Data & Things
(Spring 26)

Friday February 13

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 2
2	Data transformation and exploratory data analysis	Jens	Feb 3
3	Statistics	Jens	Feb 6
4	Data Engineering	Shabab	Feb 9
5	Regression	Jens	Feb 11
6	Time series	Shabab	Feb 11
7	**Classification I**	**Jens**	**Feb 13**
8	Classification II	Jens	Feb 16
9	IoT and sensor data	Shabab	Feb 18
10	Clustering	Jens	Feb 20

Lect.	Content	Lect.	Date
11	Machine Learning Operations (MLOps)	Shabab	Feb 23
12	Introduction UCloud	Jakub/ Jens	Feb 25
13	Social Network Analysis and large data	Frederik	Feb 25
14	Neural networks and deep learning I	Jens	Feb 27
15	Neural networks and deep learning II	Jens	Mar 2
16	Generative AI	Shabab	Mar 4
17	Explainability	Shabab	Mar 6
18	Ethical reflections on data science, end of course	Jens	Mar 9
	Exan Q&A	Jens & Shabab	Mar 11

Roskilde Universitet

Outline of this lecture

- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification
- Evaluating Classification models
- Trades-offs in machine learning
    - Bias vs variance
    - Predictive accuracy vs model interpretability
- Exercises

Roskilde Universitet

Introduction to classification in general

- 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.
  - **Regression**: Predict a continuous value from a continuous range
    - E.g., predict the price of a stock or the price of a house
  - **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.

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***
  - (Note: We will treat our categorical response variable as if it was nominal in this course, however, there are also particular approaches if we insist on it being ordinal)
- ***The step of training a classification model is the same as for regression***
  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 on both 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 later

Roskilde Universitet

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) + ε** (where ε is a random error term independent of X and with mean 0)
  - Examples
    - For simple linear regression we are looking for an f of the form: f(x) = a + b*x
    - For binary classification with 0 and 1 as the class labels, **f** will always return either 0 or 1.
  - We can rarely find a f such that Y = f(X) + ε, instead, we have to settle with an f that approximately satisfy this Y ≈ f(X) + ε and ***we denote f(X) by Ŷ***.
  - 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.
    - For simple linear regression f(x) = a + b*x, a and b are that parameters that completely defines f.
    - Fitting or learning a parametric f thus reduces to finding optimal value for the parameters. Learning a non-parametric f does not involve finding optimal parameter values.
    - Today we will see example of both parametric (Logistic regression) and non-parametric (K-nearest neighbors) models

Outline of this lecture

- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification
- Evaluating Classification models
- Trades-offs in machine mearning
  - Bias vs variance
  - Predictive accuracy vs model interpretability
- Exercises

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***)

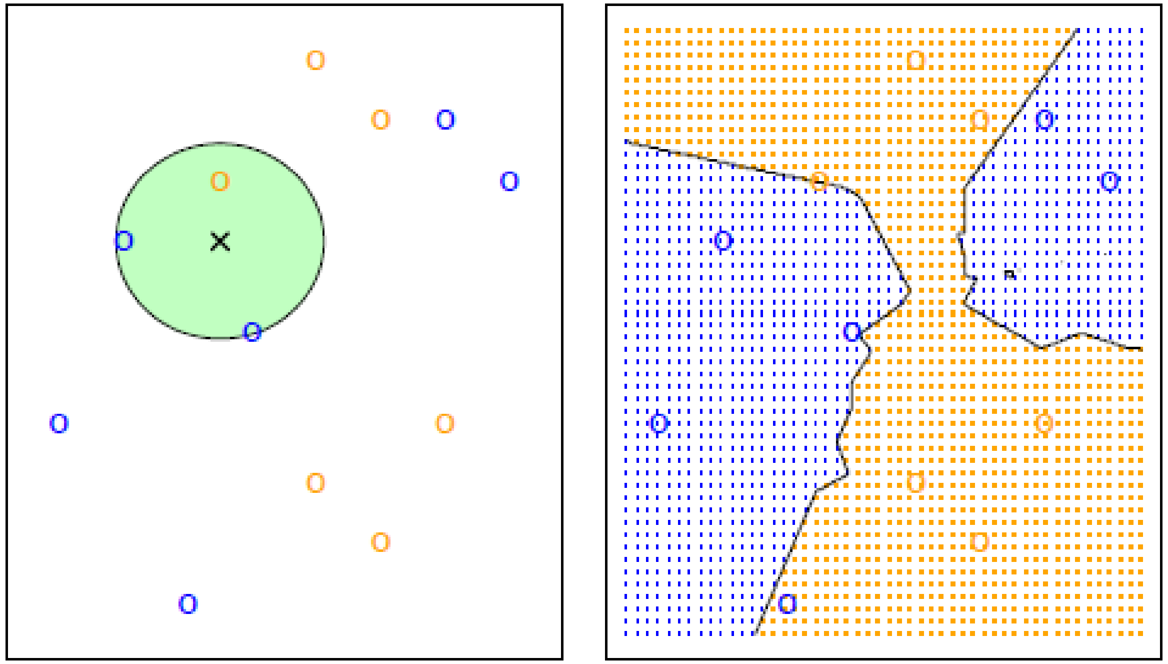


Figure 2.14 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

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***
    - ***“Nearness”*** 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 ***majority voting***

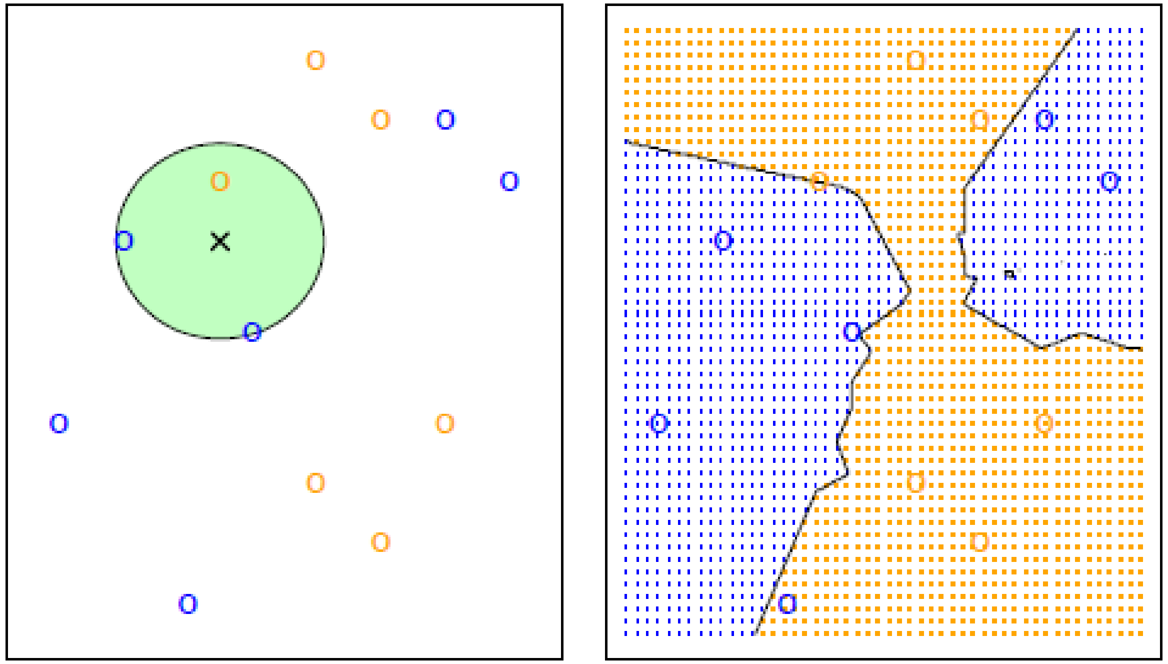
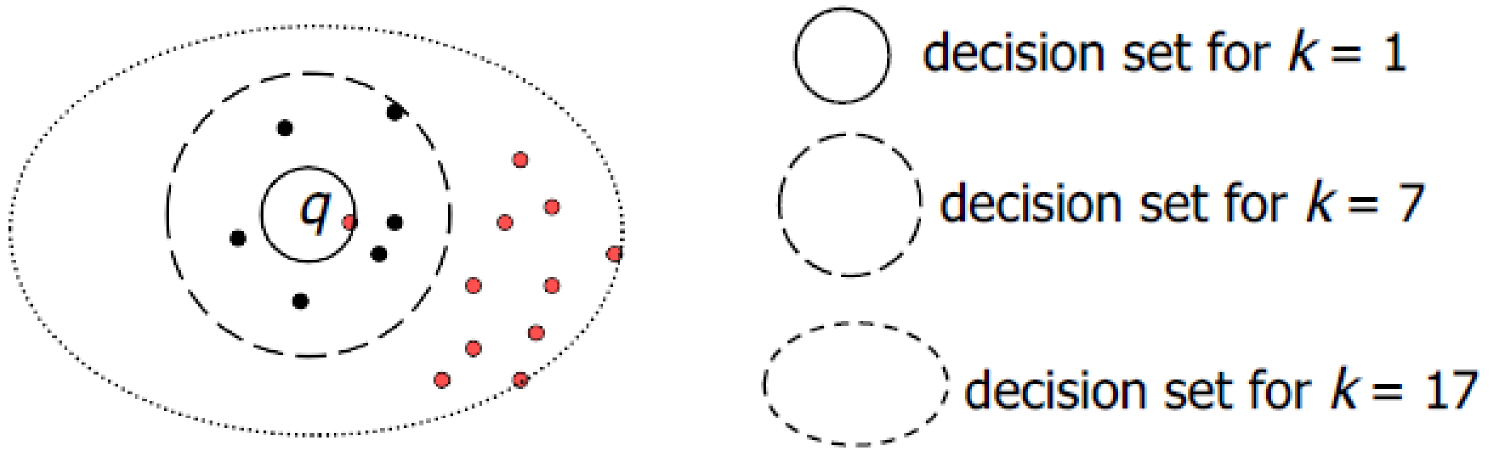


Figure 2.14 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

Roskilde Universitet

K-nearest neighbors for classification (KNN)

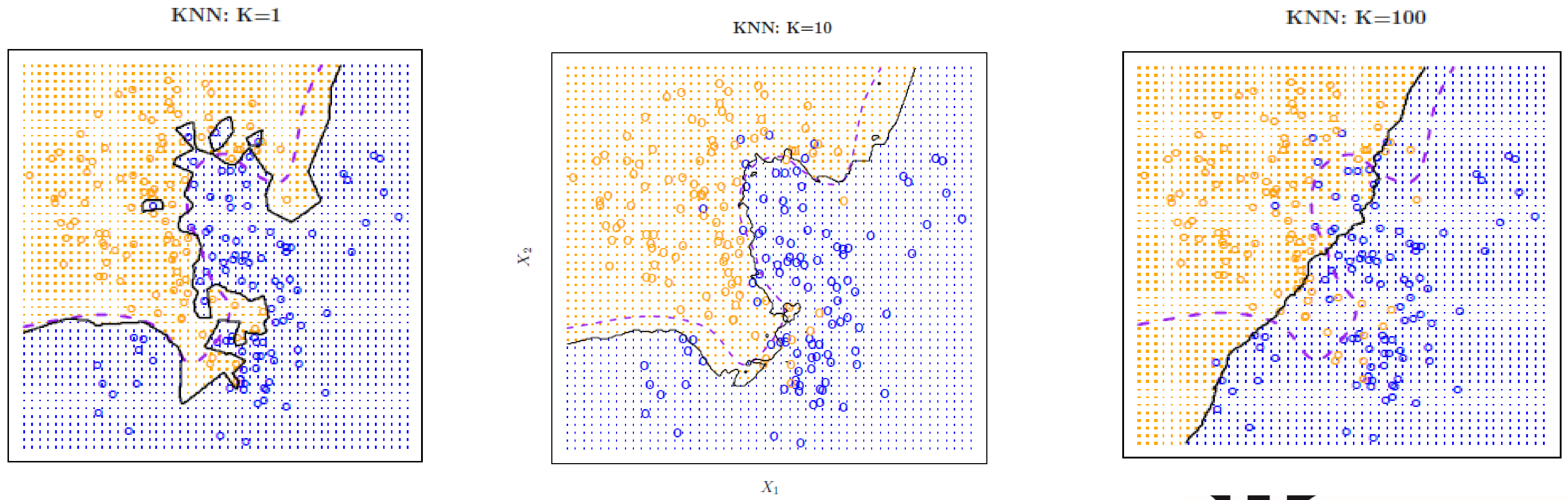
- **Appropriate Value for K**
  - *Different Ks may lead to different classification results*.
  - Too small K: High sensitivity to outliers
  - Too large K: Decision set contains many items from other classes.
  - Empirically, 1<<K<10 yields a high classification accuracy in many cases.



Roskilde Universitet

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



Roskilde Universitet

K-nearest neighbors for classification (KNN)

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

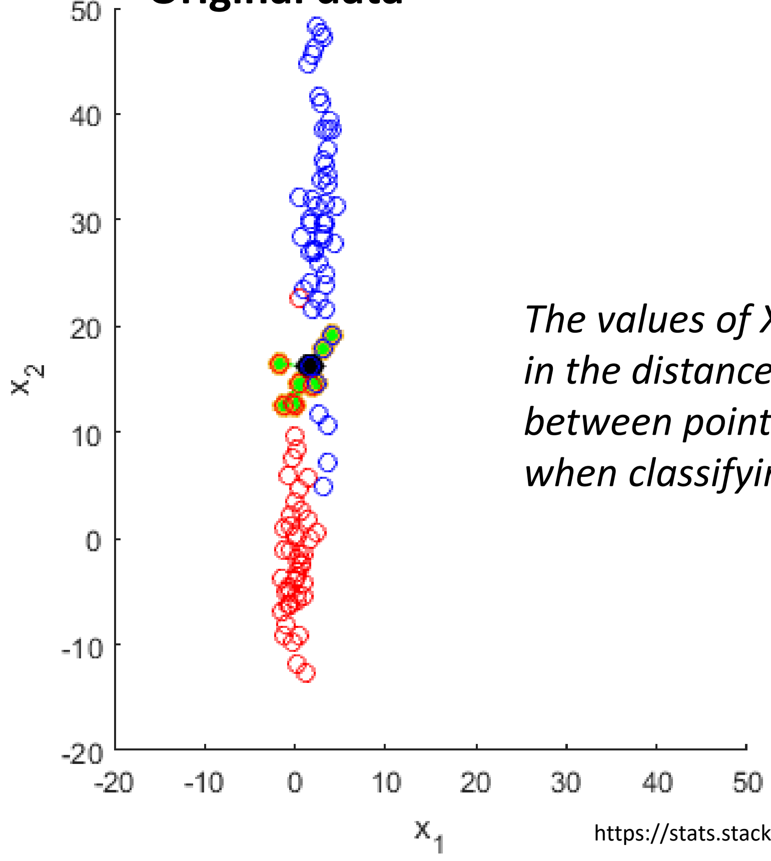
K-nearest neighbors for classification (KNN)

- **Pros and Cons of KNN**
  - High classification accuracy in many applications
  - Easy incremental adaptation to new sample objects
  - Robust to noisy data by averaging K nearest neighbors (if K is large enough)
  - 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)

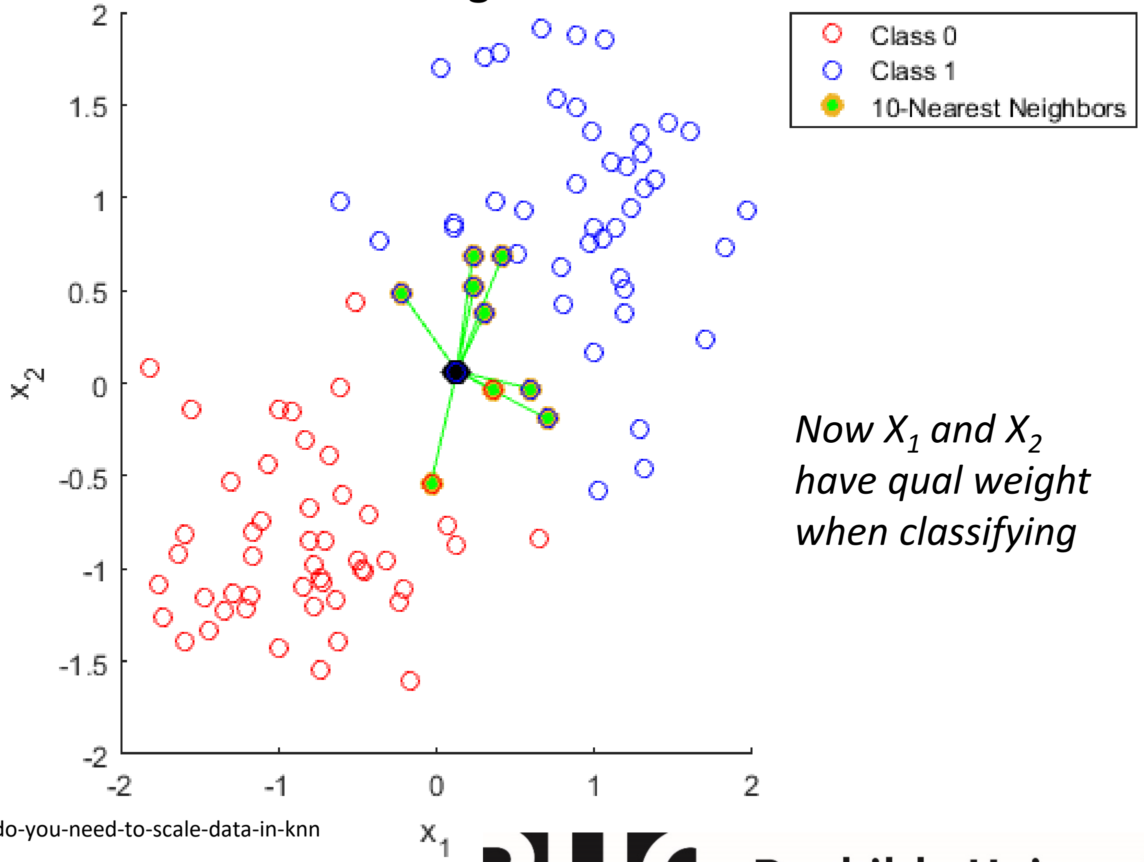
A motivating example of the need for data scaling

Original data



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

With data scaling



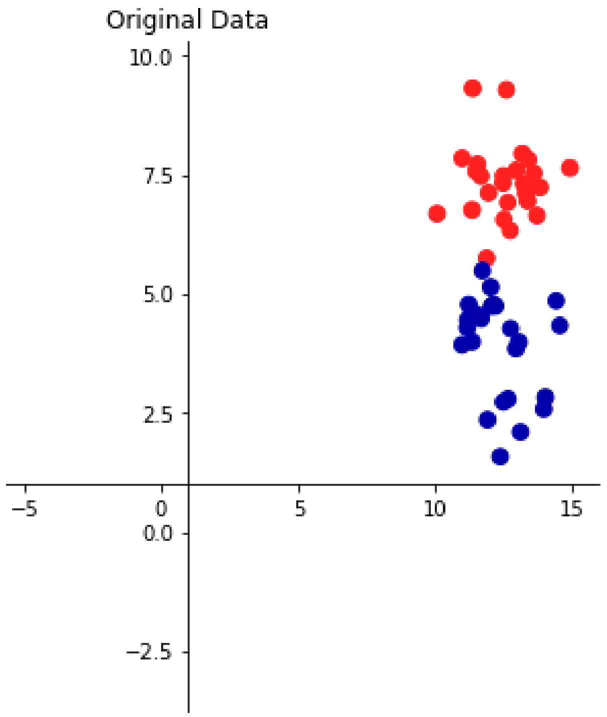
Now X_1 and X_2 have qual weight when classifying

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

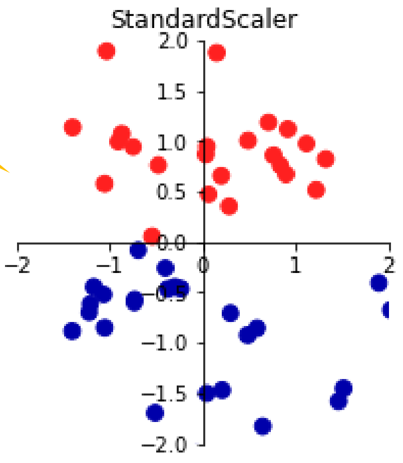
Roskilde Universitet

K-nearest neighbors for classification (KNN)

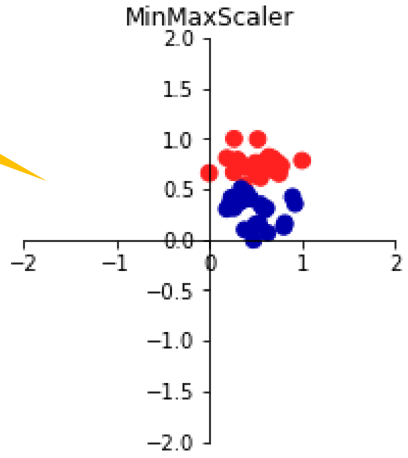
Two types of scaling



Standardization



Normalization



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

Roskilde Universitet

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
    - Do NOT apply the scaling on the whole original dataset before train-test split!
    - You should "train" the scaler on only the training data and then apply it to the test set
  - Standardization *or* Normalization? (Rule of thumb)
    - Normal data distribution: standardization; otherwise normalization
    - If uncertain: 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 classification in general
- K-nearest neighbors for classification
- Logistic regression for classification
- Evaluating Classification models
- Trades-offs in machine learning
  - Bias vs variance
  - Predictive accuracy vs model interpretability
- Exercises

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.
  - 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 then 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.

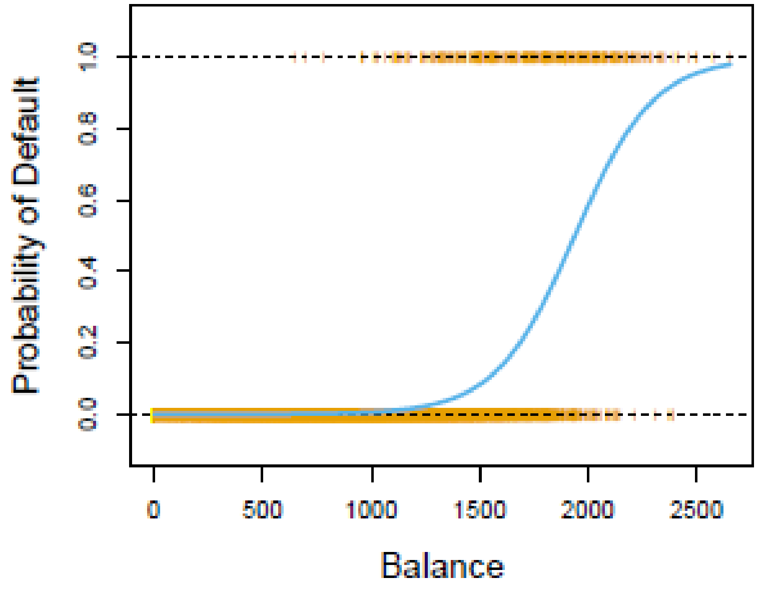


Figure 4.2 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

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)

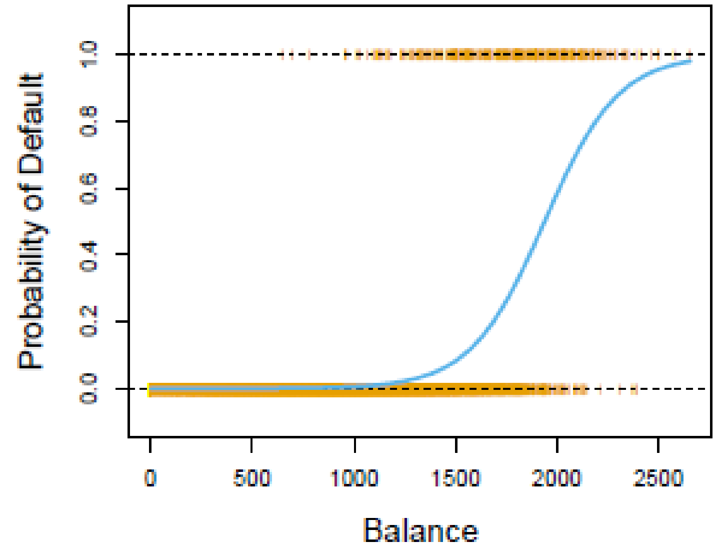


Figure 4.2 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

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)),
  - **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.

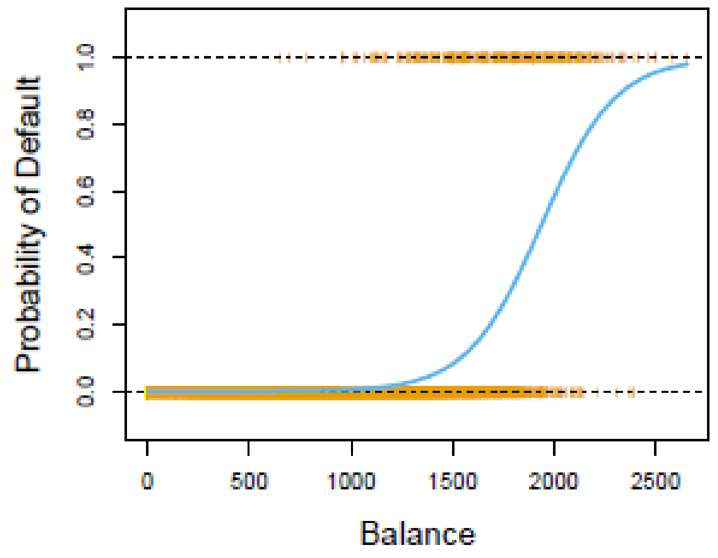


Figure 4.2 from James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. (2023). *An Introduction to Statistical Learning – with Applications in Python*. Springer

Roskilde Universitet

Logistic regression for classification

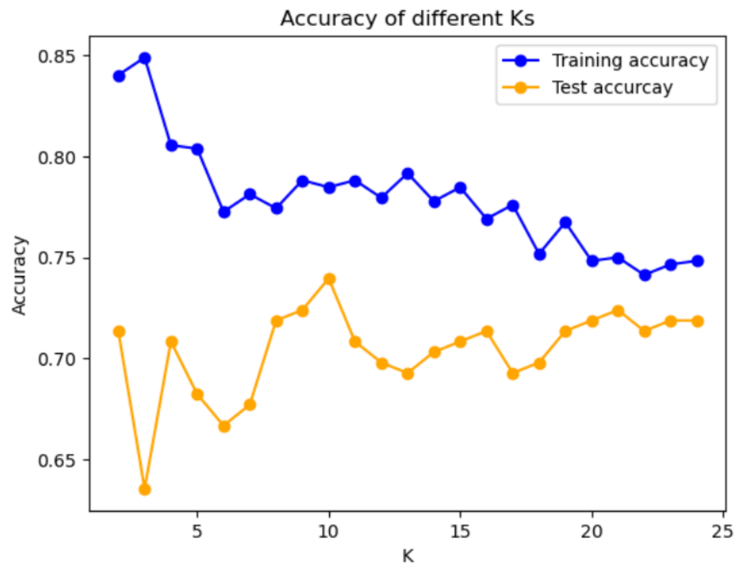
- Let us look at an example in Python, let us look at the notebook “Logistic regression.ipynb”

Outline of this lecture

- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification
- Evaluating Classification models
- Trades-offs in machine learning
  - Bias vs variance
  - Predictive accuracy vs model interpretability
- Exercises

Evaluating Classification models

- For classification models, we evaluated them by:
  - Accuracy: the fraction of correct predictions
  - Error rate: the fraction of incorrect predictions
- However, not all errors are equally important:
  - In medical diagnostics (1 = having the disease, 0 = not having it) two type of errors:
    - Predicted 1, but truly 0: Bad in the sense that it is not nice to be diagnosed with a disease, but likely later tests will likely be taken revealing that the patient did not have the disease after all.
    - Predicted 0, but truly 1: Much worse, the patient is led to believe she does not have the disease, and often no further tests are made, which might lead to severe health conditions later.
  - In spam email detection (1 = it is spam, 0 = it is not spam) two types errors:
    - Predicted 1, but truly 0: Problematic if an actual mail is deleted as being spam.
    - Predicted 0, but truly 1: A little annoying to have a spam mail arrive in your inbox, but usually not that big of a problem problem



Roskilde Universitet

Evaluating Classification models

- **The confusion matrix** (binary case)
  - Display predicted class versus actual class
  - **Accuracy:**
    - (TP + TN) / (TP + FN + FP + TN)
    - The fraction of the time the classifier makes the correct classification

	Predicted 1 (positive) (ex. spam)	Predicted 0 (negative) (ex. not spam)
Actual 1 (positive) (ex. spam)	TP (True Positive)	FN (False Negative)
Actual 0 (negative) (ex. not spam)	FP (False Positive)	TN (True Negative)

Roskilde Universitet

Evaluating Classification models

- **The confusion matrix** (binary case)
  - Display predicted class versus actual class
  - Accuracy:
    - (TP + TN) / (TP + FN + FP + TN)
    - The fraction of the time the classifier makes the correct classification
  - **Precision** (/positive predictive value):
    - TP / (TP + FP)
    - The fraction of the positive classifications that is truly positive
    - If this is high, we have few real emails that are classified as spam

	Predicted 1 (positive) (ex. spam)	**Predicted 0 (negative) (ex. not spam)**
Actual 1 (positive) (ex. spam)	**TP** (True Positive)	**FN** (False Negative)
Actual 0 (negative) (ex. not spam)	**FP** (False Positive)	**TN** (True Negative)

Evaluating Classification models

- **The confusion matrix** (binary case)
  - Display predicted class versus actual class
  - Accuracy:
    - (TP + TN) / (TP + FN + FP + TN)
    - The fraction of the time the classifier makes the correct classification
  - Precision (/positive predictive value):
    - TP / (TP + FP)
    - The fraction of the positive classifications that is truly positive
    - If this is high, we have few real emails that are classified as spam
  - **Recall** (Sensitivity/true positive rate):
    - TP / (TP + FN)
    - The fraction of the actual positive cases that the classified detect as positive
    - If this is high, we have few cases of the disease that are not detected

	Predicted 1 (positive) (ex. spam)	**Predicted 0 (negative) (ex. not spam)**
Actual 1 (positive) (ex. spam)	**TP** (True Positive)	**FN** (False Negative)
Actual 0 (negative) (ex. not spam)	**FP** (False Positive)	**TN** (True Negative)

Evaluating Classification models

- **The confusion matrix** (binary case)
  - Display predicted class versus actual class
  - Accuracy:
    - (TP + TN) / (TP + FN + FP + TN)
    - The fraction of the time the classifier makes the correct classification
  - Precision (/positive predictive value):
    - TP / (TP + FP)
    - The fraction of the positive classifications that is truly positive
    - If this is high, we have few real emails that are classified as spam
  - Recall (Sensitivity/true positive rate):
    - TP / (TP + FN)
    - The fraction of the actual positive cases that the classified detect as positive
    - If this is high, we have few cases of the disease that are not detected
  - **F1 score**:
    - (2*Precision*Recall) / (Precision+Recall)

	Predicted 1 (positive) (ex. spam)	**Predicted 0 (negative) (ex. not spam)**
Actual 1 (positive) (ex. spam)	**TP** (True Positive)	**FN** (False Negative)
Actual 0 (negative) (ex. not spam)	**FP** (False Positive)	**TN** (True Negative)

Evaluating Classification models

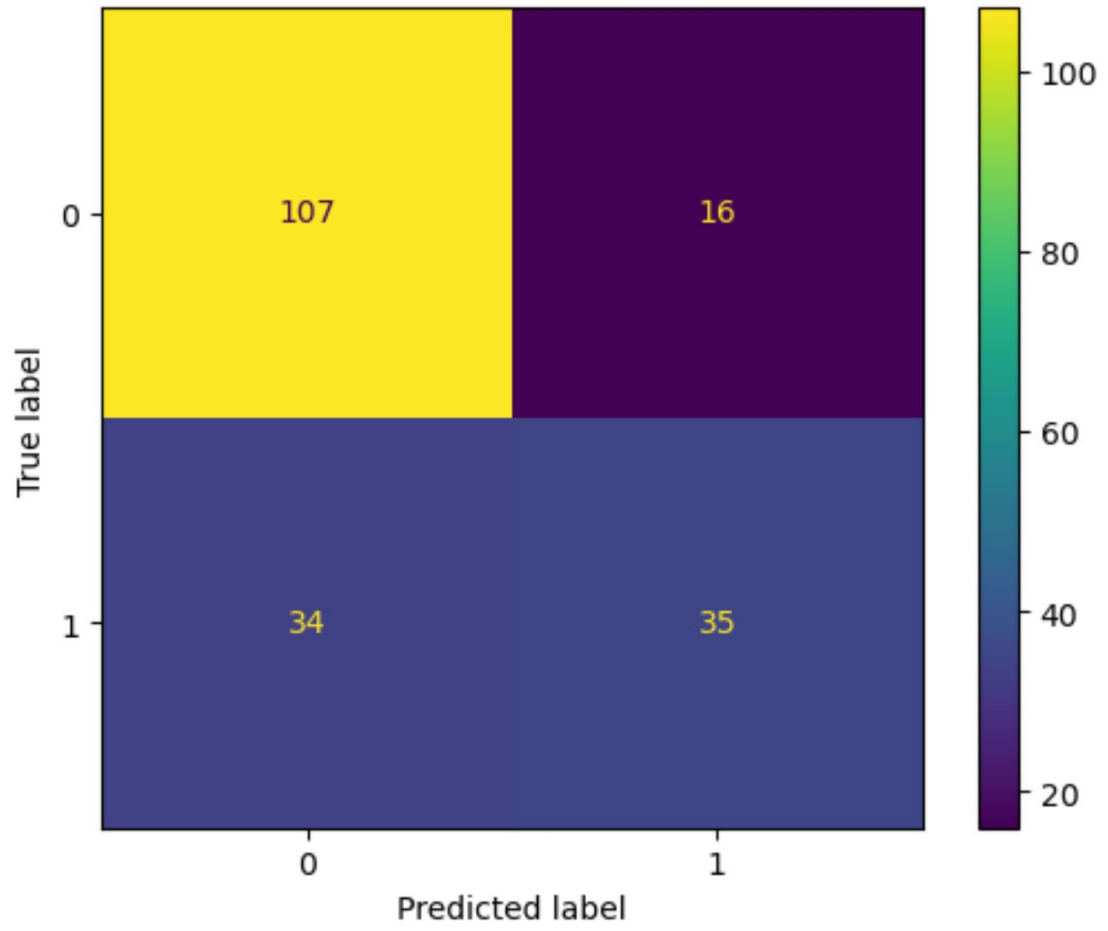
- **Other measures:**
  - **False positive rate**:
    - FP / (FP+ TN)
  - **False negative rate**:
    - FN / (FN+ TP)
  - **Specificity (/true negative rate)**:
    - TN / (TN + FP)
  - **Negative predictive value**:
    - TN / (TN + FN)
  - ….

	Predicted 1 (positive) (ex. spam)	**Predicted 0 (negative) (ex. not spam)**
Actual 1 (positive) (ex. spam)	**TP** (True Positive)	**FN** (False Negative)
Actual 0 (negative) (ex. not spam)	**FP** (False Positive)	**TN** (True Negative)

Evaluating Classification models

- Example: The KNN model (K=5) for the diabetes dataset from last time

K	Accuracy	Precision	Recall	F1
5	0.682292	0.557143	0.565217	0.561151



Roskilde Universitet

Evaluating Classification models

- **Beyond the binary Confusion Matrix**
  - Three pre-defined classes A, B, C
  - Ground truth and classification result

Object	Ground Truth	Classification Result
object-1	*A*	A
object-2	*B*	A
object-3	*C*	C
object-4	*C*	C
object-5	*B*	B
object-6	*A*	B
object-7	*B*	B

- Confusion Matrix (N x N for N classes)
  - The cell M[*i*, *j*] counts the cases with groundtruth *i* that are classified as *j*

Classification result

FN for A

Ground truth

	A	B	C	Total
A	1	1	0	2
B	1	2	0	3
C	0	0	2	2
Total	2	3	2	7

FP for A

TP for one class, and TN for others

- **Accuracy** = (TP+TN) /All
  - In this example, Accuracy = (1+2+2)/7 = 71.4%

Roskilde Universitet

Evaluating Classification models

- **Precision and Recall in general**
  - Calculate the precision p_i for each class C_i
    - Overall precision is the *average* of all p_i's
  - Calculate the recall r_i for each class C_i
    - Overall recall is the *average* of all r_i's
  - Example:
    - p_A=30/60=1/2, r_A=30/100=3/10
    - p_B=60/120=1/2, r_B=60/100=3/5
    - p_C=80/120=2/3, r_C=80/100=4/5
    - Overall precision = 5/9,
      overall recall = 17/30

Confusion matrix

		Classification result			
		A	**B**	**C**	**Total**
Ground truth	A	30	50	20	100
	B	20	60	20	100
	C	10	10	80	100
	Total	60	120	120	300

Recall for A: r_A

Precision for A: p_A

Evaluating Classification models

- **Analyze Your Confusion Matrix**
  - Essentially, the more zeroes or smaller the numbers on all cells but the diagonal, the better a classifier is. So, you may analyze your confusion matrix and tweak your features accordingly.
  - Confusion matrix gives strong clues as to where your classifier is going wrong.
    - E.g., if for Class A you can see that the classifier incorrectly predicts Class B for majority of the mislabeled cases, it indicates the classifier is somehow confused between classes A and B.
    - One way to fix this is to add discriminating features to improve classification of class A, e.g., more training data of A.

Evaluating Classification models

- **Classification with class probabilities**
  - All the evaluation metrics we have seen so far are evaluating classification models at set decision threshold (-we have set it)
  - However, as we saw for logistic regression last time, some classification algorithms give us class probabilities instead of hard class labels
  - But how do we choose the right threshold?
  - Can we evaluate a model irrespective of what threshold we chose?

Evaluating Classification models

- **Prediction probabilities** (like in logistic regression).
  - A probability of class membership is assigned instead of a label
  - A threshold can be used to control how to decide the predicted class label.

- Different thresholds lead to different metric values.
- This requires us to generate different confusion matrixes ☹

ID	Actual	Prediction Probability	>0.6	>0.7	> 0.8	Metric
1	0	0.98	1	1	1	
2	1	0.67	1	0	0	
3	1	0.58	0	0	0	
4	0	0.78	1	1	0	
5	1	0.85	1	1	1	
6	0	0.86	1	1	1	
7	0	0.79	1	1	0	
8	0	0.89	1	1	1	
9	1	0.82	1	1	1	
10	0	0.86	1	1	1	
			0.75	0.5	0.5	TPR
			1	1	0.66	FPR
			0	0	0.33	TNR
			0.25	0.5	0.5	FNR

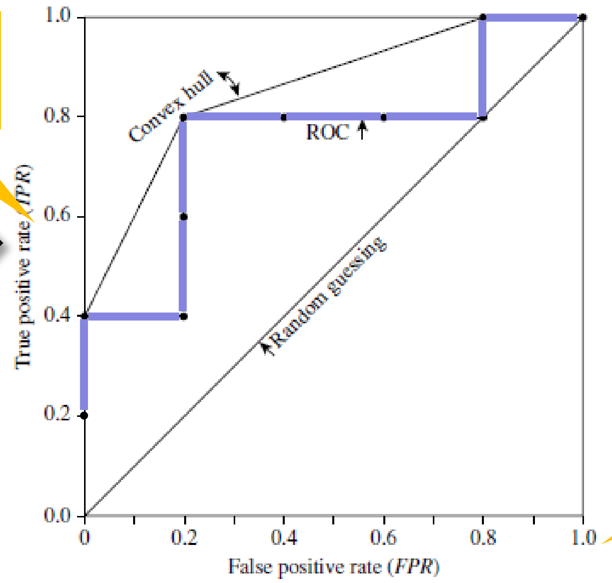
For positive label

Evaluating Classification models

- **Receiver Operating Characteristics (ROC)** curves: for visual comparison of binary classifiers
  - Rank your classification results in *descending* order of prediction probabilities
  - Calculate TPR and FPR for each current tuple in the ranked order
  - Mark each (FPR, TPR) point on the graph.
  - Connect all such points using a *convex hull*
- **NB**: TP, FP, TN, FN (and ***TPR*** and ***FPR***) change as you seen more tuples in classification result

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

Sensitivity/ recall

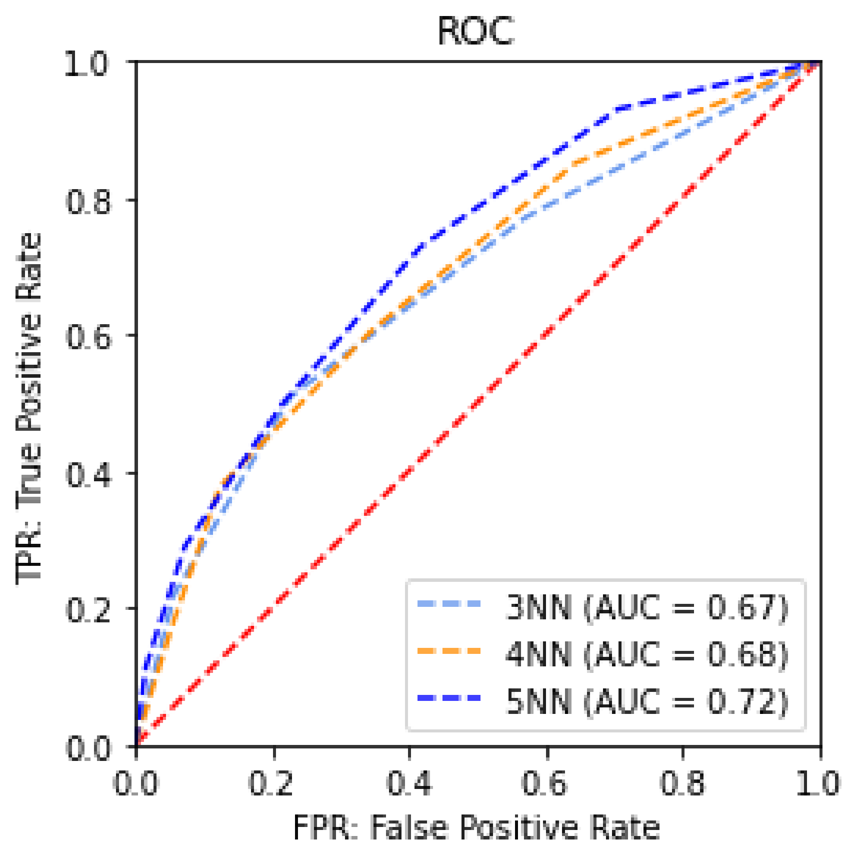


1- specificity

Roskilde Universitet

Evaluating Classification models

- ROC Curves and AUC
  - A ROC curve shows the trade-off between the True Positive Rate and the False Positive Rate
  - The diagonal represents *random guessing*
  - The *area under the ROC curve* (**AUC**) is a measure of the "accuracy" of the model
  - The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model
  - A model with perfect accuracy will have an area of 1.0



Roskilde Universitet

Evaluating Classification models revisited

- **Examples**
  - Let us look at the notebook “Evaluating classification models.ipynb”.

Outline of this lecture

- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification
- Measuring the performance of machine learning models
- Trades-offs in machine learning
  - Bias vs variance
  - Predictive accuracy vs model interpretability
- Exercises

Trades-offs in machine learning

- **Estimation of f and prediction errors**
  - 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)
  - Given such 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 X.
    - 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

Trades-offs in machine learning

- Overfitting: A model works well on the training data but generalizes poorly to unseen data – much better evaluation scores on the training set than on the test set. Can be due to:
  - 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 – poor evaluation scores on the training set. Can be due to:
  - Too few features are used in training.
  - The model type used is too simple.
  - Training data is too little, failing to contain sufficient variance.

Evaluations of regression 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/

Roskilde Universitet

Trades-offs in machine learning

- **Bias** and **Variance**
  - 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 how 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 (***underfitting***). 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!

Trades-offs in machine learning

- **Bias** and **Variance**
  - The green curve has the highest variance and is overfitting.
    - Seen by the low training MSE and high test MSE
  - The orange curve has the highest bias and is underfitting.
    - Seen by both high training MSE and high test MSE
  - The blue curve seems to be the best fit, because of it' low test MSE
    - It best balance the bias-variance trade-of
    - It's test MSE get closest to the irreducible error represented by the dashed line

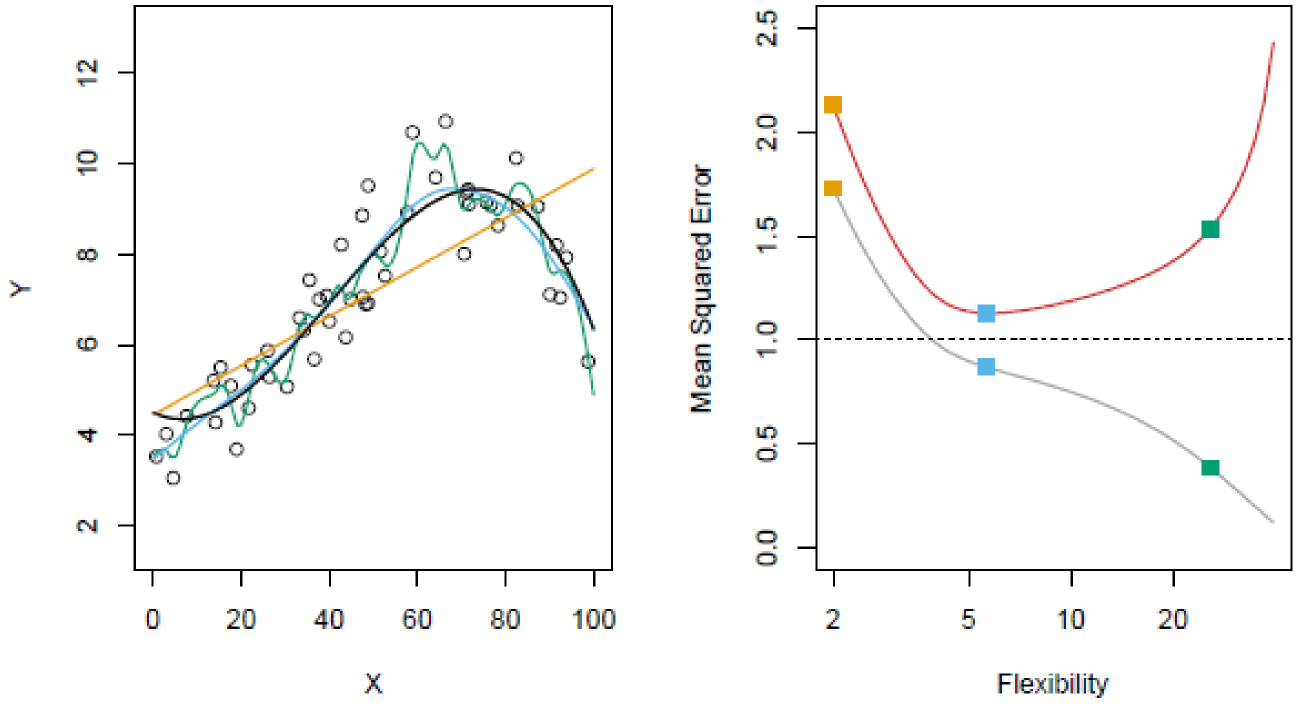


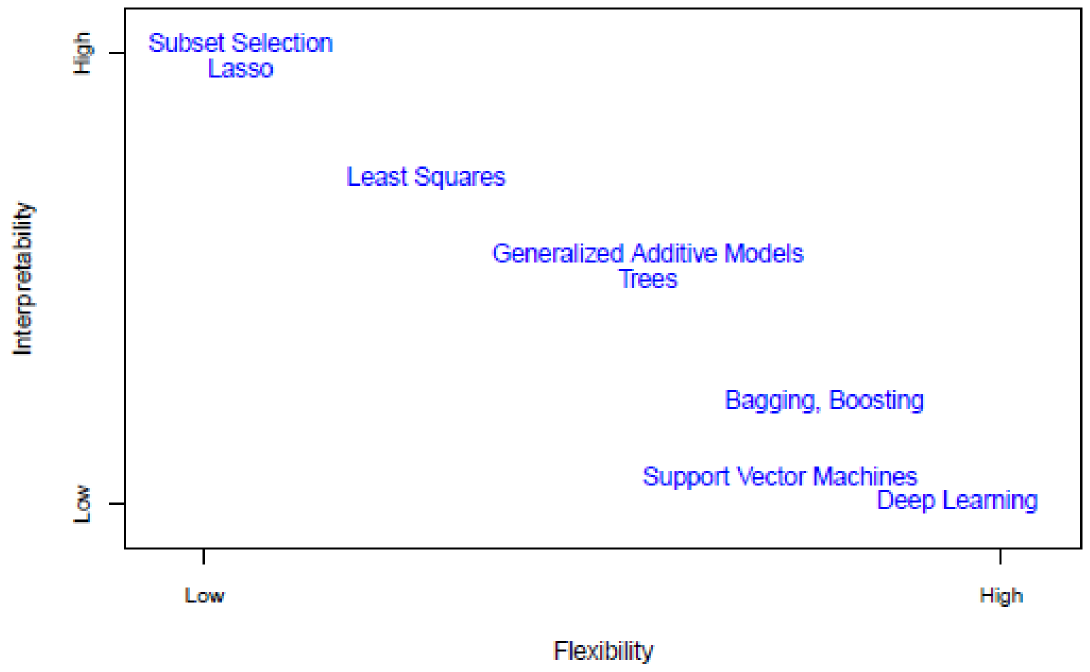
FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed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

Trades-offs in machine learning

- **Prediction vs. Inference** – Once we have a supervised machine learning model f, 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!

Trades-offs in 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, for instance if they are ***overfitting***



Roskilde Universitet

Outline of this lecture

- Introduction to classification in general
- K-nearest neighbors for classification
- Logistic regression for classification
- Measuring the performance of machine learning models
- Trades-offs in machine learning
    - Bias vs variance
    - Predictive accuracy vs model interpretability
- Exercises

Exercises

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