# Data & Things

## (Spring 25)

Friday February 14

**Lecture 7: Classification I**

Jens Ulrik Hansen

Roskilde Universitet

# Course schedule

| Lect. | Content | Lect. | Date |
|---|---|---|---|
| 1 | Intro. to course, data science, and python | Jens | Feb 3 |
| 2 | Data transformation and exploratory data analysis | Jens | Feb 5 |
| 3 | Data Engineering | Shabab | Feb 7 |
| 4 | Statistics | Jens | Feb 10 |
| 5 | Linear Regression | Jens | Feb 12 |
| 6 | Time series | Shabab | Feb 12 |
| **7** | **Classification I** | **Jens** | **Feb 14** |
| 8 | Classification II | Jens | Feb 17 |
| 9 | IoT and sensor data | Shabab | Feb 19 |
| 10 | Clustering | Jens | Feb 21 |

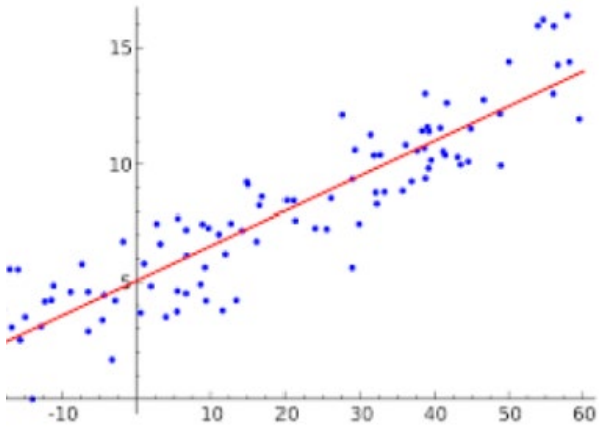
| Lect. | Content | Lect. | Date |
|---|---|---|---|
| 11 | Improving and selecting ML models | Jens | Feb 24 |
| 12 | Machine Learning Operations (MLOps) | Shabab | Feb 26 |
| 13 | Recommender systems | Jens | Feb 26 |
| 14 | Neural networks and deep learning I | Jens | Feb 28 |
| 15 | Neural networks and deep learning II | Jens | Mar 3 |
| 16 | Generative AI | Shabab | Mar 5 |
| 17 | Explainability | Shabab | Mar 7 |
| 18 | Ethical reflections on data science, end of course | Jens | Mar 10 |
| | Exan Q&A | Jens & Shabab | Mar 12 |

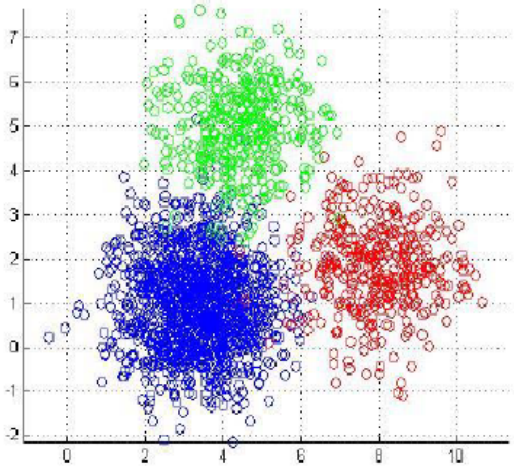
# Outline of this lecture

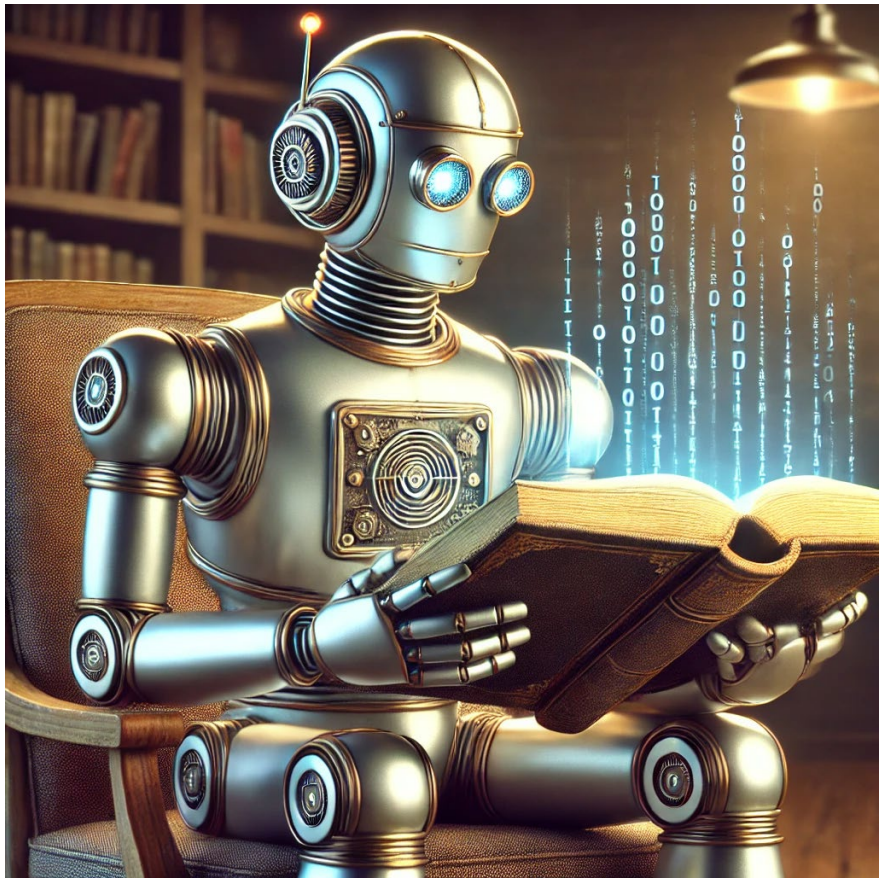
- Introduction to machine learning
- Measuring the performance of machine learning models
- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification

# Introduction to machine learning

- Machine learning is about developing algorithms or systems that can learn from data
- Machine learning is about detecting/ learning patterns in historical data useful for making predictions about new cases



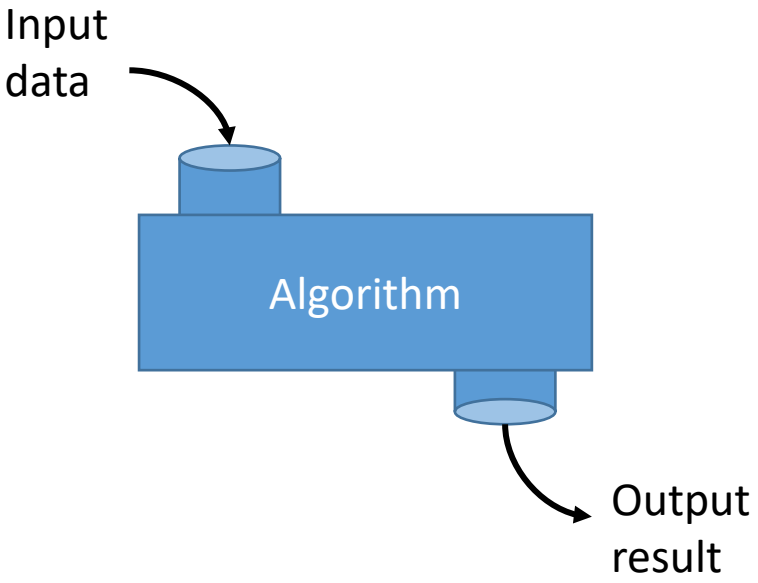



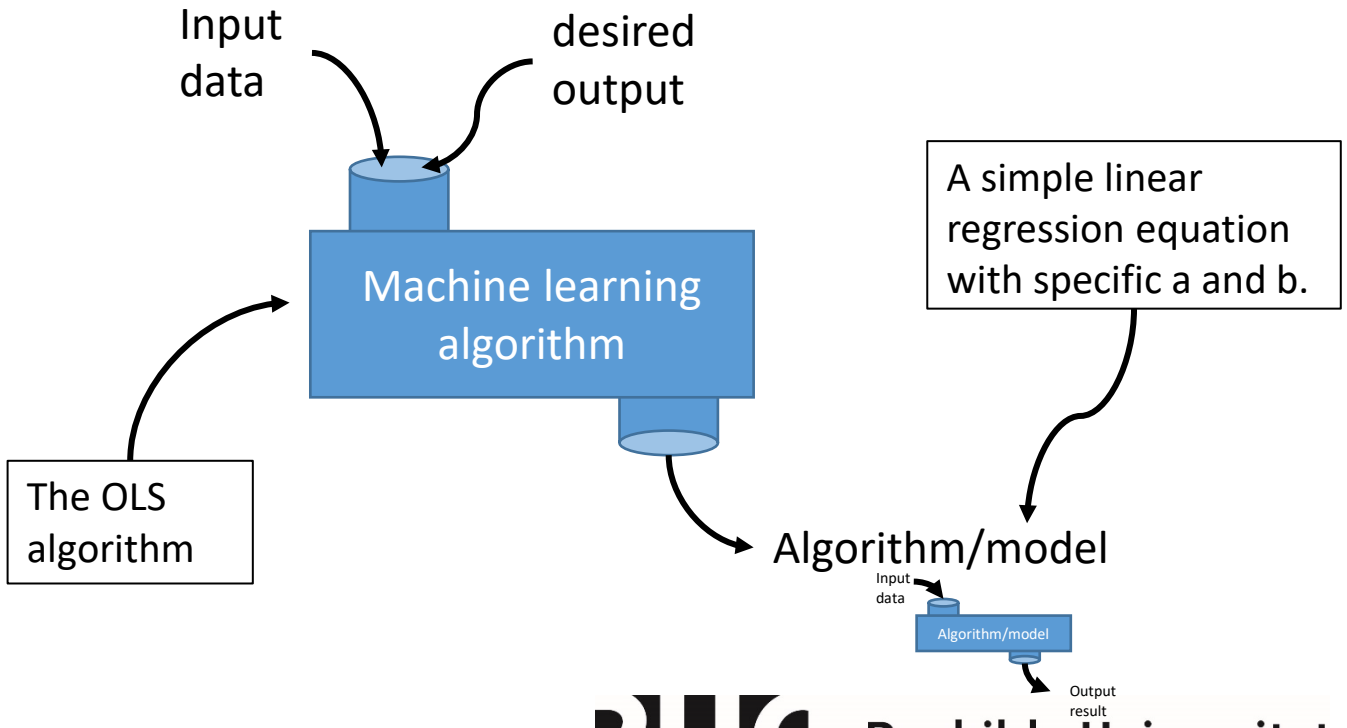
Dall-E, 2025

# Introduction to machine learning

- **Traditional programming (and symbolic AI)**
  - Some input is transformed using rules and specification to generate some output




- **Machine learning (supervised)**
  - Given input and desired output, we want to automatically learn an algorithm that generate that output from the input.



# Introduction to machine learning


Dall-E, 2025

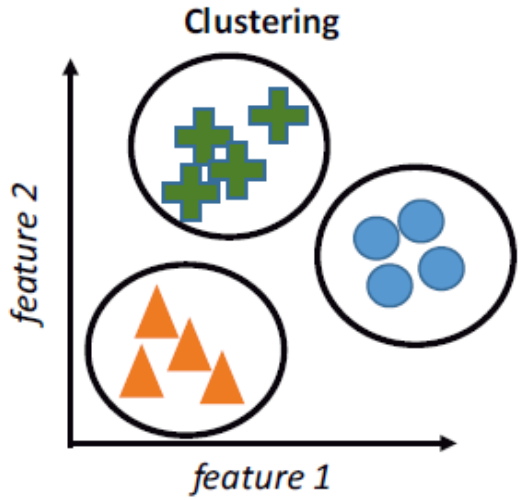
**Examples of Machine Learning applications**

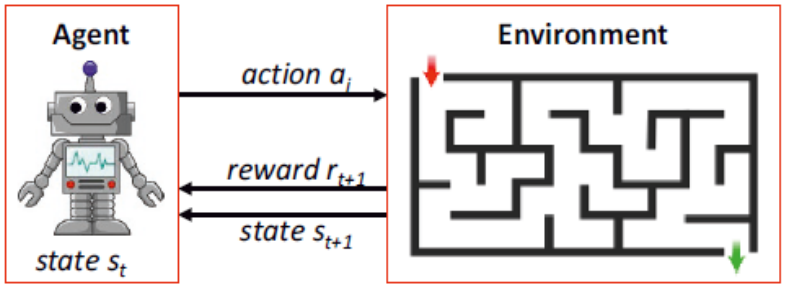
- Find out which website design make most visitors subscribe to a service (A/B testing)
- Predicting when a factory machine will break down or when a product will be returned (Predictive maintenances)
- Predicting when a customer will cancel a subscription or when an employee will quit (Churn analysis, People Analytics)
- Segmenting customers into groups to target individually (Clustering)
- Predicting if an image of a skin lesion shows sign of skin cancer (Image recognition)
- Learning the seasonality in a company's sales data (time series analysis)
- Recommending new products to buy, new series to watch, or posts to engage with (Recommender systems)

# Introduction to machine learning

- Three classic types of machine learning

  *1) Supervised learning*  *2) Unsupervised learning*  *3) Reinforcement learning*


** Mixing supervised and unsupervised learning or combining them is also becoming more common…*

# Introduction to machine learning

- Unsupervised learning find patterns in *unlabeled data*. It works on input data directly. (*Future lectures*)
  - E.g., clustering, customer segmentation, outlier detection for website access patterns, etc.
- Supervised learning generalizes from *Labeled data* to facilitate future predictions of label based on input data.
  - **Classification**: Predict a discrete value from a *pre-defined* set of class labels
    - E.g., given a loan applicant, predict if she/he is a *good* or *bad* client. (*Approval* or *rejection*)
    - More examples: Churn prediction, skin cancer detection, fraud detection in banking, spam filtering, etc.
  - **Regression**: Predict a continuous value from a continuous range
    - E.g., predict the price of a stock or the price of a house

# Introduction to machine learning

- **Data for supervised learning**
  - Uses supervised or label data, data of the form (input, output).
  - The *input data* is usually denoted by *X* and referred to as the *independent variable(s)*, *feature(s)*, or *predictor variable(s)*.
  - The output data is usually denoted by *Y* and referred to as the *dependent variable*, *response variable*, *target variable*, or *labels*.
  - Examples:
    - Features; Data about a house such as size, number of rooms, lot size, distance to school, neighborhood, etc. Response variable: House price
    - Features: Data about Titanic passengers such as age, passenger class, embarked, gender, etc. Response variable: Survived or not
    - Features: Data images of handwritten digits from 0 to 9, Response variable: a specification of what number the image depict

# Introduction to machine learning

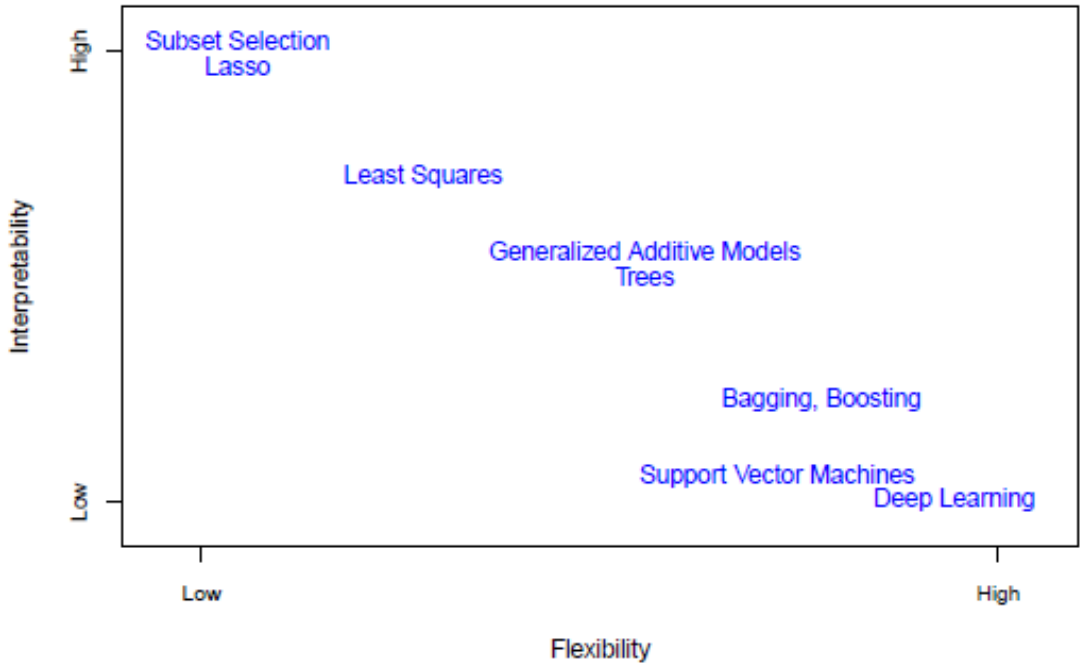
- **Supervised learning algorithms/models**
  - Given X and Y variables, we are essentially looking for a function **f** such that:
    - **$Y = f(X) + \varepsilon$** (where ε is a random error term independent of X and with mean 0)
  - Example: For simple linear regression we are looking for an f of the form:
    - $f(x) = a + b*x$
  - We can rarely find a f such that $Y = f(X) + \varepsilon$, instead, we have to settle with an f that approximately satisfy this $Y \approx f(X) + \varepsilon$ and ***we denote f(X) by $\hat{Y}$***.
  - Commonly, the f's we are looking for are ***parametric***, in the sense that f will have a particular form and there will be some parameters that defines f.
    - Example: For simple linear regression, f will have the form $f(x) = a + b*x$ and a and b are parameters that completely defines f.
    - If we are looking for an f of a particular form (we always are), finding f thus reduces to estimating these parameters – doing this is also referred to ***learning*** or ***fitting*** a model.

# Introduction to machine learning

- **Prediction vs. Inference** – Once we have such an f (a supervised machine learning model), we can use it for both:
  - **Prediction:** Given an f, we can use it to make predictions $\hat{Y}$ (=f(X)) of Y
    - Example: From our multiple linear regression model, we could predict the price of a house
    - Many tasks can be turned into a prediction task: Diagnosing cancer from an image, identifying a face, detecting credit card fraud, profiling citizens, churn prediction, when a factory machine will break down, whether an email is spam, …etc. ***– this makes supervised machine learning so powerful and useful!***
    - In this case, we are not interested in the inner workings of f
    - All the matters is predictive accuracy!
  - **Inference:** If f(X) is a good estimate of Y, we can potentially use f to understand the association/relationship between X and Y.
    - Example: The slope/x-coefficient in linear regression tells us how much Y changes whenever the corresponding x-variable changes by one unit – everything else being equal, how much is our house worth if we add another bathroom?
    - In this case, we are really interested in the inner workings of f!
    - We do not care as much about predictive accuracy
    - For more complex models than linear regression (and decision trees), interpretability of the models can be complicated and thus the models can be hard to use for inference!

# Introduction to machine learning

- **Trade-off between predictive accuracy and model Interpretability**
  - More complex models often allows for larger flexibility in f and thereby also higher predictive accuracy
  - Contrarily, more complex models are often much hard to interpret and thereby harder to use for inference
  - This is the ***interpretability-flexibility trade-off!***
  - Note, it is not always given that there must be a trade-off – sometimes we can improve both interpretability and accuracy!
  - Note also, that more flexible models do not always yield more accurate models, which has to do with ***overfitting***, which we will return to several times



# Outline of this lecture

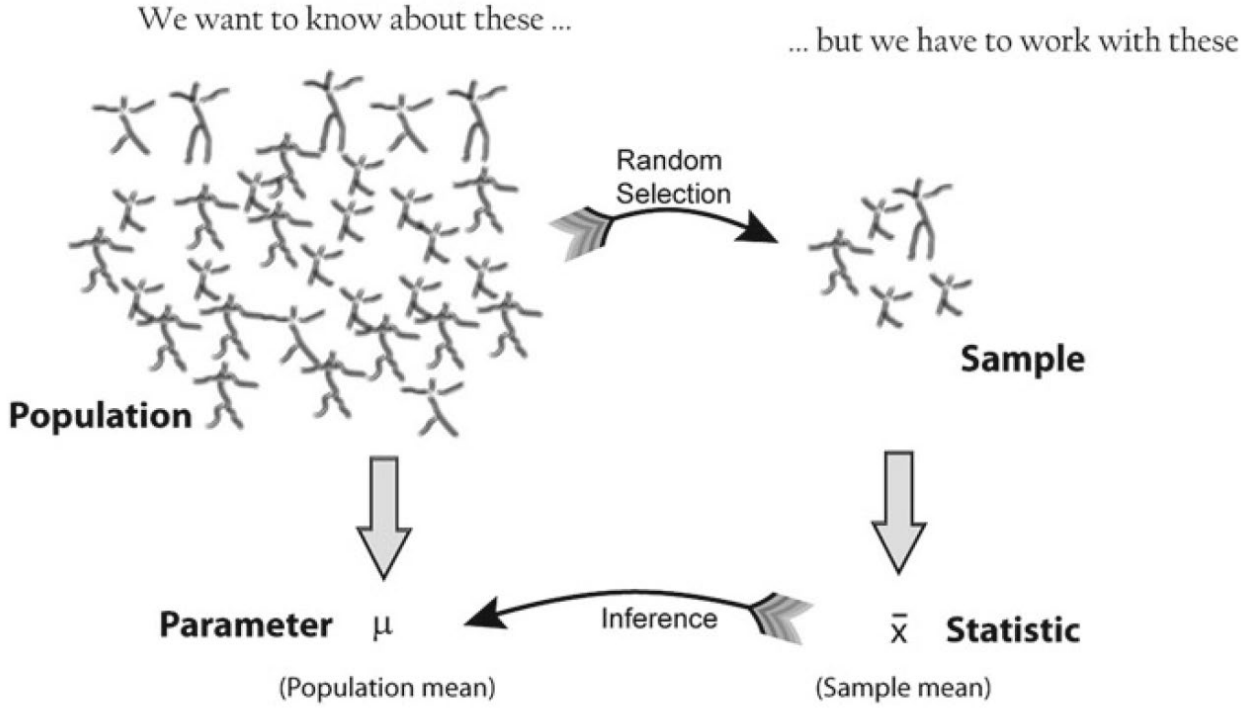
- Introduction to machine learning
- Measuring the performance of machine learning models
- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification

# Measuring the performance of machine learning models

- **Estimation of f and prediction errors**
  - Given an f, we can use it to make predictions $\hat{Y}$ (=f(X)) of Y.
  - The errors (Y- $\hat{Y}$) of $\hat{Y}$ in predicting Y can be decomposed into a ***reducible*** and an ***irreducible*** part.
    - The reducible error, we can ret rid of by making better models/estimating f better.
    - The irreducible error is ε, which is the inherent random errors in the system/nature or the things that effect Y, which we did not measure by Y.
    - In machine learning we try to reduce the reducible error as much as possible
    - The irreducible error will always be an upper bound on how accurate our models can become

# Measuring the performance of machine learning models

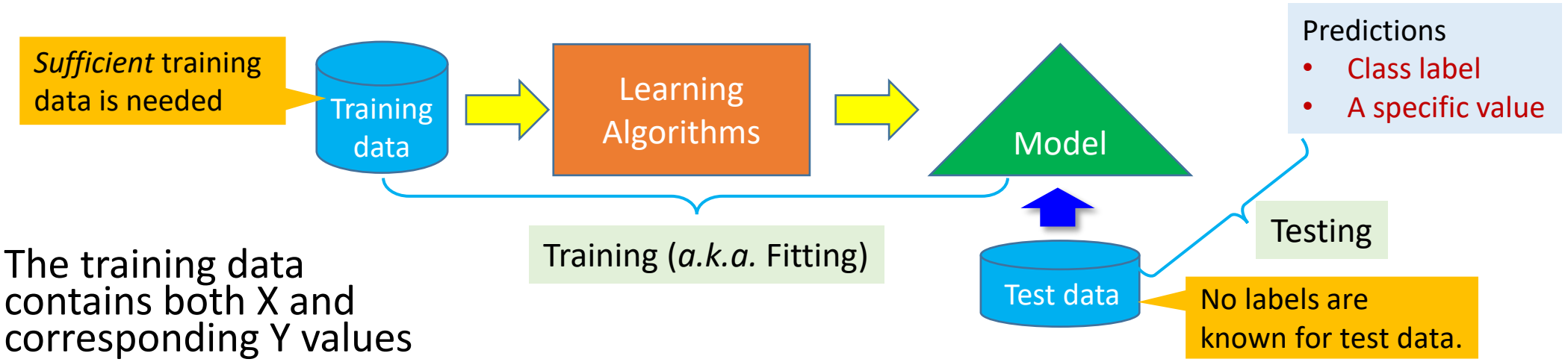
- In ***statistics***, we want to ***generalize*** from sample statistics to population statistics (from sample mean to population mean)
- In ***machine learning***, we want to build predictive models that make accurate predictions on the population – i.e. we want machine learning models that ***generalize*** well to the population
- In ***statistics***, we get an estimate about how good our generalization is by making ***assumptions about the data generation process*** (such as data comes from a normal distribution) that allow us to calculate things like p-values
- In ***machine learning***, we often make ***no assumption about the data generation process*** (we treat our predictive models as black boxes), so we need another way of measuring how well we generalize…




Haslwanter, T. (2022). *An Introduction to Statistics with Python - With Applications in the Life Sciences*. Springer, Cham.

# Measuring the performance of machine learning models

- To make sure we generalize well in machine learning, make a distinction between the data we ***train*** a model on and data we use to ***test/evaluate*** our model on.

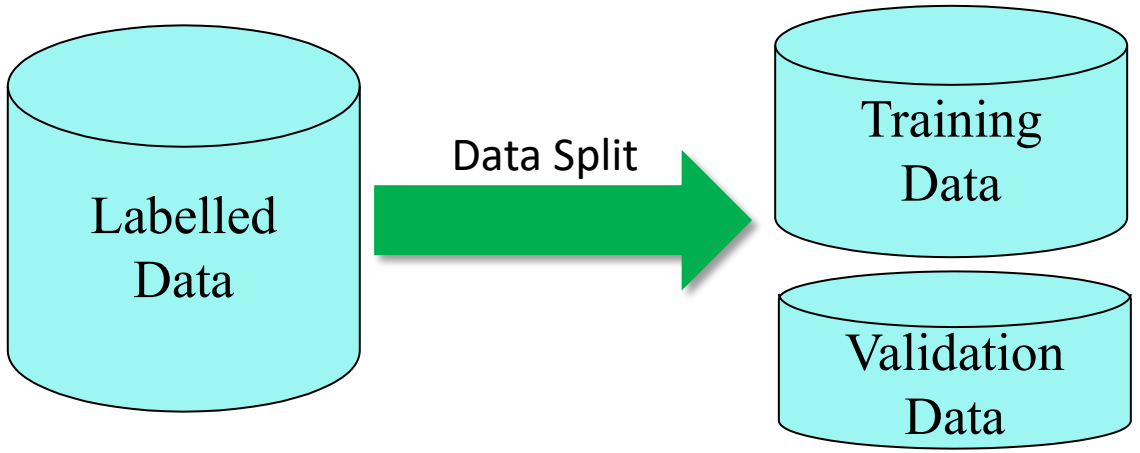



- The training data contains both X and corresponding Y values
- The test data is completely distinct from the train data
- Once the model is trained on the training data, we feed in the X part of the test data to make predictions $\hat{Y}$ on the test data.
- We then calculate our final evaluation of the model based on metrics (such as RMSE) that compare $\hat{Y}$ to the true Y from the test data.

# Measuring the performance of machine learning models

- **Example of train-test split in Python**
  - To make train-test split in Python. Let us look at the notebook “Measuring the performance of machine learning models.ipynb”.

- **Exercise**
  - Do Exercise 1 in the notebook “Exercises in Classification I.ipynb”



# Measuring the performance of machine learning models

- Overfitting: A model works well on the training data but generalizes poorly to unseen data (the test data).
  - Noises in the training data is learned as pattern.
  - Too many features are used in training.
  - The model type used is too complex.
  - Lack of proper variance in the training data
- Underfitting: A model even does not work well on the training data.
  - Too few features are used in training.
  - The model type used is too simple.
  - Training data is too little, failing to contain sufficient variance.

# Measuring the performance of machine learning models

## Example: Leaf Detection/Classification

- Training data (leaf samples)

- Test data (unseen)

Include more training samples.
E.g., those without sawtooth.

An *overfitting* model might say ”Not a leaf”.
- The training data samples all have sawtooth
- The model thinks a leaf must have sawtooth.

Use more features or
a more complex model.

An *underfitting* model might say ”A leaf”.
- Only color is used as the feature.
- The model thinks everything green is a leaf.

Include more training samples.
E.g., those in other colors.

An *overfitting* model might say ”Not a leaf”.
- The training data samples are all green.
- The model thinks a leaf must be green.

Leaf images from https://www.dreamstime.com/

# Measuring the performance of machine learning models

- **Overfitting** and **Underfitting**
  - The prediction ***error*** of a machine learning model f, can be ***decomposed into*** three components: ***The irreducible error***, ***the bias*** of f and ***the variance*** of f.
  - ***The variance*** is much f varies if we train f on different datasets. It is usually connected to the model's flexibility or tendency to overfit
  - ***The bias*** is the model's error when estimating complex phenomena with a simple model. In general, flexible models tends to have less bias
  - There is often a ***bias-variance trade-off*** as – we usually want to minimize both variance and bias!

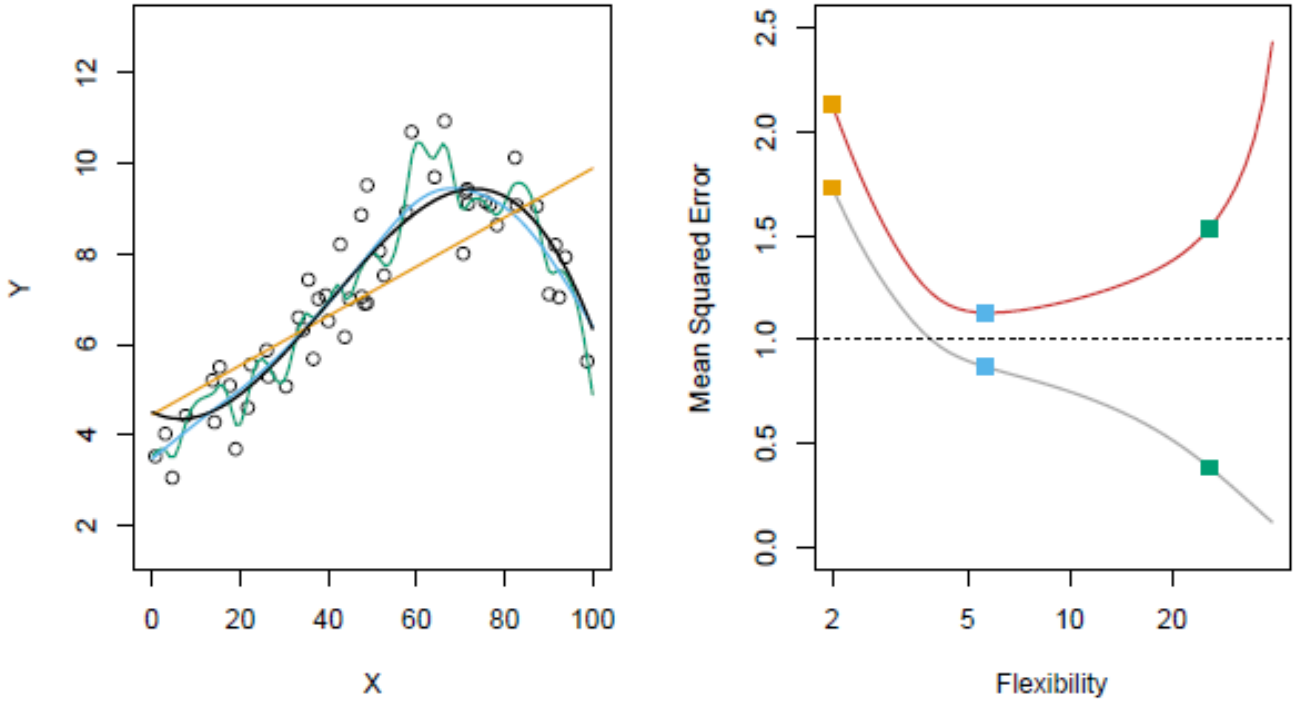



FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

# Outline of this lecture

- Introduction to machine learning
- Measuring the performance of machine learning models
- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification

# Introduction to classification in general

- Our target/response variable is now a *categorical variable*
  - If the categorical variable takes on only two values, we talk about ***binary classification***
  - If the categorical variable takes on more than two values, we talk about ***multi-class classification***
- ***The step of training a classification model is the same as just described***
  1. Import data and do data transformation and cleaning etc.
  2. split the labelled data into train and test sets
  3. Train the model on the training set
  4. Evaluate the model on the test set (and compare it training errors)

# Introduction to classification in general

- **Evaluate the model on the test set**
  - Note, we cannot quite use R-square, MAE, or RMSE for classification…
  - Instead, we can look at how many of the cases (data points/rows) the classification model correctly classified.
    - Example: For binary classification, we encode the two classes as 0 and 1 (the two values of the categorical response variable). If the true value is 0, but the model predicts a 1, it is an error, likewise, if the true value is 1 and the model predicts 0, it is also an error. Otherwise, there is no errors.
  - Counting the number of errors and dividing by the total number of cases, is called the ***error rate*** of a classification model, while the number of correctly predicted cases divided by the total number of cases is called the ***accuracy*** of the classification model.
    - Note that both the error rate and the accuracy can be calculated both on the training set and the test set
    - It turns out that there is a lot of other possible and interesting measures of how good a classification model is – we will talk more about this next time!

# Outline of this lecture

- Introduction to machine learning
- Measuring the performance of machine learning models
- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification

# K-nearest neighbors for classification (KNN)

- A non-parametric classification model
- Essentially, we try to classify a new data point based on the classes of its ***K nearest neighbors***, where K is some fixed number.
  - What is a ***neighbor***?
  - What does ***nearest mean***?
  - ***How do we decide*** what class a new point should belong to if we know the classes of the K nearest neighbors?
- Example to the right: 2D data (two features $X_1$ and $X_2$), two classes blue and orange
  - (The blank line(s) in the right figure is called the ***decision boundary***)

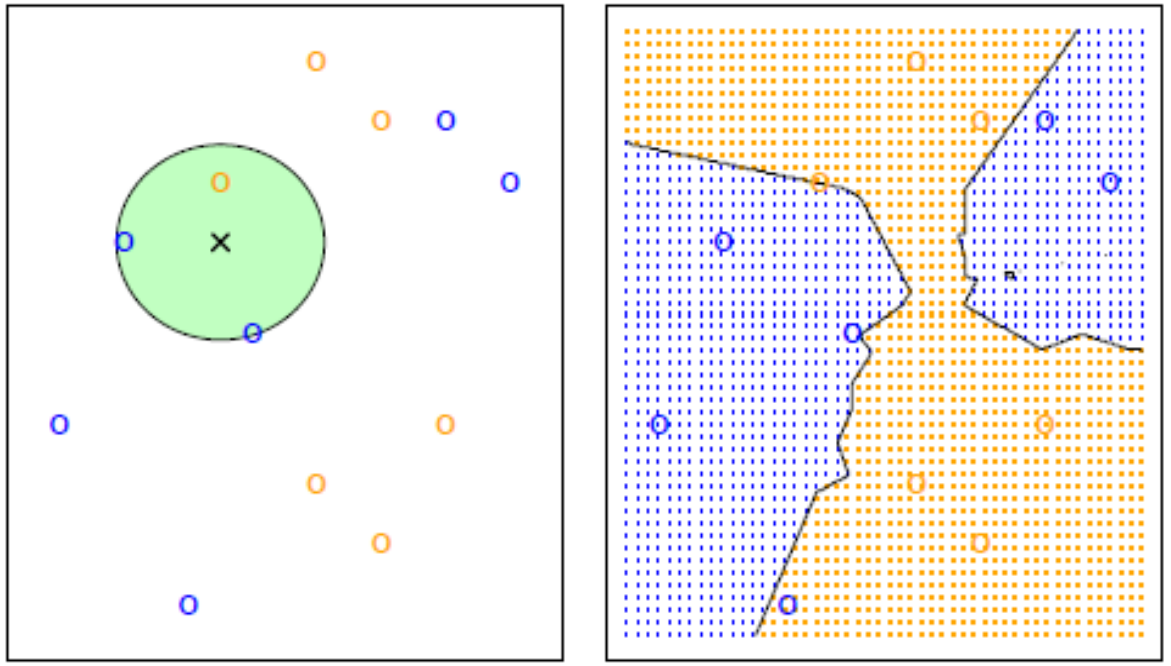


Figure 2.14 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

# K-nearest neighbors for classification (KNN)

- We classify a new data point based on the classes of its ***K nearest neighbors***, where K is some fixed number.
  - ***Neighbors*** are other ***datapoint from the training set***
  - ***"Neareness"*** is measured by some ***distance metrics***, often the ***Euclidian distance***
  - ***How do we decide:*** We assign the new datapoint to the class that has the highest presence among the K neighbors – i.e. by ***voting***

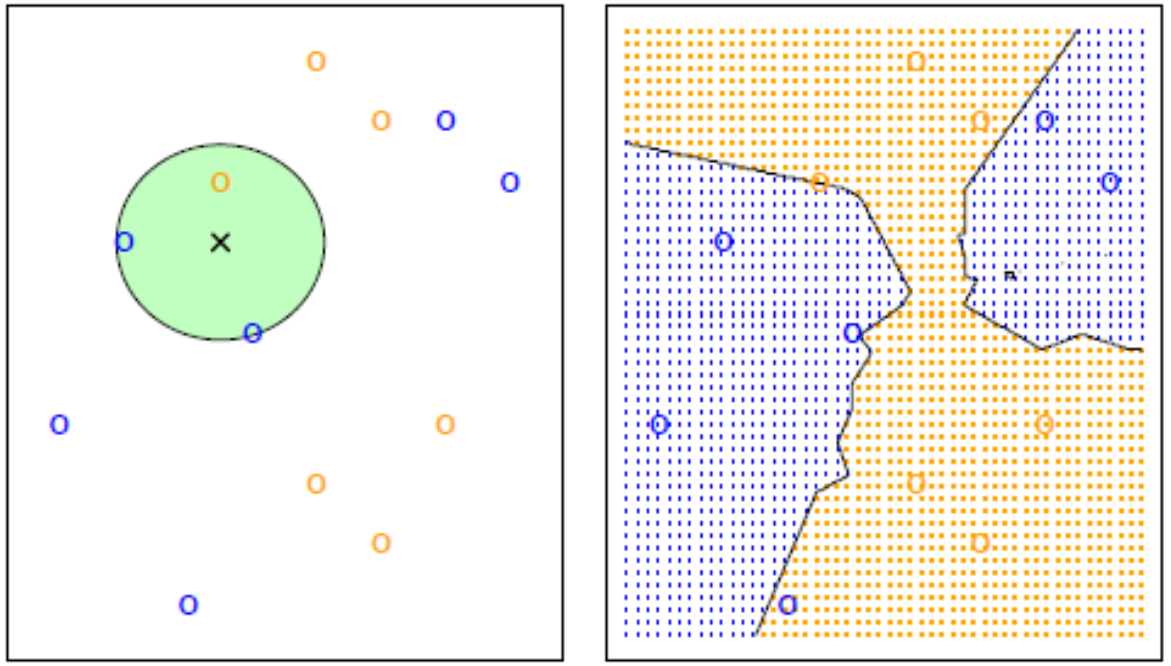


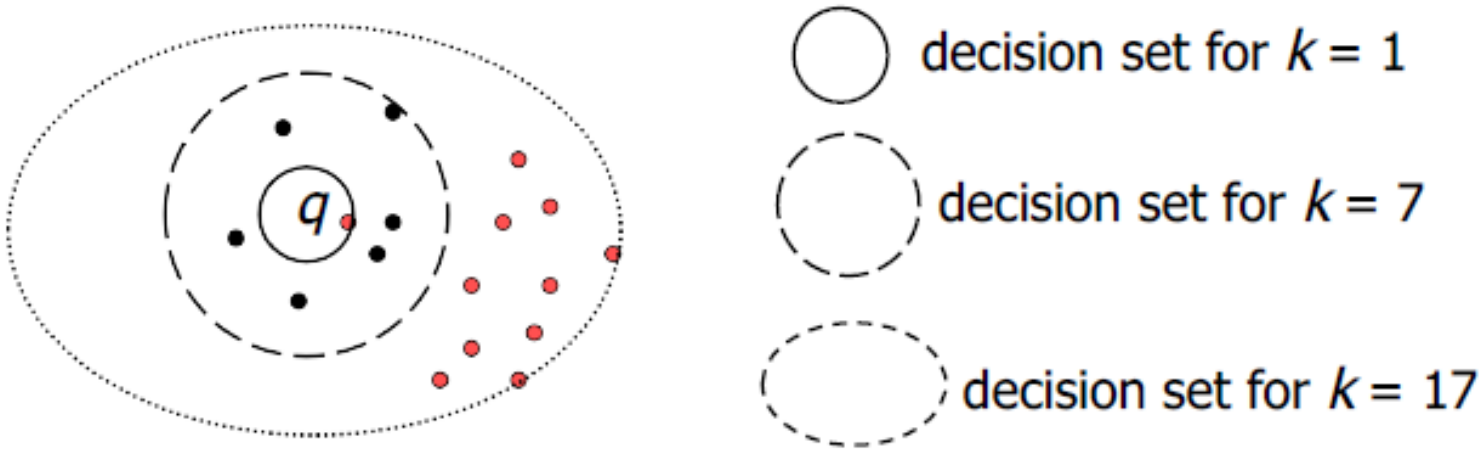
Figure 2.14 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

# K-nearest neighbors for classification (KNN)

- The training data
  - A training set containing $X_1$, $X_2$, …, $X_m$ different feature variables, where each row (data point) is labeled with a class represented by a response variable y.
- Classification
  - For a given new data point ($x_{i1}$, $x_{i2}$, …, $x_{im}$) to be classified, we find its **K** nearest neighbor data points from the training set according to some distance measure (the Euclidian distance, for instance).
  - Count the class labels in the KNNs, and gi*ve the new data point* the most frequent class label.
    - NB: the **decision rule** can be different.

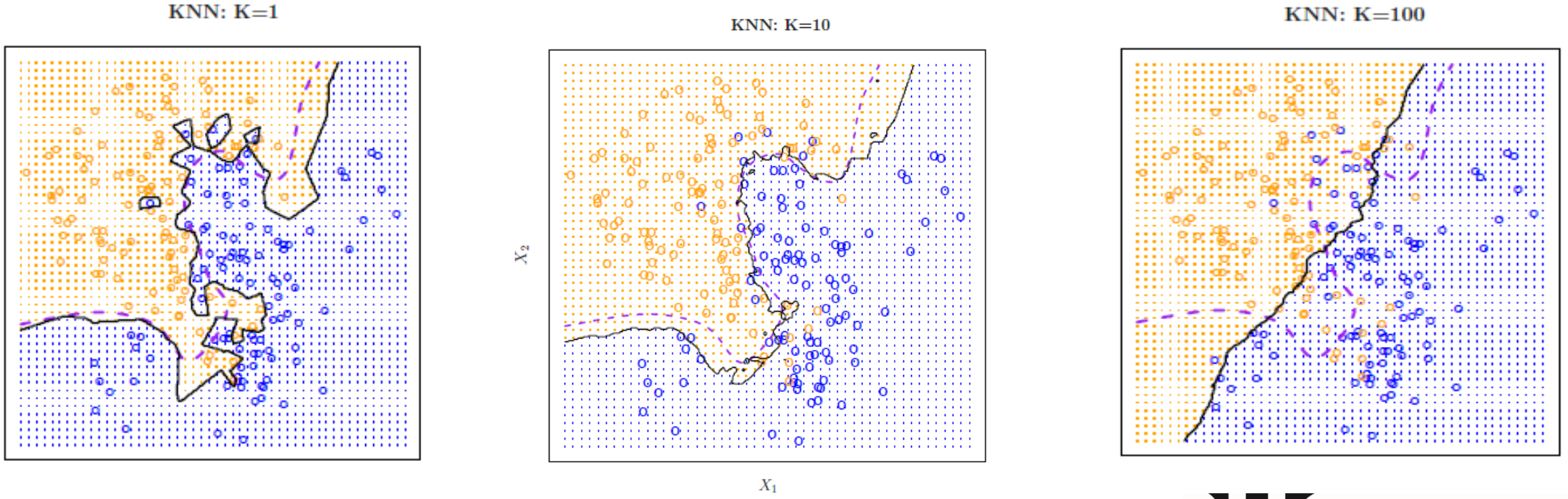
# K-nearest neighbors for classification (KNN)

- **Appropriate Value for K**
  - *Different Ks may lead to different classification results*.
  - Too small K: High sensitivity to outliers
  - Too large K: Decision set contains many items from other classes.
  - Empirically, 1<<K<10 yields a high classification accuracy in many cases.



# K-nearest neighbors for classification (KNN)

- **Appropriate Value for K**
  - Smaller K, more flexibility and variance and tendency to overfitting
  - Larger K, less flexibility and more bias



# K-nearest neighbors for classification (KNN)

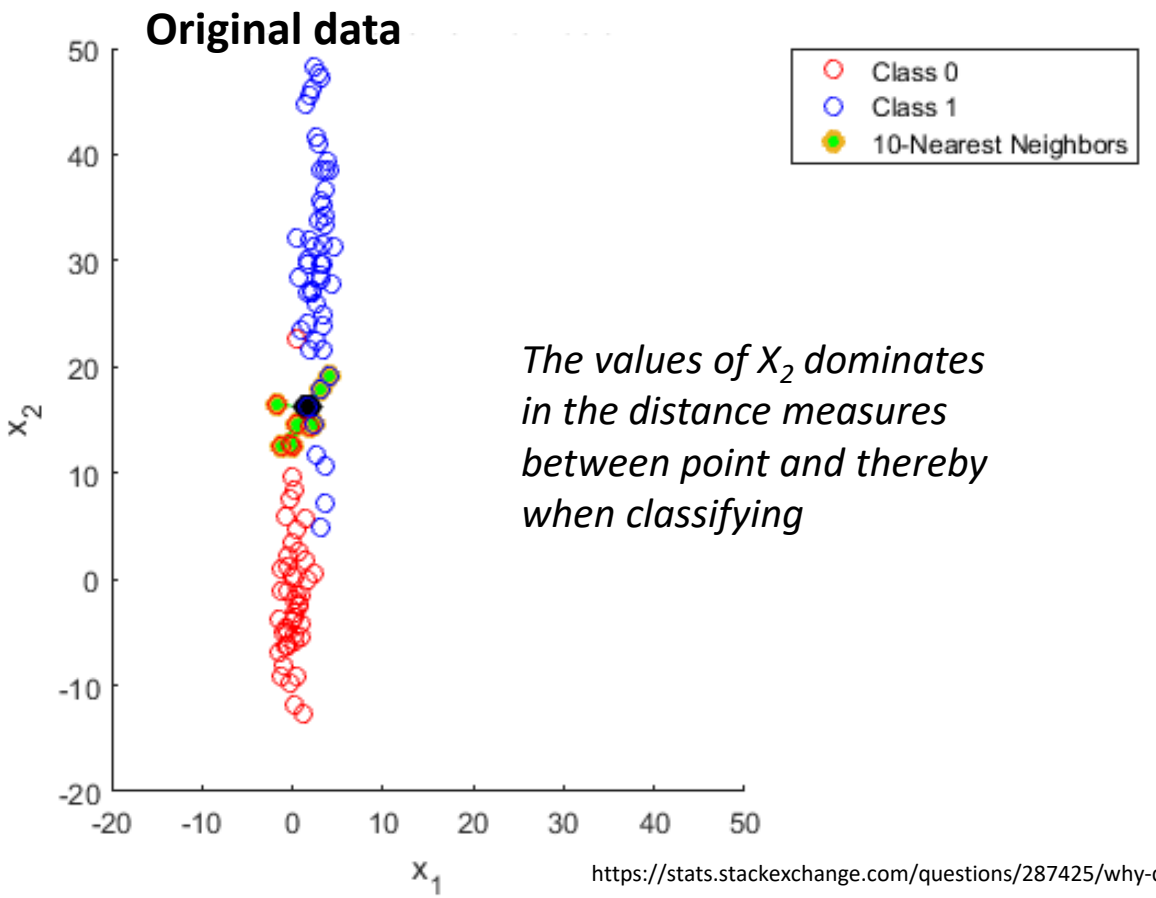
- Let us look at an example in Python, let us look at the notebook “K-nearest neighbors.ipynb”

# K-nearest neighbors for classification (KNN)

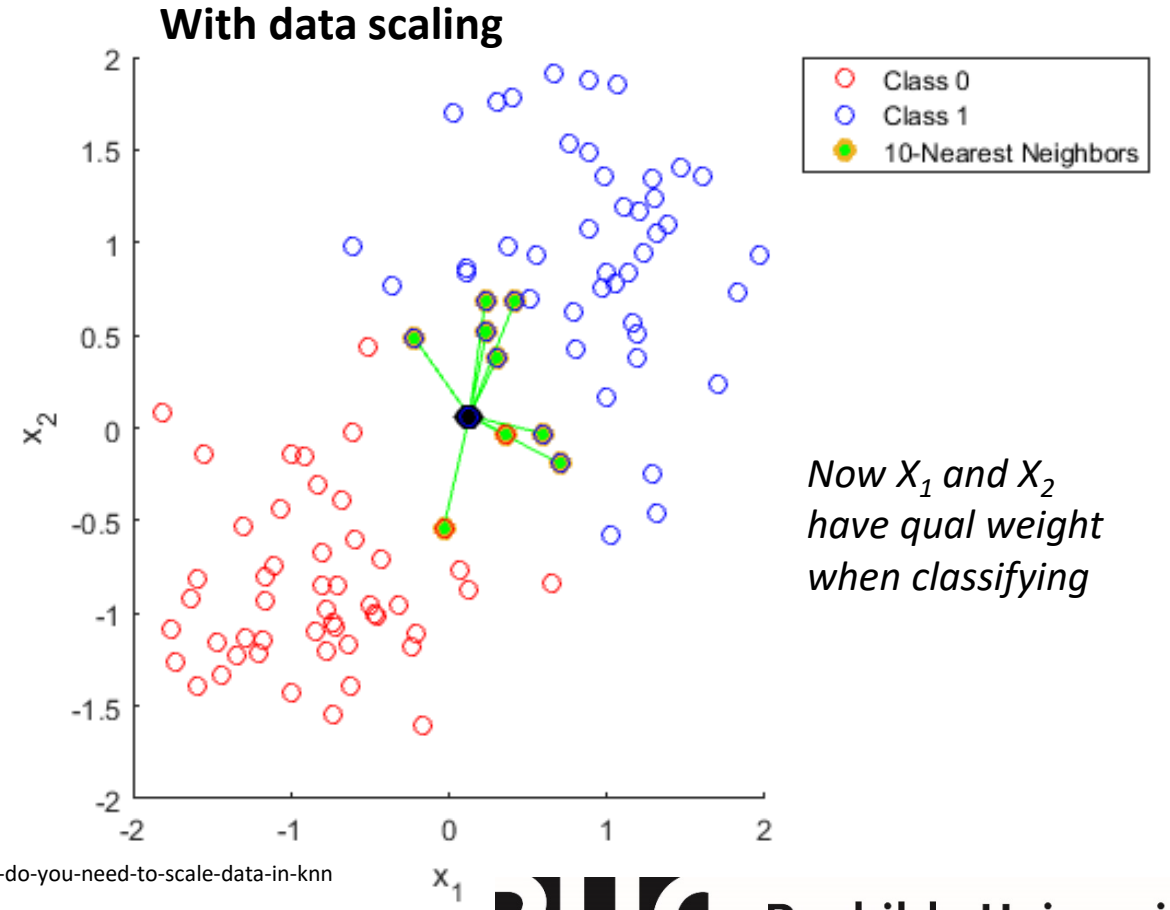
- **Pros and Cons of KNN**
  - Applicability: sample (training) data required only without training
  - High classification accuracy in many applications
  - Easy incremental adaptation to new sample objects
  - Also useful for prediction
  - Robust to noisy data by averaging K nearest neighbors
  - Naïve implementation is inefficient
    - KNN search is not straightforward. Support by database in query processing may help.
  - Does not produce explicit knowledge about classes but some explanation information.
  - Dimensionality issues:
    - Curse of dimensionality: distance becomes meaningless when there're many dimensions
    - Distance could be dominated by irrelevant attributes -> dimensionality reduction or data scaling

# K-nearest neighbors for classification (KNN)

## A motivating example of the need for scaling




*The values of $X_2$ dominates in the distance measures between point and thereby when classifying*

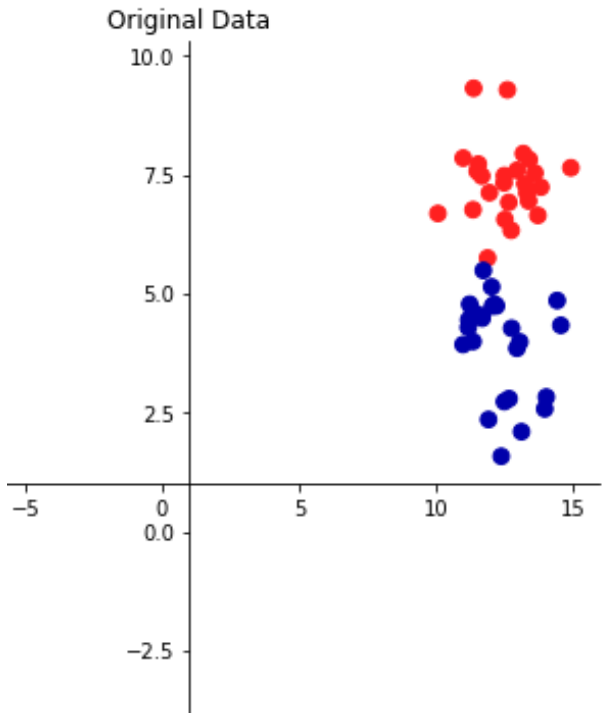



*Now $X_1$ and $X_2$ have qual weight when classifying*

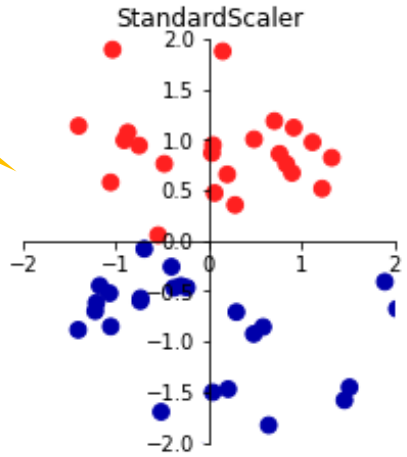
https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

# K-nearest neighbors for classification (KNN)

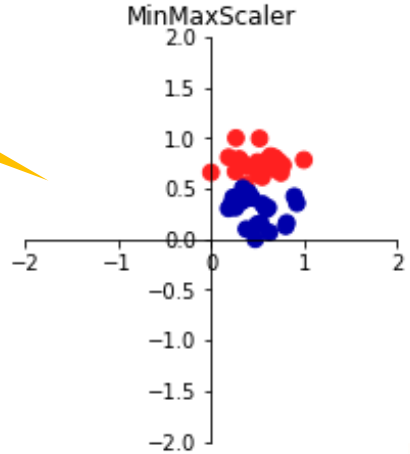
## Two types of scaling




**Standardization**




**Normalization**




- **StandardScaler**
  - For each feature: mean=0 and variance=1
- **MinMaxScaler**
  - Shifts the data, each feature falls in [0..1]

# K-nearest neighbors for classification (KNN)

- **Some notes on Data Scaling**
  - Observe and/or plot your data to see how it skews
  - Choose the right scaler you want to use
  - Apply the scaler to *both* training and testing data
    - It is easiest to apply the scaling on the whole original dataset and then split it…
    - However, strictly speaking we should train the scaler on only the training data and then apply it to the test set before running test
  - Standardization *or* Normalization? (Rule of thumb)
    - Normal data distribution: standardization; otherwise normalization
    - If uncertain: normalization; or standardization followed by normalization
    - Try different ways and decide the option with the best model performance

# K-nearest neighbors for classification (KNN)

- Let us look at an example in Python, let us look at the notebook “K-nearest neighbors.ipynb”

# Outline of this lecture

- Introduction to machine learning
- Measuring the performance of machine learning models
- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification

# Logistic regression for classification

- ***Logistic regression is a classification method*** despite its name!
  - Instead of just returning a class label as we saw for KNN, ***it returns a probability of belonging to each class*** (– in that sense it is a bit like linear regression)
- It is ***parametric*** model that in its basic form ***works for binary classification***
  - However, it is possible to extent it to work for more than two classes in several ways

# Logistic regression for classification

- **The Logistic regression model**
  - Assume we have a single feature variable X (for now numeric) and response variable Y that takes on the values 0 or 1.
  - The we want to model the probability of 0 or 1 given any particular value of X. Written as conditional probabilities, we want to model:
    - P(Y = 1|X) (and P(Y = 0 | X) = 1 - P(Y = 1|X) )
  - In logistic regression, we use the ***logistic function*** to model this probability:
    - P(Y = 1|X) = exp(a + b*X) / exp(1 + exp(a + b*X)), where exp is the exponential function
  - The logistic function will produce an s-shaped curve that is always bigger than 0 and always smaller than 1.

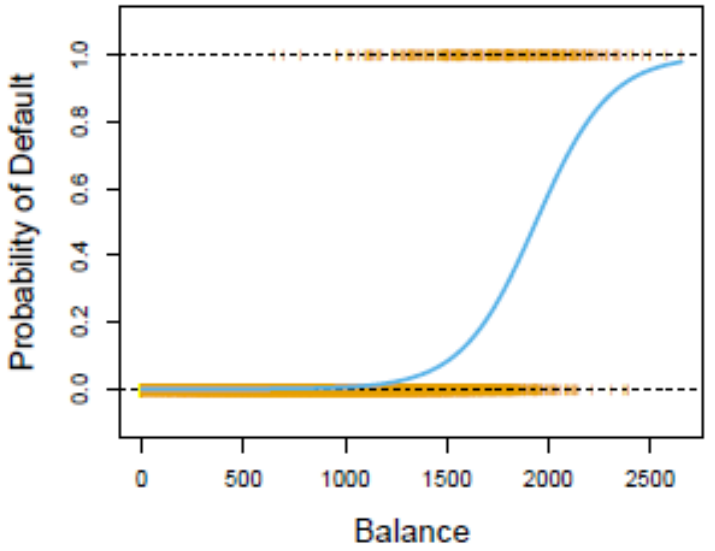



Figure 4.2 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

# Logistic regression for classification

- **The Logistic regression model**
  - The ***logistic function***:
    - P(Y = 1|X) = exp(a + b*X) / exp(1 + exp(a + b*X)), where exp is the exponential function
  - We can rewrite the formula to attain:
    - log(P(Y = 1|X) / (1 - P(Y = 1|X) ) = a + b*X
  - (In other words, the log odds is linear in X)
  - Interpretability is a bit harder than for linear regression, however:
    - If b is positive, then an increase in X, will result in an increase in P(Y = 1|X)
    - If b is negative, then an increase in X, will result in a decrease in P(Y = 1|X)
    - A unit change in X, will multiply the odds by Exp(b)

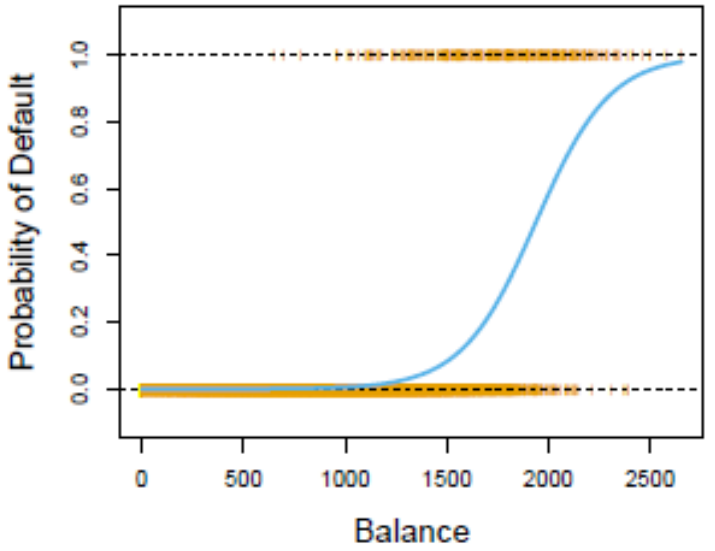



Figure 4.2 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

# Logistic regression for classification

- **The Logistic regression model**
  - The ***logistic function***:
    - P(Y = 1|X) = exp(a + b*X) / exp(1 + exp(a + b*X)), where exp is the exponential function
  - **Class assignment**
    - For logistic regression it is natural to assign the class represented by 1 whenever P(Y = 1|X) > 0.5 (and the class 0 otherwise), however, we could use other values than 0.5 if we want to be more cautious, for instance.
  - **Fitting** a logistic regression model amounts to finding optimal values for a and b.
    - The preferred method for this is something called the ***Maximum Likelihood Method***, which is beyond the scope of this course
    - We can instead use the statsmodels or scikit-learn modules in Python.

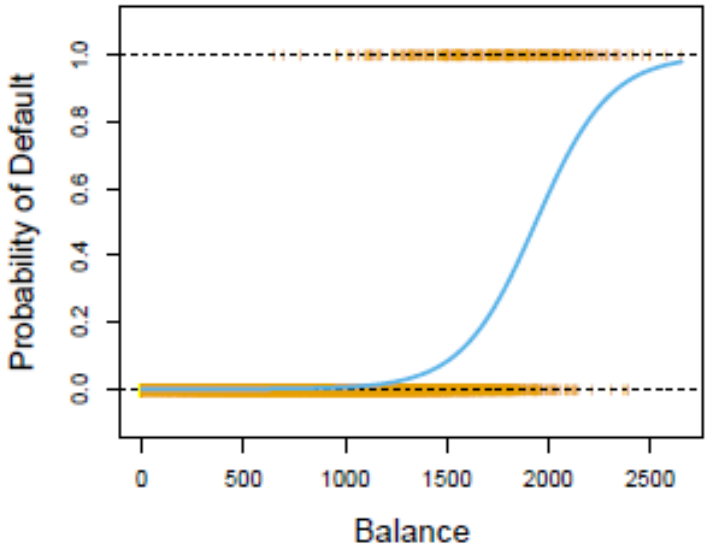



Figure 4.2 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

# Logistic regression for classification

- Let us look at an example in Python, let us look at the notebook “Logistic regression.ipynb”

# Exercises

- Do question 1 to 5 of Exercise 2 of “Exercises in Classification I.ipynb”