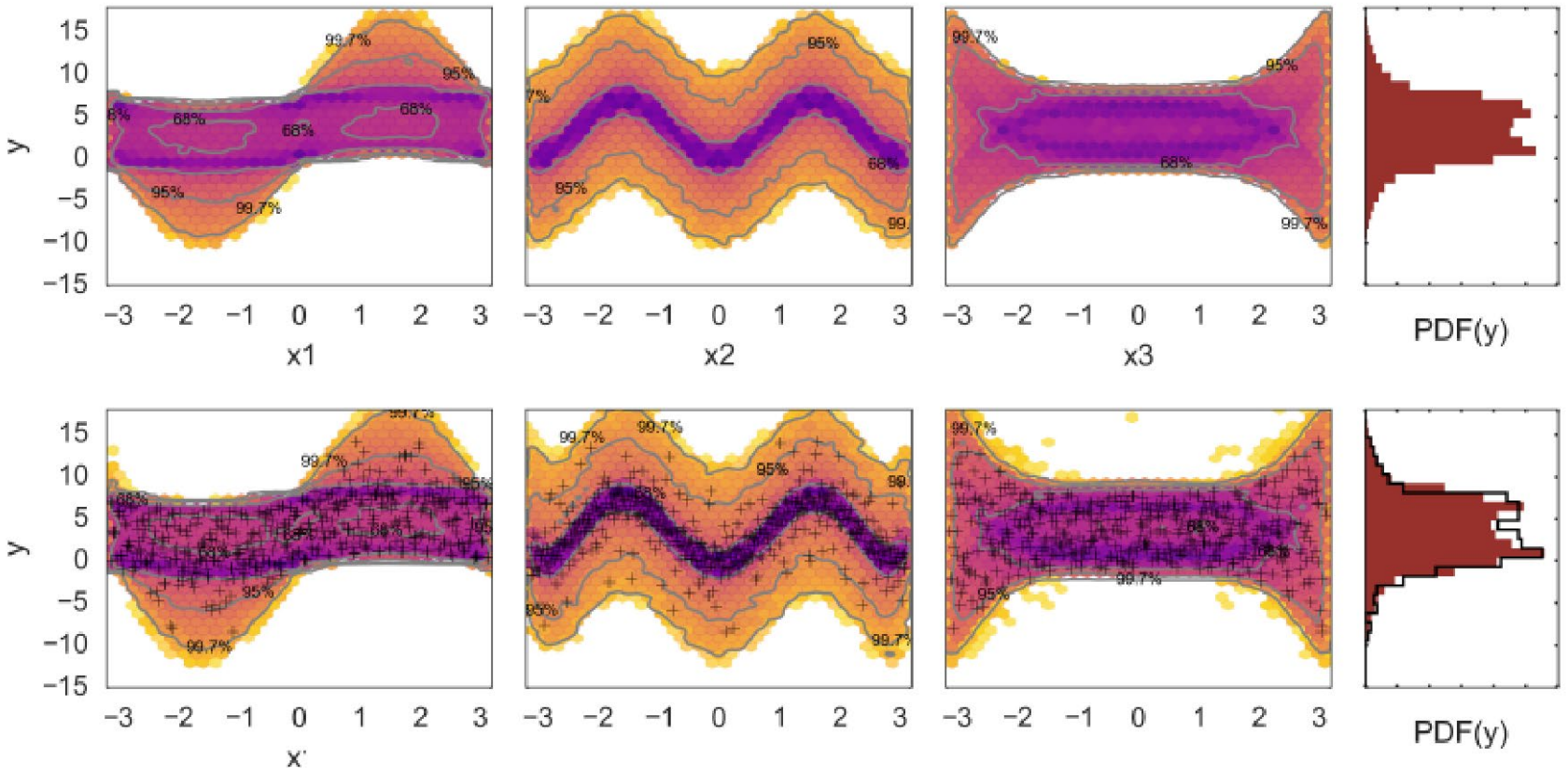
Regression:

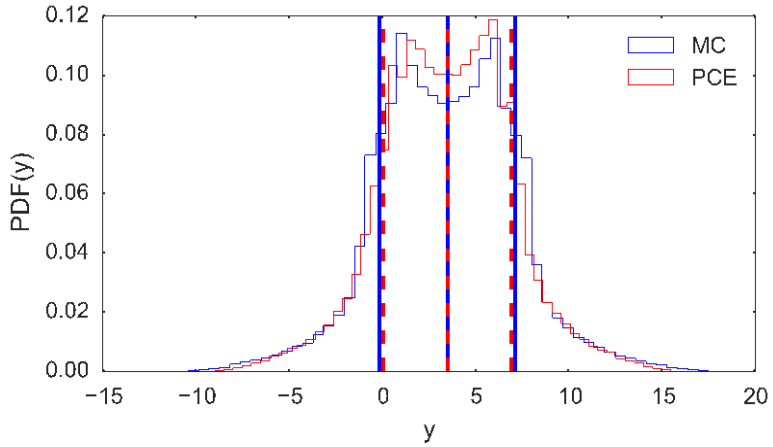
- Supervised learning technique that trains a model to follow the observations. Regressors also called surrogates.
- **Classic: fit the model to the observations**
- Multidimensional regression
- **Algorithms:**
  - Generalized Linear Models (polynomials)
  - RBF (Radial basis functions)
  - SVM (Support vector machines)
  - NN (neural networks)



Regression:

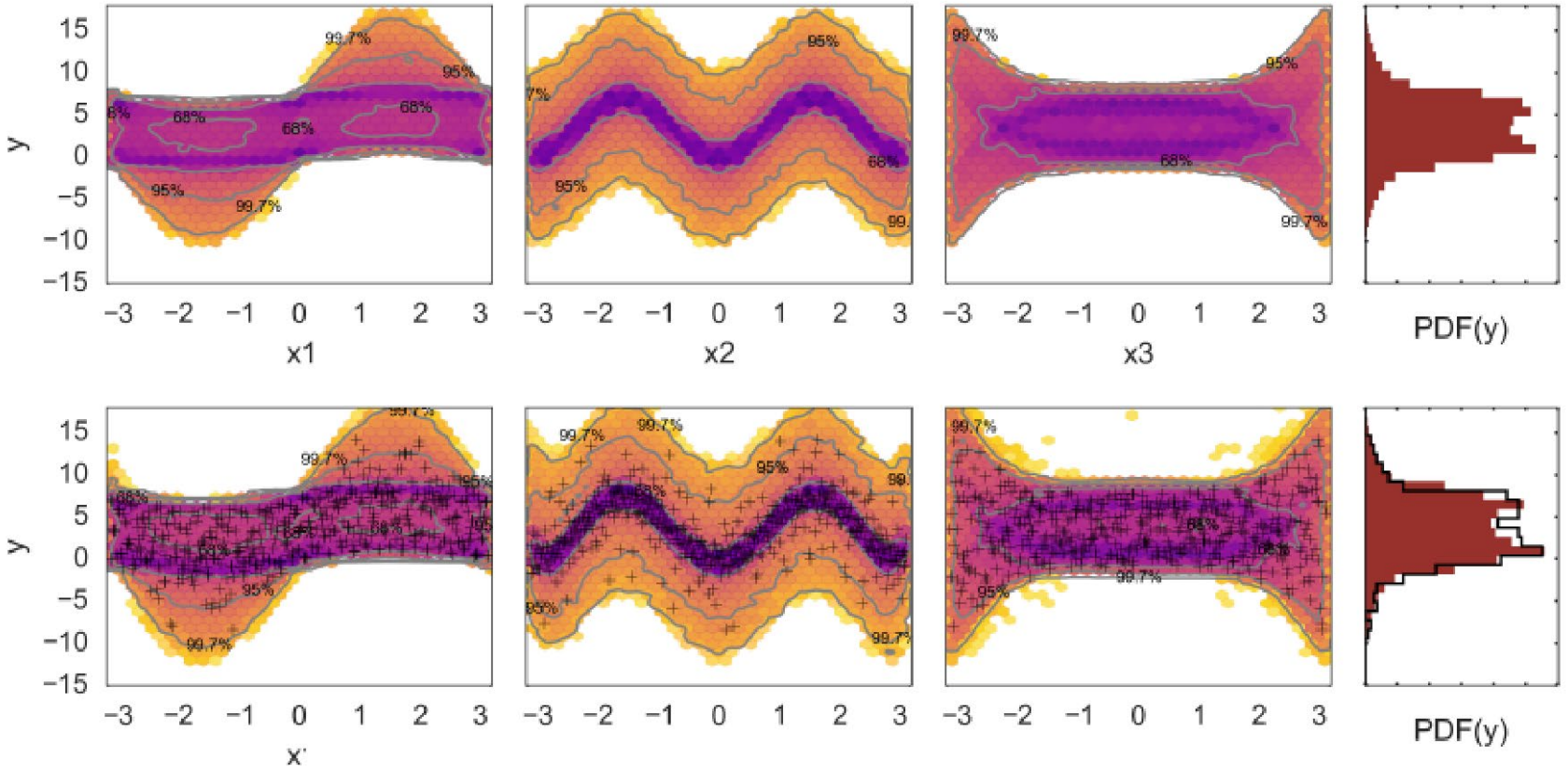
- Supervised learning technique that trains a model to follow the observations

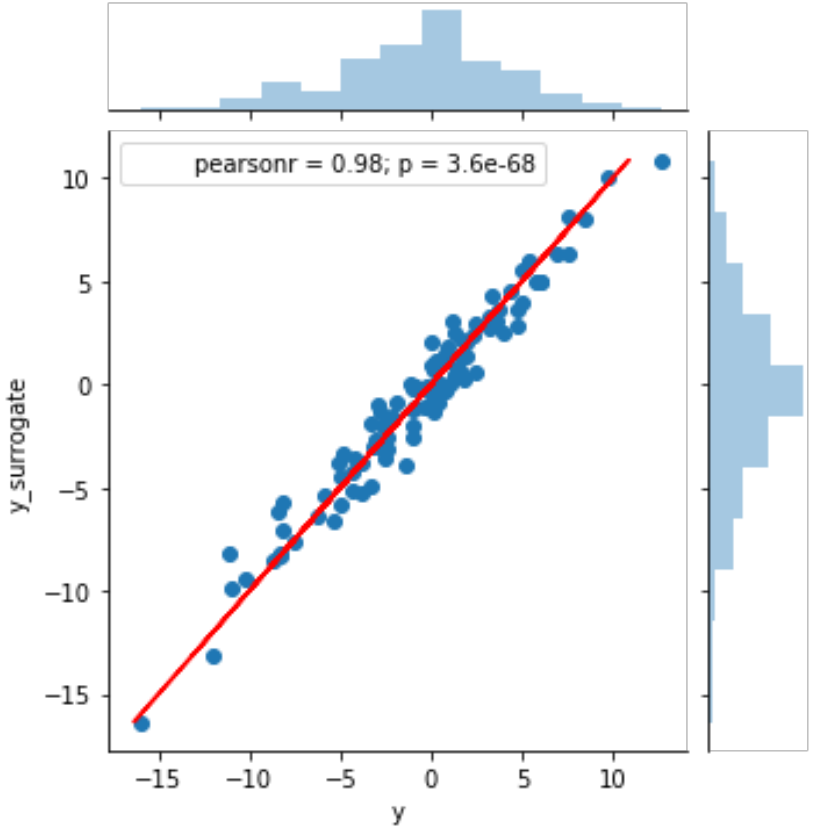




Regression:

- Supervised learning technique that trains a model to follow the observations





Classification model performance metrics

- Here presented with terminology related to binary classification, but also valid for cases with multiple classes
- P = Positive; N = Negative; TP = True Positive; FP = False Positive; TN = True Negative; FN = False Negative

Metric	Formula	Description
True Positive Rate (TPR) (also sensitivity, specificity)	$TPR = \frac{TP}{P} = 1 - FNR$	The fraction of correctly identified positive classes
True Negative Rate (TNR) (also specificity, selectivity)	$TNR = \frac{TN}{N} = 1 - FPR$	The fraction of correctly identified negative classes
Positive Predictive Value (PPV) (also precision)	$PPV = \frac{TP}{TP + FP} = 1 - FPR$	The ratio between true positives to total number of positive calls
False Positive Rate (fall-out)	$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - TPR$	Number of “false alarms”
False Negative Rate (miss rate)	$FNR = \frac{FN}{P} = \frac{FN}{FN + TP} = 1 - TNR$	Number of misses (no indication of true positive class)
Accuracy	$ACC = \frac{TP + TN}{P + N}$	Total fraction of correctly identified classes
F1 score	$F_1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR}$	Harmonic mean of precision and recall

Classification model performance metrics

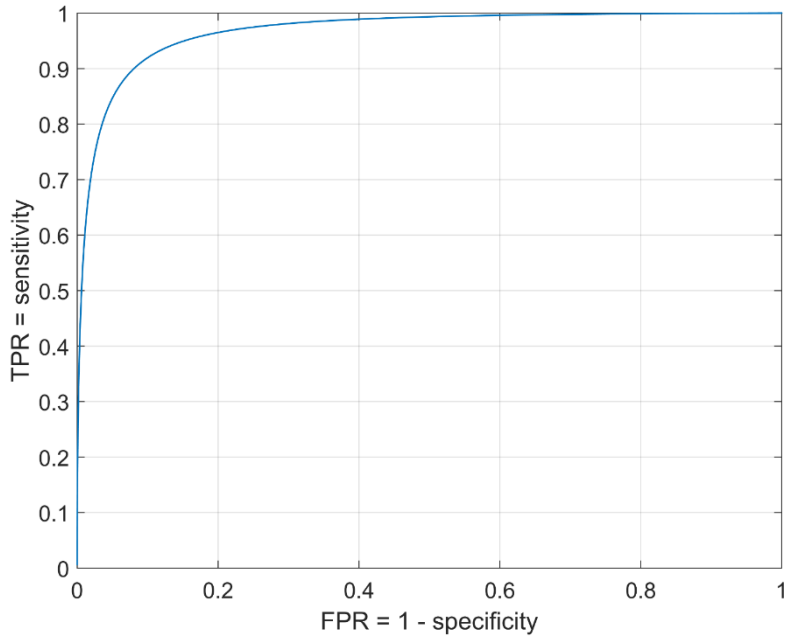
- P = Positive; N = Negative; TP = True Positive; FP = False Positive; TN = True Negative; FN = False Negative, TPR = True Positive Rate, FPR = False Positive Rate

Confusion matrix

True class		0	1
	0	True negative	False positive
	1	False negative	True positive

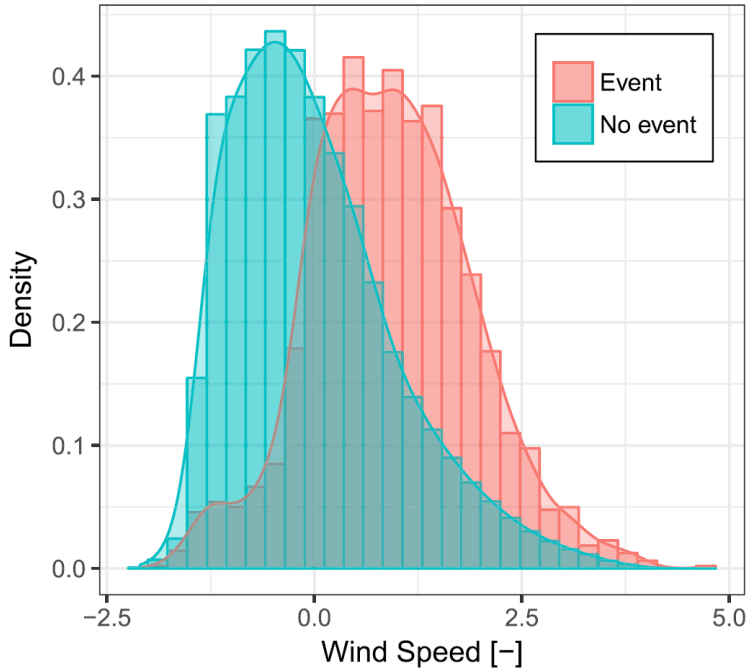
Predicted class

ROC (receiver operator characteristic) curve



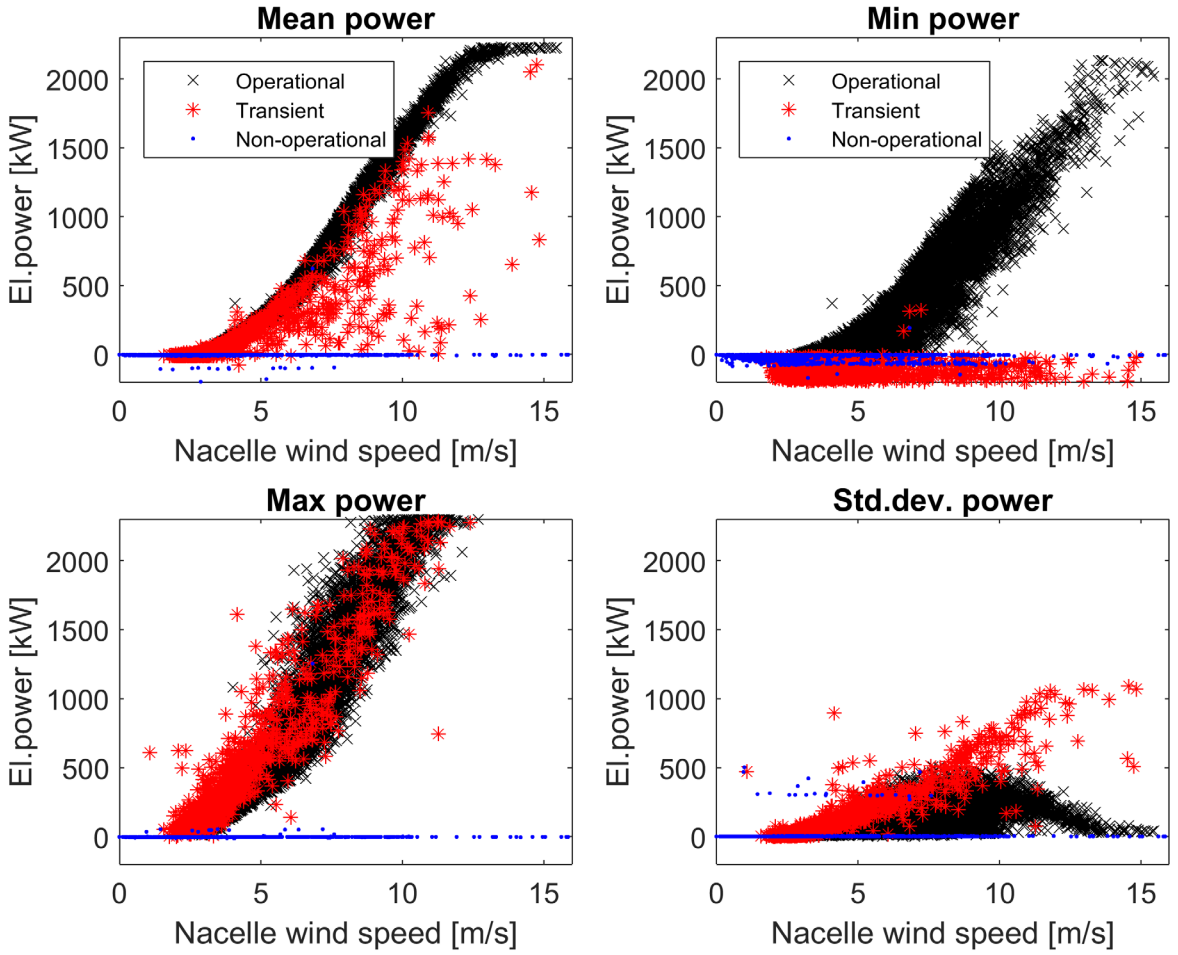
Kullback-Leibler Divergence Metric (KL metric)

- Defines the difference between two probability distributions
- Could show the discriminative power of an input variable
- Can be used to assess the change in model behavior after introducing model changes by e.g. checking the KL metric on the model error before and after the change
- Related to value of information



Classification – example from Wind Energy

- Based on operational (SCADA) data, determine the status of the turbine

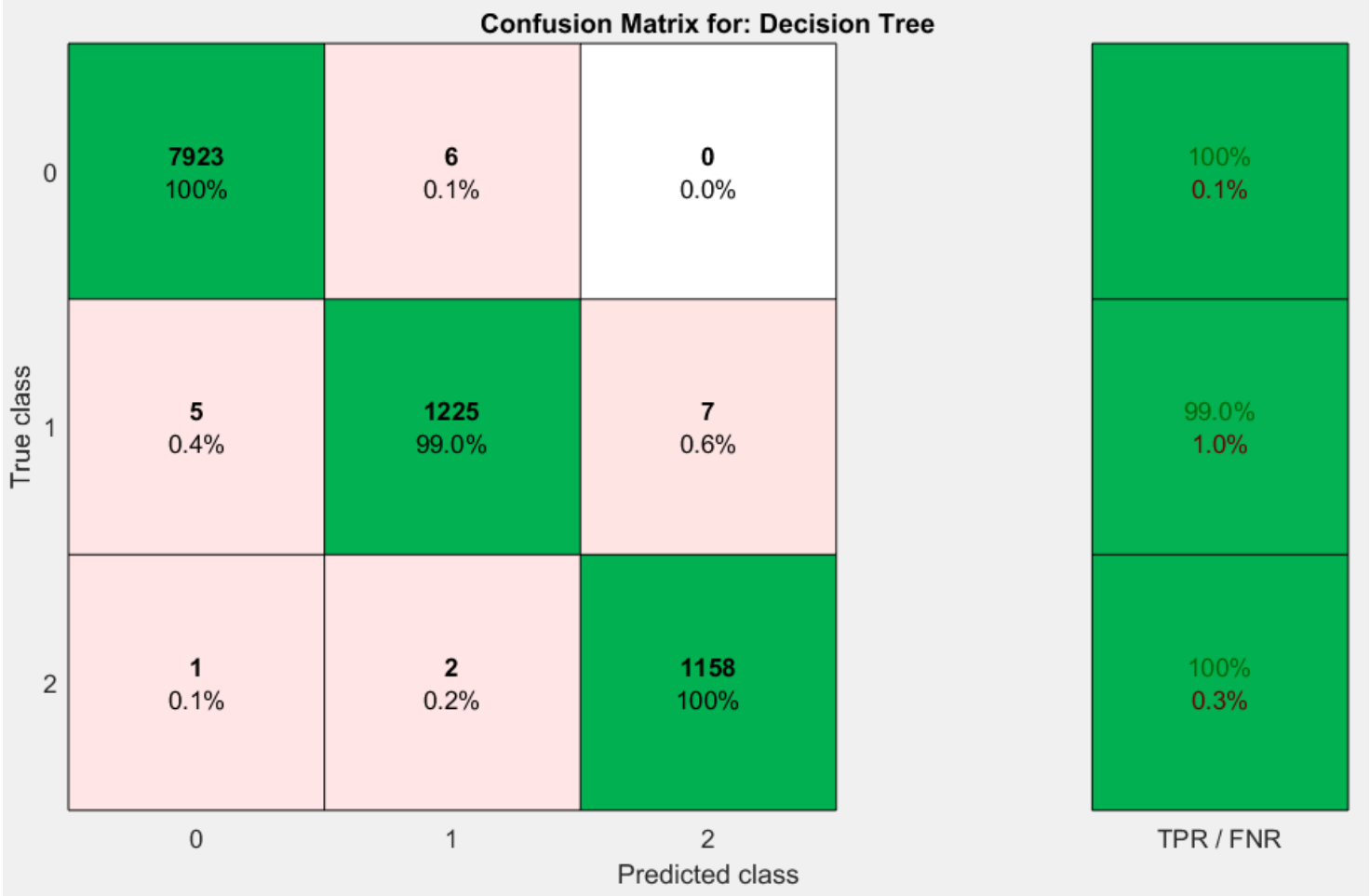


Classification – example from Wind Energy

- Random forest algorithm trained using 16 dependent variables:

- Nacelle wind speed
  (mean, std.dev, min, max)
- Active power
  (mean, std.dev, min, max)
- Blade 1 pitch angle
  (mean, std.dev, min, max)
- Rotor rpm
  (mean, std.dev, min, max)

(yaw error did not have much effect)



The concept of statistical hypothesis tests

- We have a hypothesis and want to test to what extent our data supports it.
- The “null” hypothesis: assumes there is no relationship between the observed phenomena (any relationship has occurred by chance)
- Our goal is to find out whether we can reject the null hypothesis
- The “p-value”: the level of statistical significance of our test.
- What is the relation between the p-value and the null hypothesis?

The concept of statistical hypothesis tests

- We have a hypothesis and want to test to what extent our data supports it.
- The “null” hypothesis: assumes there is no relationship between the observed phenomena (any relationship has occurred by chance)
- Our goal is to find out whether we can reject the null hypothesis
- The “p-value”: the level of statistical significance of our test.
- What is the relation between the p-value and the null hypothesis?

  … the p-value is the probability of the null hypothesis being true!

Statistical test examples

t-test:
measure whether the averages (expected values) of two populations differ significantly

- Null hypothesis: the two samples have identical averages

ks-test: (Kolmogorov-Smirnov test):
measure whether two samples could belong to the same statistical distribution

- Null hypothesis: the two samples have the same distribution

chi-square test:
measure whether there is agreement between the frequency of observation of two categorical variables. Can be used to test whether an input feature has a statistically significant effect on the frequency of occurrence of output categories

- Null hypothesis: the difference in observed category frequencies is due to chance

f-test:
check whether a certain variable has an effect on the variance of the dependent variable. Can be used to eliminate independent variables which have no effect on the output

- Null hypothesis: a specific variable has no effect on the explained variance

Statistical test examples

t-test: Python scipy implementation: scipy.stats.ttest_ind()

measure whether the averages (expected values) of two populations differ significantly

- Null hypothesis: the two samples have identical averages

ks-test: (Kolmogorov-Smirnov test): Python scipy implementation: scipy.stats.ks_2samp()

measure whether two samples could belong to the same statistical distribution

- Null hypothesis: the two samples have the same distribution

chi-square test: Python scipy implementation: scipy.stats.chisquare()

measure whether there is agreement between the frequency of observation of two categorical variables. Can be used to test whether an input feature has a statistically significant effect on the frequency of occurrence of output categories

- Null hypothesis: the difference in observed category frequencies is due to chance

f-test: Python implementation: sklearn.feature_selection.f_regression()

check whether a certain variable has an effect on the variance of the dependent variable. Can be used to eliminate independent variables which have no effect on the output

- Null hypothesis: a specific variable has no effect on the explained variance

Confidence intervals

Confidence intervals

- A specified range of values for which there is a given probability (p) that the true value of the variable falls within the range

- Confidence level:

$$\alpha = 1 - p$$

- Assuming a normally distributed population, the interval containing the probability mass p is

$$CI_{\pm}(X) = \mu_Y(X) \pm \Phi^{-1}(1 - \alpha/2)\sigma_Y(X)$$

Confidence intervals

Confidence interval for the mean of a normal population

The sample mean $\bar{X}$ of a normally-distributed population is normally distributed with mean μ and standard deviation $\sigma/\sqrt{n}$. It follows that

$$U = \frac{\sqrt{n}(\bar{X} - \mu)}{\sigma}$$

has the standard normal distribution.

Denoting $k_\alpha = \Phi^{-1}\left(\frac{\alpha}{2}\right) = -\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)$

the confidence interval for the mean of a population with known variance becomes

$$CI_{\pm} = \bar{y} \pm k_\alpha \frac{\sigma}{\sqrt{n}}$$

Confidence intervals

Confidence interval for the mean of a normal population

If the true variance of the population is unknown, it has to be replaced by an estimate – an obvious candidate is the sample variance, S.

Then, the expression $U = \sqrt{n}(\bar{X} - \mu)/\sigma$ becomes

$$T_{n-1} = \frac{\sqrt{n}(\bar{X} - \mu)}{S}$$

where T_{n-1} has the student's t-distribution with $n - 1$ degrees of freedom.

Denoting $t_{n-1,\frac{\alpha}{2}} = T^{-1}\left(\frac{\alpha}{2}, n-1\right) = -T^{-1}\left(1 - \frac{\alpha}{2}, n-1\right)$:

$$CI_{\pm} = \bar{y} \pm t_{n-1,\frac{\alpha}{2}} \frac{S}{\sqrt{n}}$$

i.e., $t_{n-1,\frac{\alpha}{2}}$ is the inverse student's t-distribution at $\frac{\alpha}{2}$ with n-1 degrees of freedom

Confidence intervals

Student's t-distribution

$$f_{T_n}(t) = \frac{1}{\sqrt{n\pi}} \frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \left(1 + \frac{t^2}{n}\right)^{-\frac{n+1}{2}}$$

- For finite n, it has heavier tails than the normal distribution
- Converges to the normal distribution when $n \to \infty$

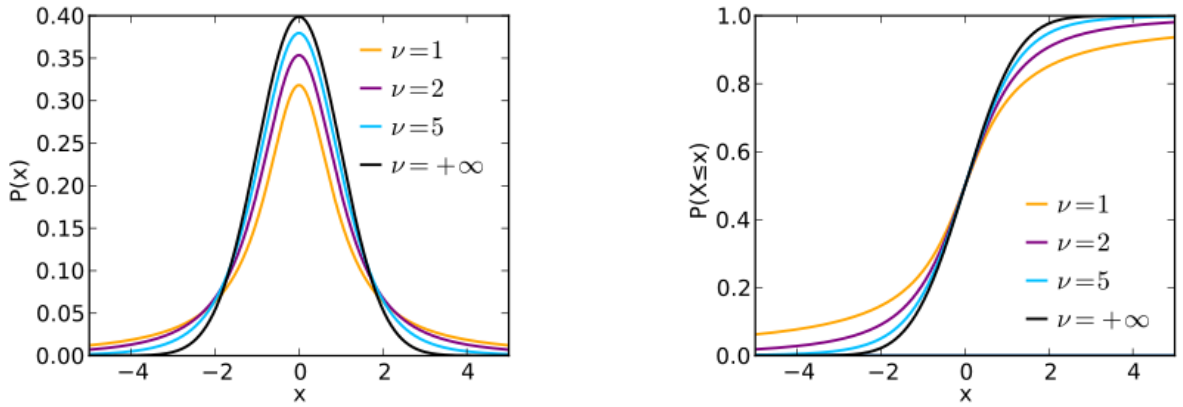


Image courtesy: Wikipedia

Confidence intervals

Student's t-distribution

$$f_{T_n}(t) = \frac{1}{\sqrt{n\pi}} \frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \left(1 + \frac{t^2}{n}\right)^{-\frac{n+1}{2}}$$

What is the relation between the student's t-distribution and the t-test?

- For finite n, it has heavier tails than the normal distribution
- Converges to the normal distribution when $n \to \infty$

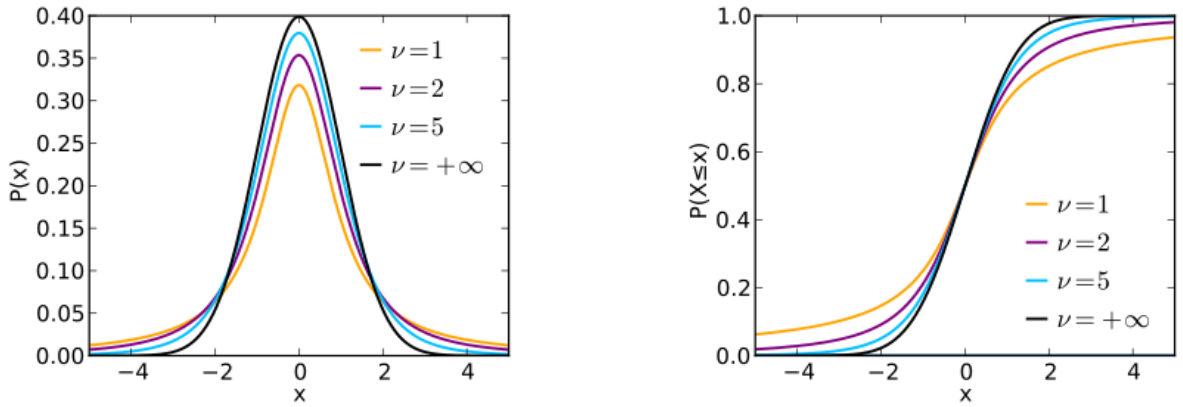


Image courtesy: Wikipedia

Confidence intervals

The bootstrapping method

(a.k.a. the Münchhausen principle)

- What if we have only one sample realization?

- We can simulate sample “realizations” based on parts or the entire original sample

1) All data points from the original sample of size N are given unique integer number $[1, N]$
2) Draw M random integers from the interval $[1, N]$, where $M \leq N$
3) Create a new sample by taking data points with indexes corresponding to the randomly drawn integers. Even for $M = N$, the new sample can differ from the original one, because repetitions are allowed when drawing the random integers

Confidence intervals

Confidence intervals by bootstrapping

- Based on a sample with size N, create l bootstrap samples of size $M \leq N$
- Calculate the quantity of interest R for each sample, e.g.

$$R_i = \bar{X}_i, \qquad i = 1,2, \dots, l$$

- Order the results by rank into a vector R^*, where the i^{th} ranked value is designated R_i^*
- At confidence level $\alpha = 1 - p$, the confidence intervals are

$$CI_- = R^*_{\left[\frac{\alpha}{2}(l+1)\right]} \quad ; \quad CI_+ = R^*_{[(1-\alpha/2)(l+1)]}$$

where the square brackets [] denote the nearest integer

Thank you!

Co-financed by the Connecting Europe Facility of the European Union

Funding scheme

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101006689

Project Coordinator

Nikolay Krasimirov Dimitrov

DTU Wind and Energy Systems

nkdi@dtu.dk