Chapter 07

Classification

Dr. Steffen Herbold

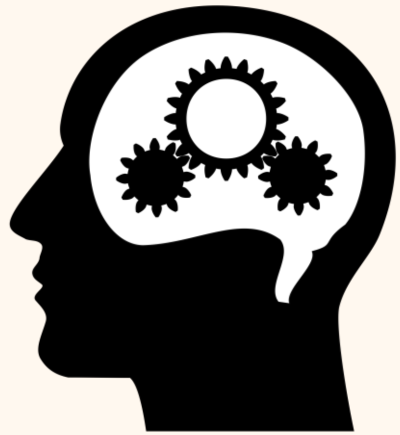
herbold@cs.uni-goettingen.de

Outline

- Overview
- Classification Models
- Comparison of Classification Models
- Summary

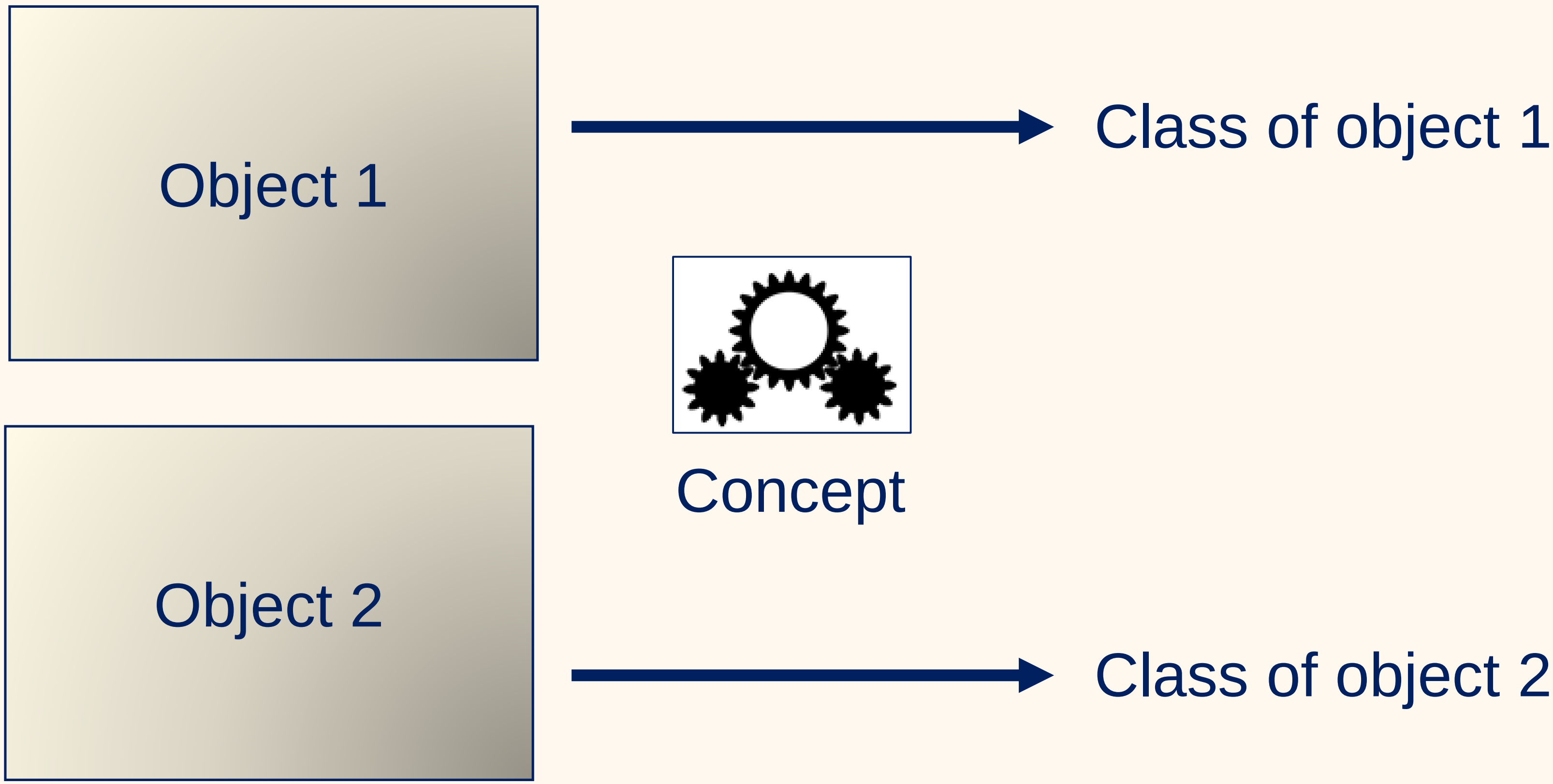
Example of Classification

This is a whale



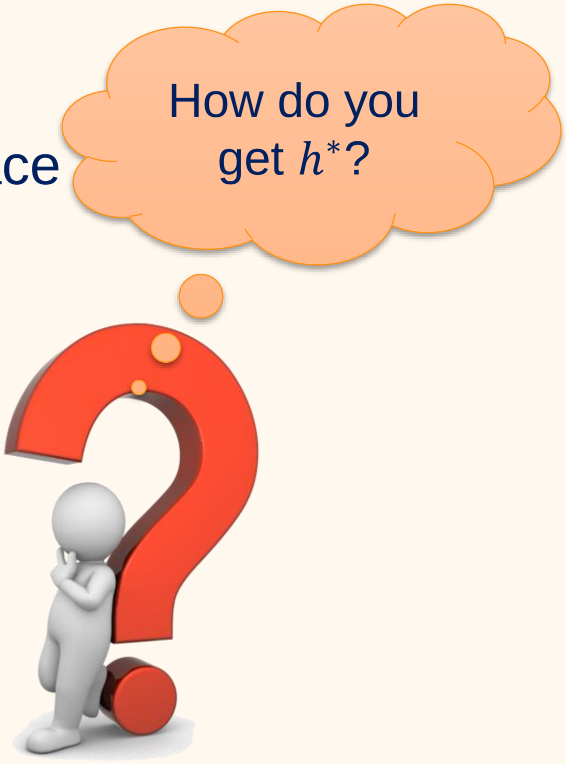
This is a bear

The General Problem



The Formal Problem

- Object space
  - $O = \{object_1, object_2, \dots\}$
  - Often infinite
- Representations of the objects in a feature space
  - $\mathcal{F} = \{\phi(o), o \in O\}$
- Set of classes
  - $C = \{class_1, \dots, class_n\}$
- A *target concept* that maps objects to classes
  - $h^*: O \rightarrow C$
- Classification
  - Finding an approximation of the target concept



The „Whale“ Hypothesis

- Why do we know this is a whale?

Has a fin

Blue background

Oval body

Black top, white bottom

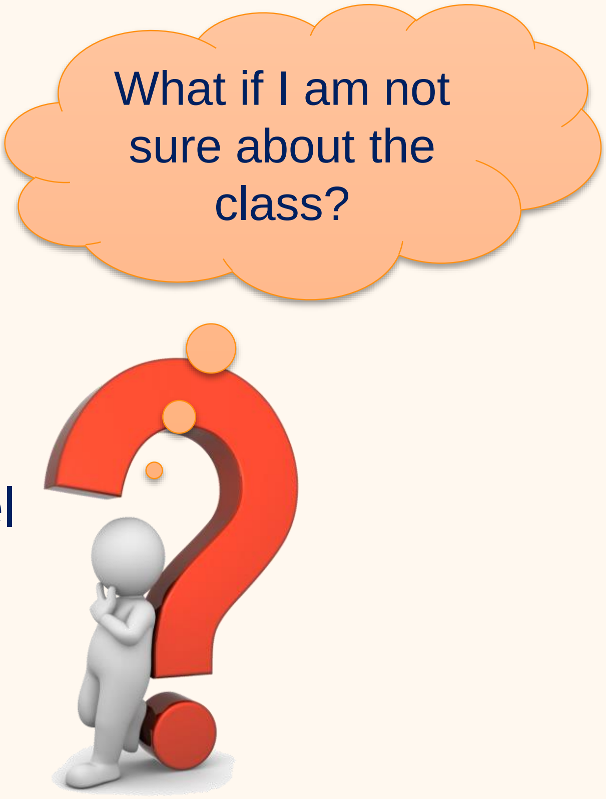
Hypothesis: Objects with fins, an oval general shape that are black on top and white on the bottom in front of a blue background are whales.

The Hypothesis

- A hypothesis maps features to classes
  - $h: \mathcal{F} \rightarrow C$
  - $h: \phi(o) \rightarrow C$

- Approximation of the target concept h^*
  - $h^*(o) \approx h(\phi(o))$

- Hypothesis = Classifier = Classification Model



Classification using Scores

- A numeric score for each class $c \in C$
- Often a probability distribution
  - $h': \phi(o) \rightarrow [0,1]^{|C|}$
  - $\|h'(\phi(o))\|_1 = 1$
- Example
  - Three classes: „whale“, „bear“, „other“
  - $h'\left(\phi(\text{"whalepicture"})\right) = (0.7, 0.1, 0.2)$

score whale

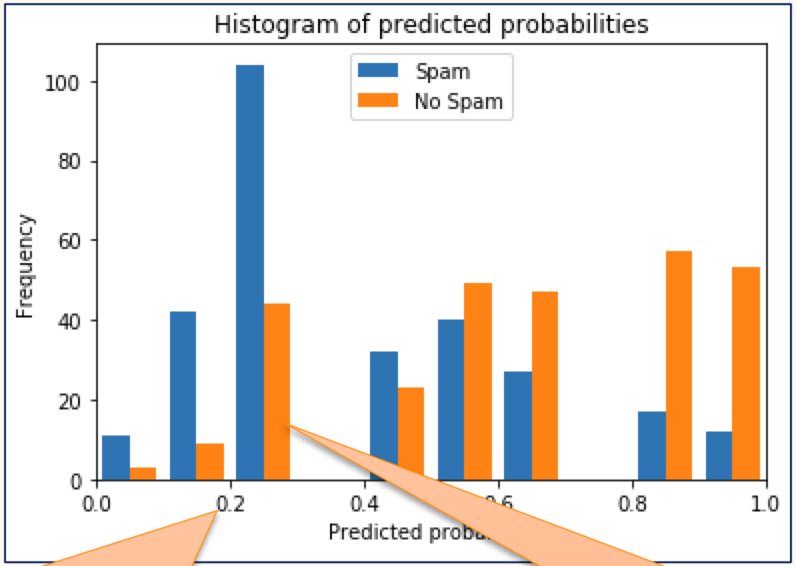
score bear

score other

- Standard approach:
  - Classification is class with highest score

Thresholds for Scores

- Different thresholds also possible



Threshold of 0.2 would miss “Spam” but better identify “No Spam”

Many “No Spam” incorrectly detected as spam if “highest” score is used

Quality of Hypothesis

How do you evaluate
$h^*(o) \approx h(\phi(o))$

- Goal: Approximation of the target concept
  - $h^*(o) \approx h(\phi(o))$

→ Use Test Data
  - Structure is the same as training data
  - Apply hypothesis

$\boldsymbol{\phi(o)}$					$\boldsymbol{h^*(o)}$	$\boldsymbol{h(\phi(o))}$
hasFin	shape	colorTop	colorBottom	background	class	prediction
true	oval	black	black	blue	**whale**	**whale**
false	rectangle	brown	brown	green	**bear**	**whale**
…	…	…	…	…	…	

The Confusion Matrix

- Table of actual values versus prediction

Actual class

Predicted Class

	whale	bear	other
whale	29	1	3
bear	2	22	13
other	4	11	51

Two whales were incorrectly predicted as bears

Binary Classification

- Many problems are binary
  - Will I get my money back?
  - Is this credit card fraud?
  - Will my paper be accepted?
  - …

- Can all be formulated as either being in a class or not

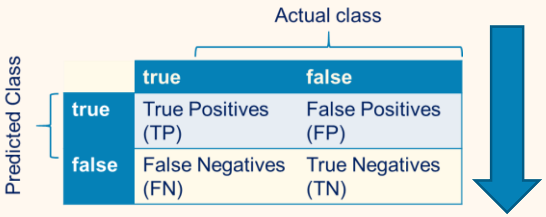
→Labels *true* and *false*

The Binary Confusion Matrix

		Actual class	
		true	**false**
Predicted Class	**true**	True Positives (TP)	False Positives (FP)
	false	False Negatives (FN)	True Negatives (TN)

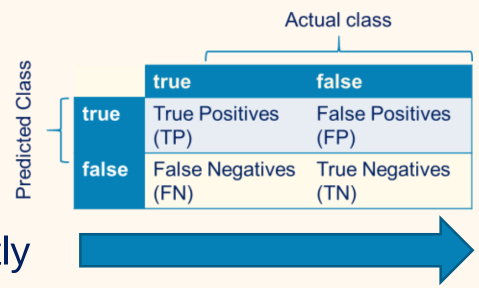
- False positives are also called Type I error
- False negatives are also called Type II error

Binary Performance Metrics (1)



- Rates per actual class
  - True positive rate, recall, sensitivity
    - Percentage of actually „True“ that is predicted correctly
    - $TPR = \frac{TP}{TP+FN}$
  - True negative rate, specificity
    - Percentage of actually „False“ that is predicted correctly
    - $TNR = \frac{TN}{TN+FP}$
  - False negative rate
    - Percentage of actually „True“ that is predicted wrongly
    - $FNR = \frac{FN}{FN+TP}$
  - False positive rate
    - Percentage of actually „False“ that is predicted wrongly
    - $FPR = \frac{FP}{FP+TN}$

Binary Performance Metrics (2)



- Rates per predicted class
  - Positive predictive value, precision
    - Percentage of predicted „True“ that is predicted correctly
    - $PPV = \frac{TP}{TP+FP}$
  - Negative predictive value
    - Percentage of predicted „False“ that is predicted correctly
    - $NPV = \frac{TN}{TN+FN}$
  - False discovery rate
    - Percentage of predicted „True“ that is predicted wrongly
    - $FDR = \frac{FP}{TP+FP}$
  - False omission rate
    - Percentage of predicted „False“ that is predicted wrongly
    - $FOR = \frac{FN}{FN+TN}$

Binary Performance Metrics (3)

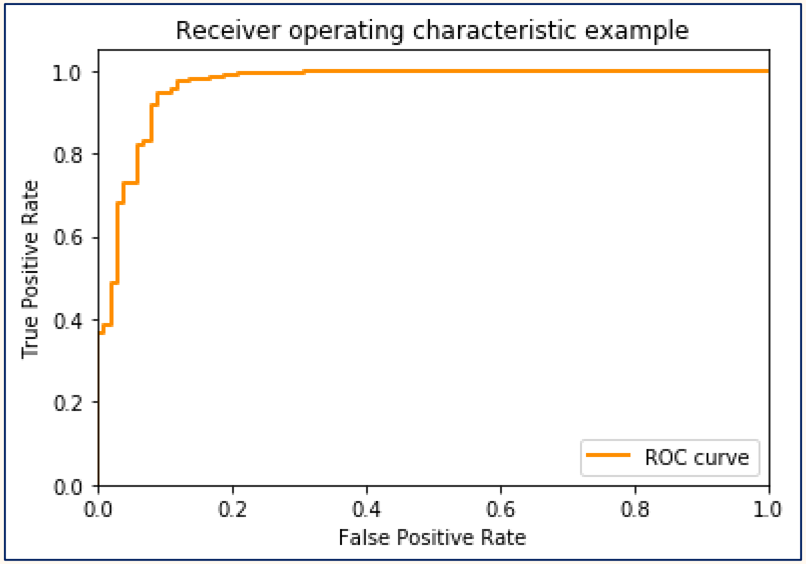
Actual class

Predicted Class	true	false
true	True Positives (TP)	False Positives (FP)
false	False Negatives (FN)	True Negatives (TN)

- Metrics that take „everything“ into account
  - Accuracy
    - Percentage of data that is predicted correctly
    - $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
  - F1 measure
    - Harmonic mean of precision and recall
    - $F_1 = 2\frac{precision \times recall}{precision+recall}$
  - Matthews correlation coefficient (MCC)
    - Chi-squared correlation between prediction and actual values
    - $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Receiver Operator Characteristics (ROC)

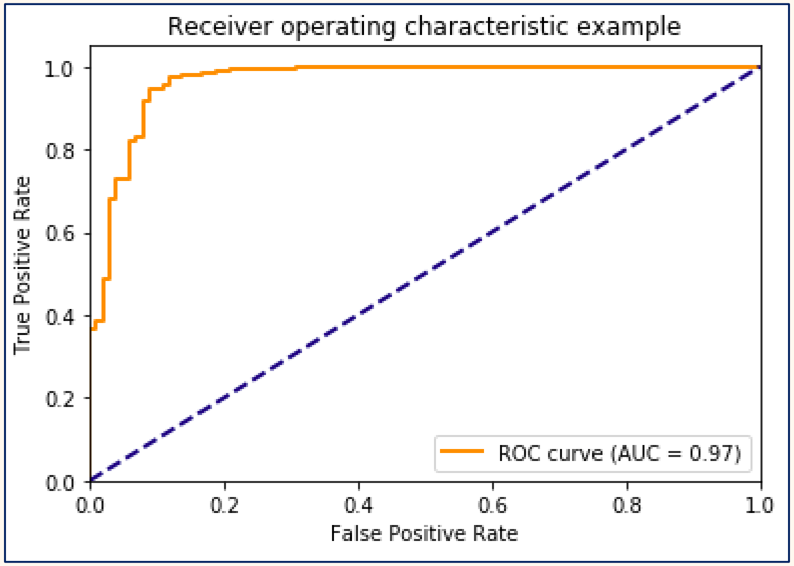
- Plot of true positive rate (TPR) versus false positive rate (FPR)
- Different TPR/FPR values possible due to thresholds for scores



Area Under the Curve (AUC)

- Large Area = Good Performance

- Accounts for tradeoffs between TPR and FPR



Micro and Macro Averaging

- Metrics not directly applicable for more than two classes
  - Accuracy is the exception
- Micro Averaging
  - Expand formulas to use individual positive, negative examples for each class
- Macro Averaging
  - Assume one class as true, combine all other as false
  - Compute metrics for all such combinations
  - Take average
- Example for the true positive rate:
  - $TPR_{micro} = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C} TP_c + \sum_{c \in C} FN_c}$
  - $TPR_{macro} = \frac{\sum_{c \in C} \frac{TP_c}{TP_c + TN_c}}{|C|}$

Outline

- Overview
- **Classification Models**
- Comparison of Classification Models
- Summary

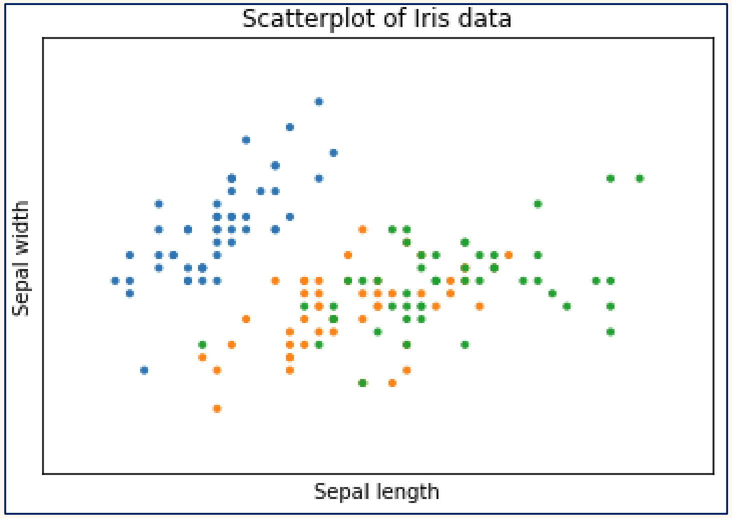
Overview of Classifiers

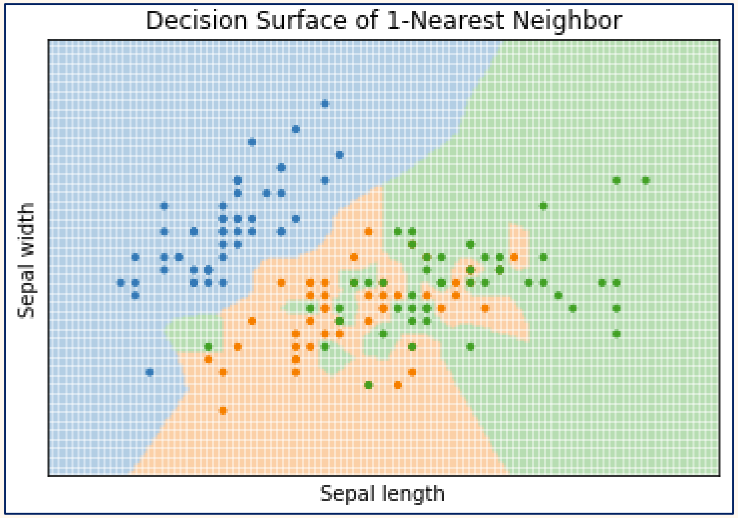
- The following classifiers are introduced
  - k-nearest Neighbor
  - Decision Trees
  - Random Forests
  - Logistic Regression
  - Naive Bayes
  - Support Vector Machines
  - Neural Networks

k-nearest Neighbor

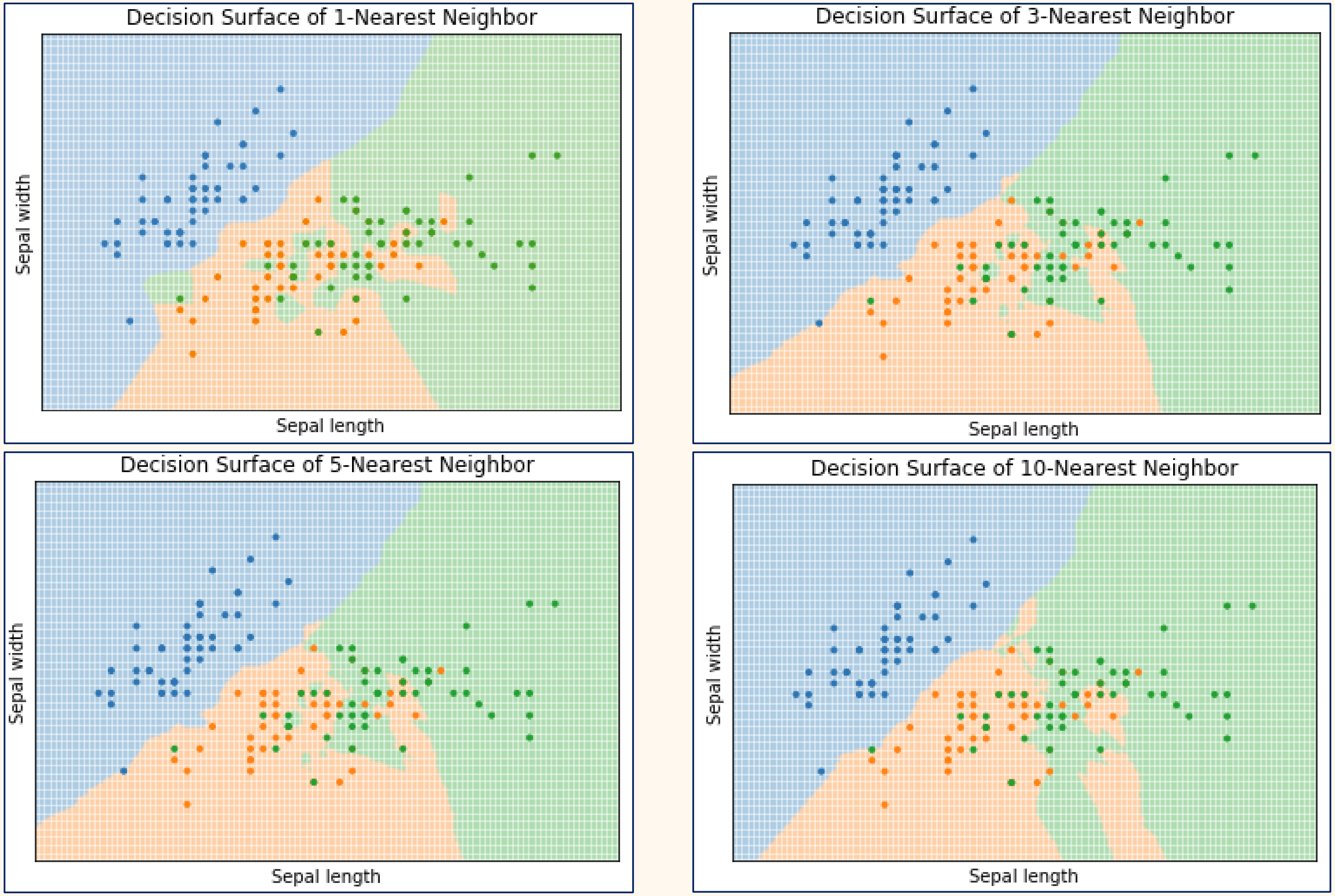
- Basic Idea
  - Instances with similar feature values should have the same class
  - Class can be determined by looking at instances that are similar

→ Assign each instance the mode of its k nearest instances



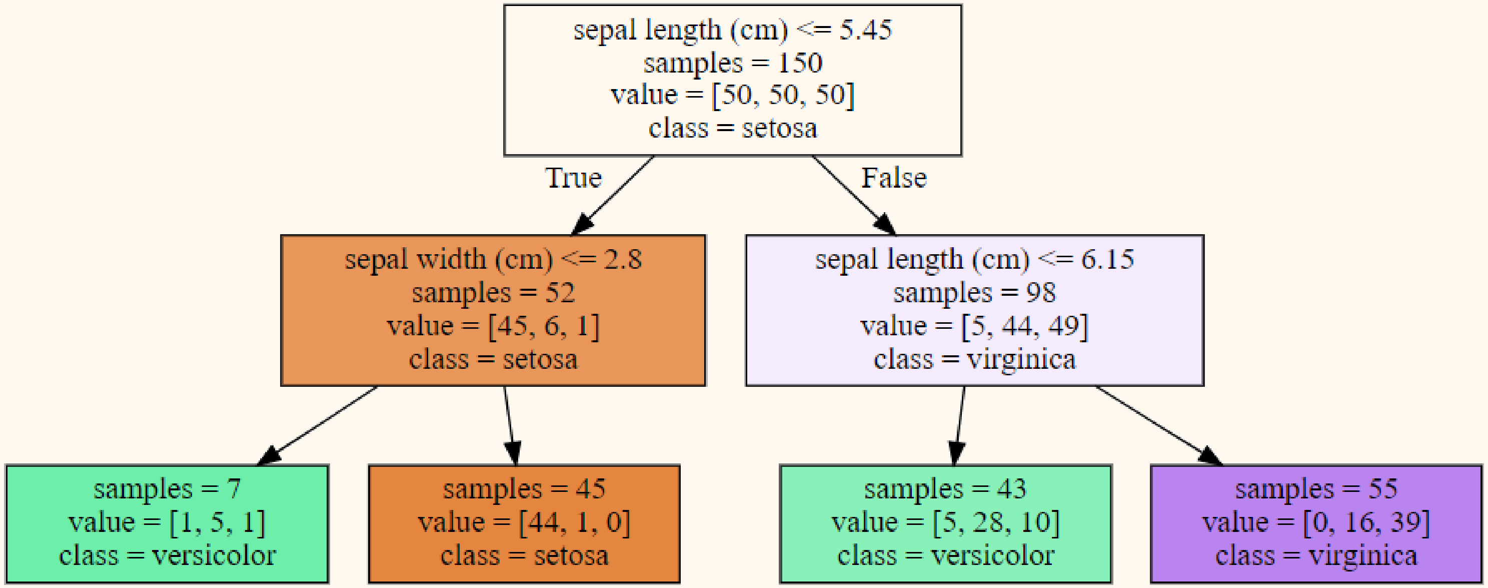


Impact of k


Decision Surface of 1-Nearest Neighbor
Sepal width
Sepal length
Decision Surface of 3-Nearest Neighbor
Sepal width
Sepal length
Decision Surface of 5-Nearest Neighbor
Sepal width
Sepal length
Decision Surface of 10-Nearest Neighbor
Sepal width
Sepal length

Decision Trees

- Basic Idea
  - Make decisions based on logical rules about features
  - Organize rules as a tree



Basic Decision Tree Algorithm

- Recursive algorithm
  - Stop if
    - Data is “pure”, i.e. mostly from class
    - Amount of data is too small, i.e., only few instances in partition
  - Otherwise
    - Determine „most informative feature“ X
    - Partition training data using X
    - Recursively create subtree for each partition
- Details may vary depending on the specific algorithm
  - For example, CART, ID3, C4.5
- General concept always the same

The „Most Informative Feature“

- Information theory based approach

- Entropy of the class label
  - $H(C) = -\sum_{c \in C} p(c) \log p(c)$

Can be used as measure for purity

- Conditional entropy of the class label based on feature X
  - $H(C|X) = -\sum_{x \in X} p(x) \sum_{c \in C} p(c|x) \log p(c|x)$

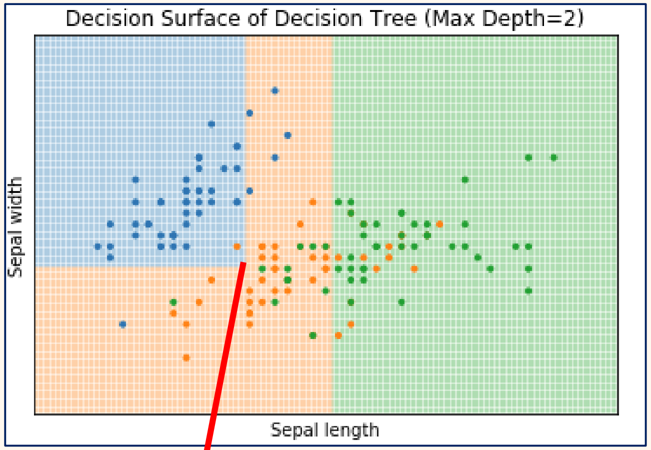
Interpret each dimension as random variable

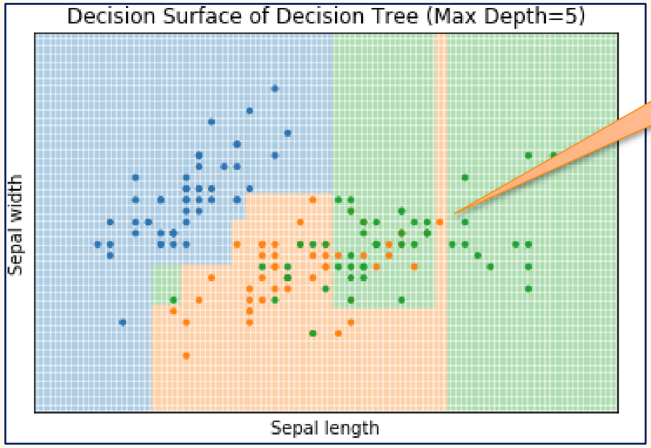
- Mutual Information
  - $I(C, X) = H(C) - H(C|X)$

→ Feature with highest mutual information is most informative

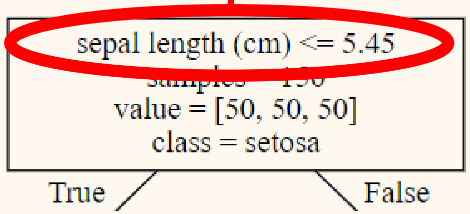
Decision Surface of Decision Trees

- All decisions are axis-aligned





Overfitting

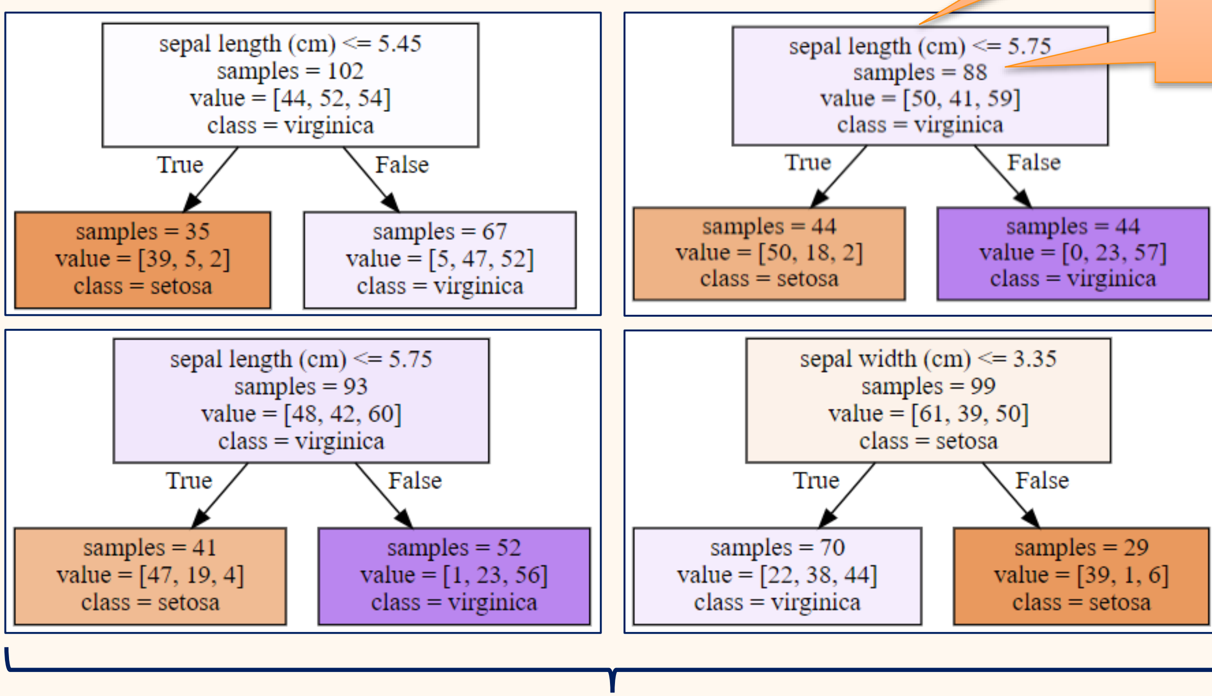


Random Forest

- Basic Idea
  - Ensemble of randomized decision trees

Randomized attributes

Randomized subset



Classification as majority vote of random trees

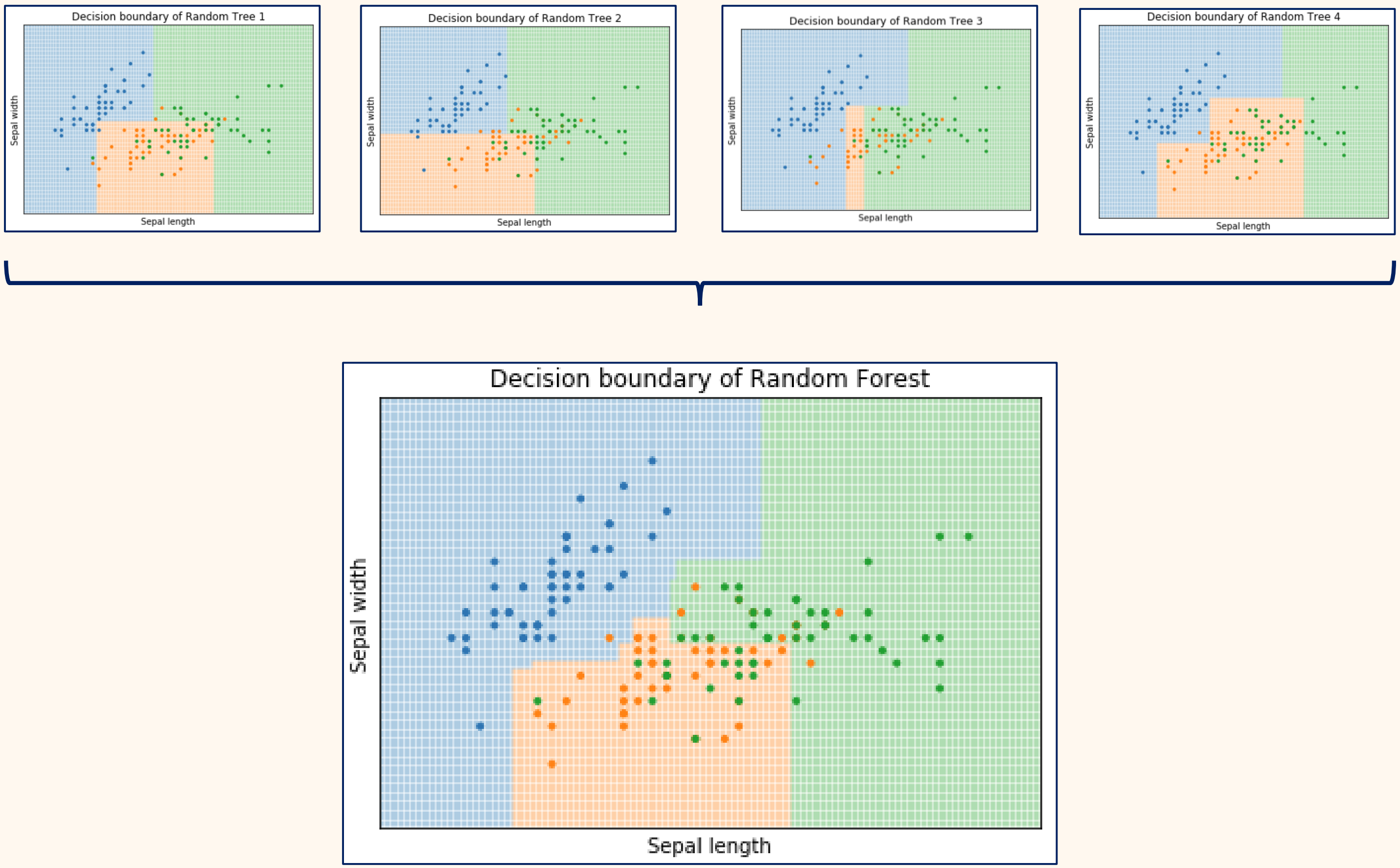
Bagging as Ensemble Learner

- Bagging is short for *bootstrap aggregating*

- Randomly draw subsamples of training data
- Build model for each subsample → ensemble of models
- Voting to create class
  - Can be weighted, e.g., using quality of ensemble models

- Random Forests combine Bagging with
  - Short decision trees, i.e., low depth
  - Allowing only a random subset of features for each decision

Decision Surface of Random Forests



Logistic Regression

- Basic Idea:
  - Regression model of the probability that an object belongs to a class
  - Combines the $logit$ function with *linear regression*

- Linear Regression
  - y as linear combination of $x_1, \ldots, x_n$
  - $y = b_0 + b_1 x_1 + \cdots + b_n x_n$
- The $logit$ function
  - $logit\big(P(y=c)\big) = \ln \frac{P(y=c)}{1-P(y=c)}$
- Logistic Regression
  - $logit\big(P(y=c)\big) = b_0 + b_1 x_1 + \cdots + b_n x_n$

Odds Ratios

- Probabilities vs. Odds
  - Probability: $P(\text{pass\_exam}) = 0.75$
  - Odds of passing the exam: $odds(\text{pass\_exam}) = \frac{0.75}{1-0.75} = 3$
    - The odds if passing the exam is 3 to 1

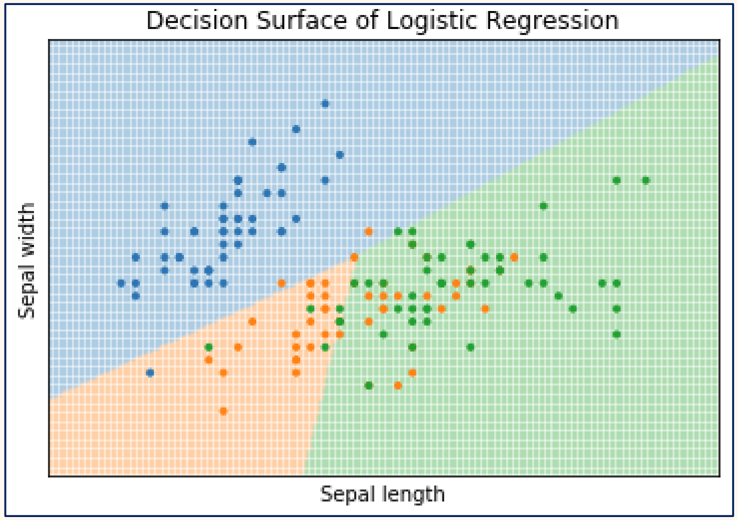
- If we invert the natural logarithm, we get
  - $\frac{P(y=c)}{1-P(y=c)} = \exp(b_0 + b_1 x_1 + \cdots + b_n x_n) = \prod_{j=0}^{n} \exp(b_j x_j)$

Definition of odds

- It follows that $\exp(b_j)$ is the odds ratio of feature j
  - Odds ratio means the change in odds if we increase x_j by one.
  - Odds ratio greater than one means increased odds
  - Odds ratio less than one mean decreased odds

Decision Surface of Logistic Regression

- Decision boundaries are linear



Naive Bayes

- Basic idea:
  - Assume all features as independent
  - Score classes using the conditional probability

- Bayes Law
  - $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$

- Conditional probability of a class:
  - $P(c|x_1, \dots, x_n) = \frac{P(x_1,\dots,x_n|c)P(c)}{P(x_1,\dots,x_n)}$

From Bayes Law to Naive Bayes

- Probability following Bayes law
  - $P(c|x_1, \dots, x_n) = \frac{P(x_1,\dots,x_n|c)P(c)}{P(x_1,\dots,x_n)}$
- „Naive“ assumption: $x_1, \dots, x_n$ conditionally independent given c
  - $P(c|x_1, \dots, x_n) = \frac{P(x_1|c) \dots P(x_n|c)\; P(c)}{P(x_1,\dots,x_n)} = \frac{\prod_{j=1}^{n} P(x_j|c)\; P(c)}{P(x_1,\dots,x_n)}$
- $P(x_1, \dots, x_n)$ is independent of c and always the same
  - $score(c|x_1, \dots, x_n) = \prod_{j=1}^{n} P(x_j|c)\;\; P(c)$
- Assign the class with highest score

Multinomial and Gaussian Naive Bayes

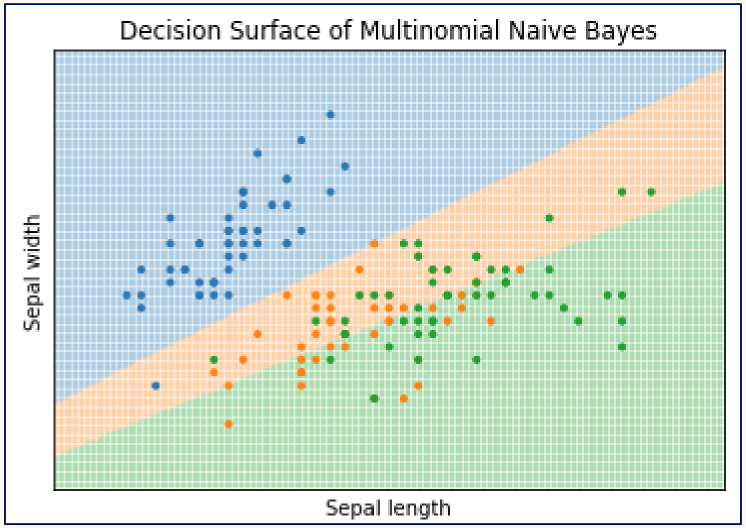
- Different variants on how $P(x_j|c)$ is estimated

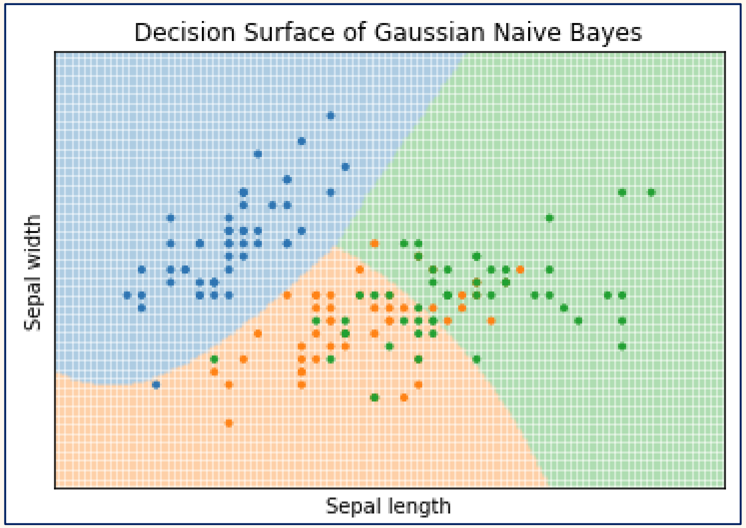
- Multinomial
  - $P(x_j|c)$ is the empirical probability of observing a feature
  - "Counts" observations of x_j in the data

- Gaussian
  - Assumes features follow a gaussian/normal distribution
  - Estimates $P(x_j|c)$ conditional probability using the gaussian density function

Decision Surface of Naive Bayes

- Multinomial has linear decision boundaries
- Gaussian has piecewise quadratic decision boundaries

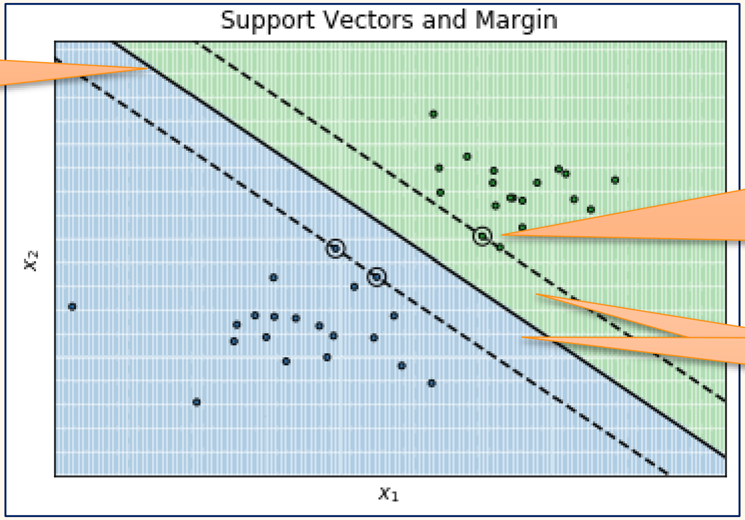




Support Vector Machines (SVM)

- Basic Idea:
  - Calculate decision boundary such that it is “far away” from data

Linear decision boundary



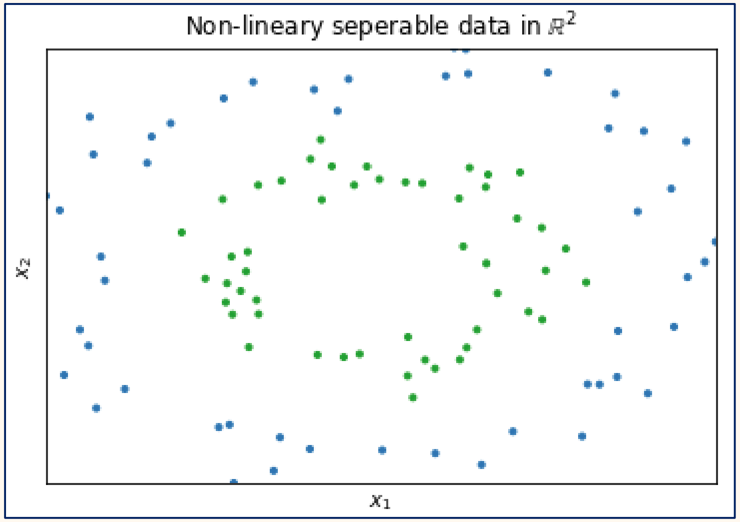
Support vectors = Instances with minimal distance to decision bounday

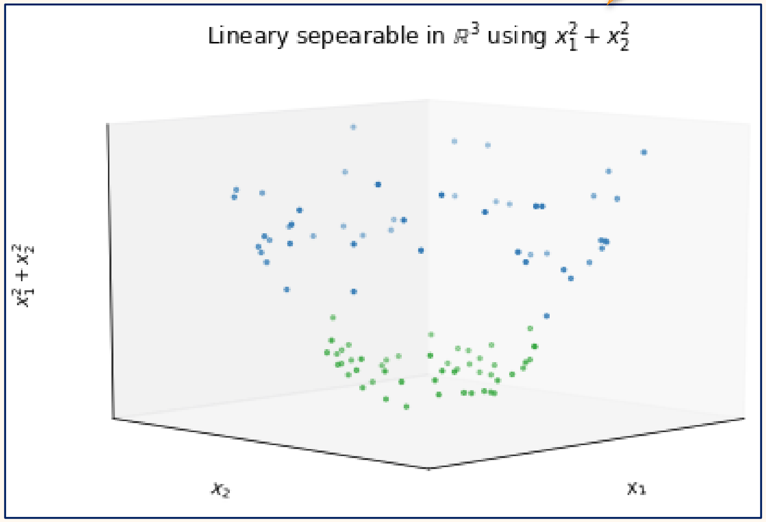
Margin is maximized

Non-linear SVMs through Kernels

- Expand features using *kernels* to separate non-linear data
  - Transformation into high-dimensional *kernel space*
    - Can be infinite (e.g., Gaussian kernel, RBF kernel) !
  - Calculate linear separation in kernel space
  - Use *kernel trick* to avoid actual expansion

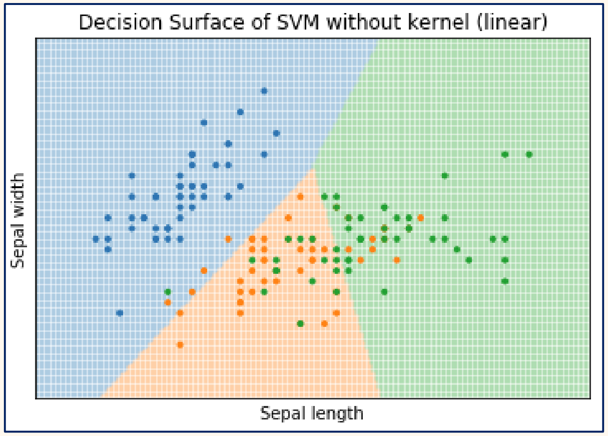
Quadractic kernel

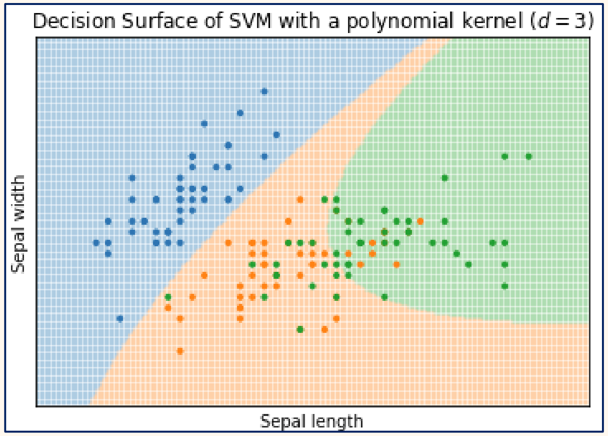


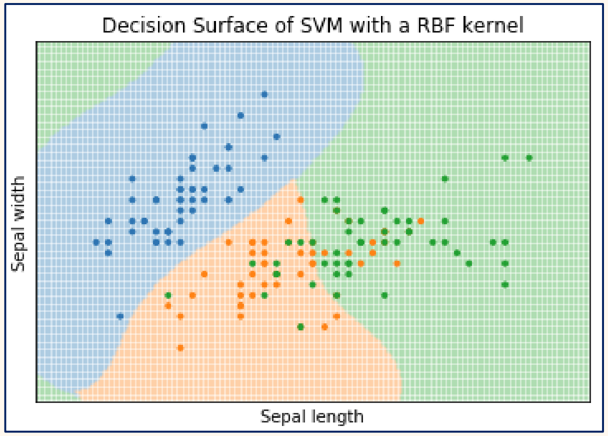


Decision Surface of SVMs

- Shape of decision surface depends on kernel

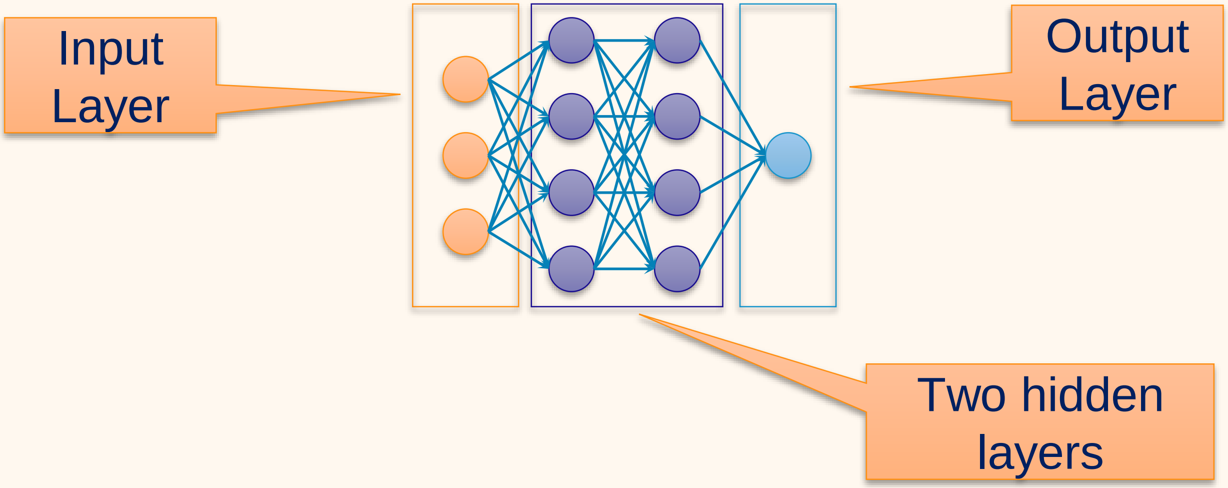






Neural Networks

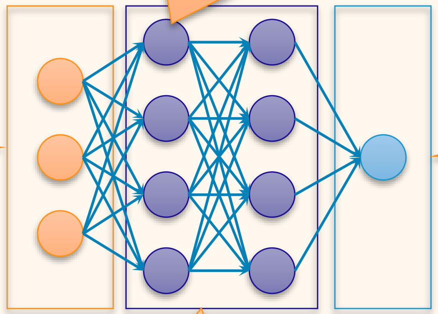
- Basic Idea:
  - Network of neurons with different layers and communication between neurons
  - Input layer feeds data into the network
  - Hidden layers “correlate” data
  - Output layer gives computation results



Multilayer Perceptron (MLP)

- First weighted sum of inputs
- Then *activation function*, e.g, $sigmoid/tanh$

Each feature gets an input neuron

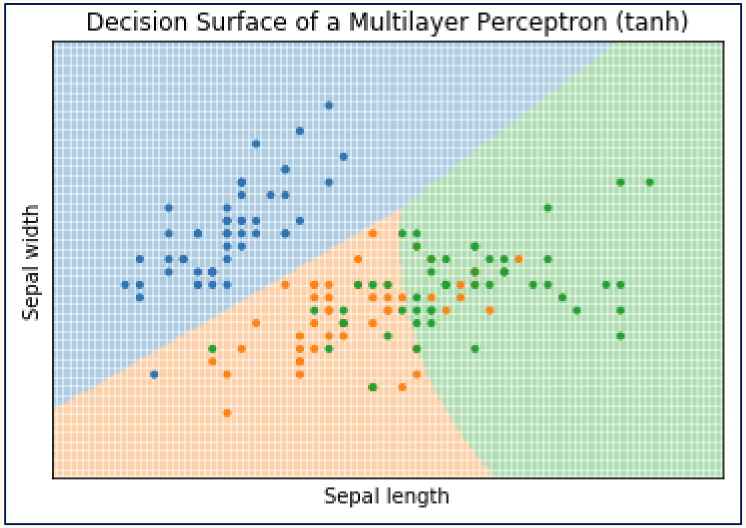


Single output neuron with the classification

Multiple fully connected hidden layers

Decision Surface of MLP

- Shape of decision boundary depends on
  - Activation function
  - Number of hidden layers
  - Number of neurons in the hidden layers



Outline

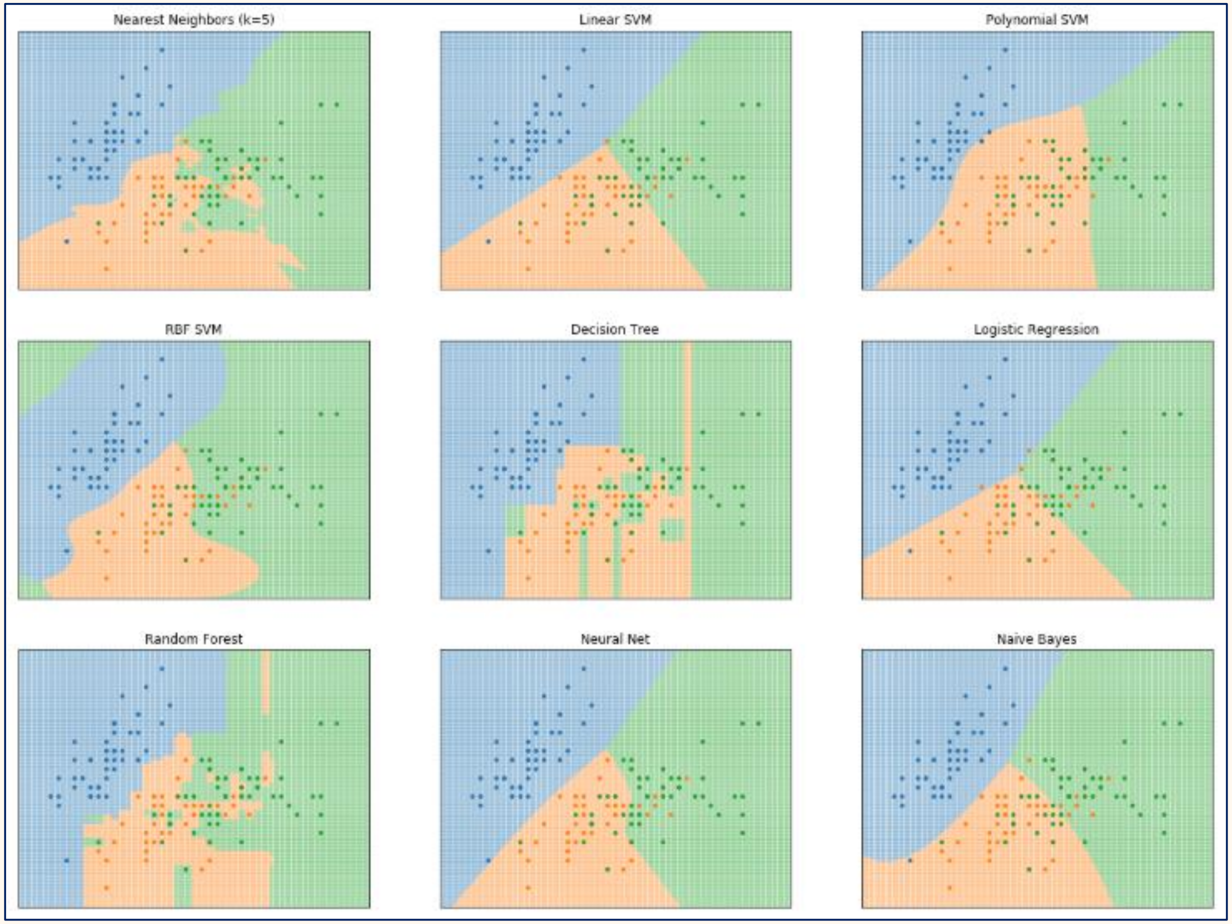
- Overview
- Classification Models
- **Comparison of Classification Models**
- Summary

General Approach

- Different approaches behind all covered classifiers

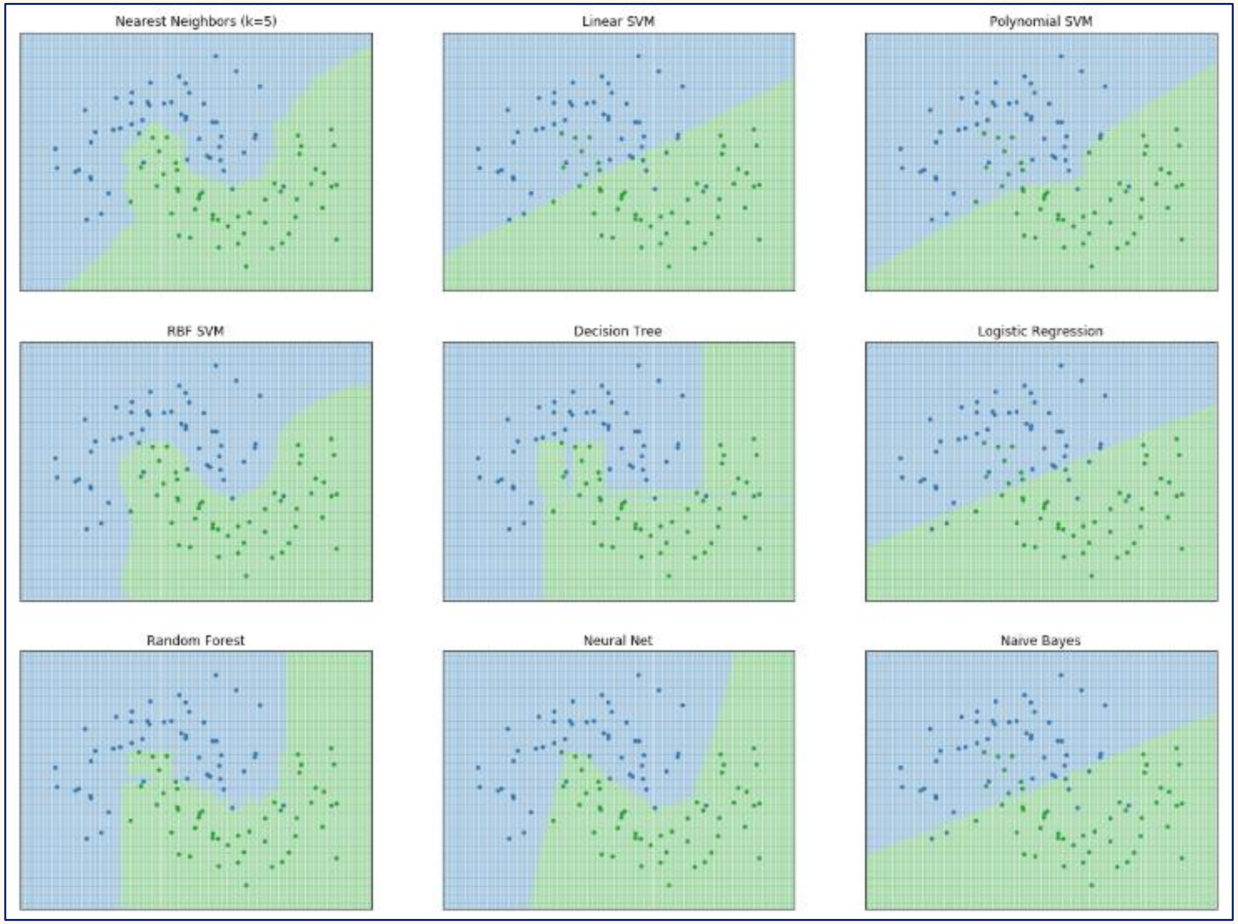
  - k-nearest Neighbor → Instance based
  - Decision Trees → Rule based + information theory
  - Random Forests → Randomized ensemble
  - Logistic Regression → Regression
  - Naive Bayes → Conditional probability
  - Support Vector Machines → Margin maximization + kernels
  - Neural Networks → (Very complex) Regression

Comparison of Decision Surfaces
IRIS Data



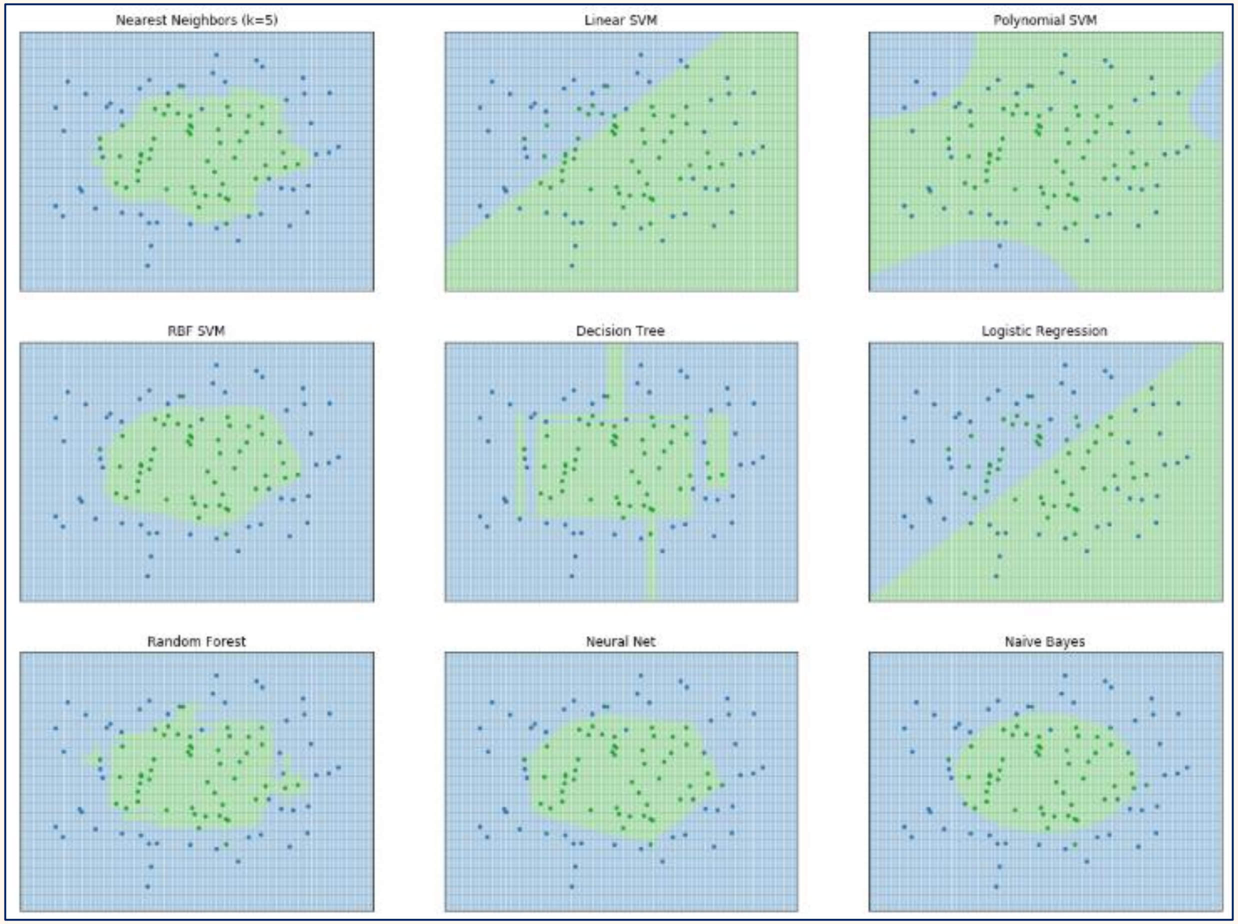
Results may vary with hyper parameter tuning

Comparison of Decision Surfaces
Non-linear separable



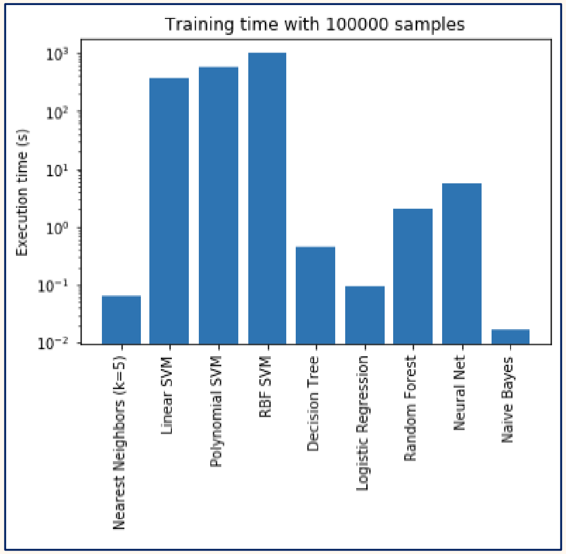
Results may vary with hyper parameter tuning

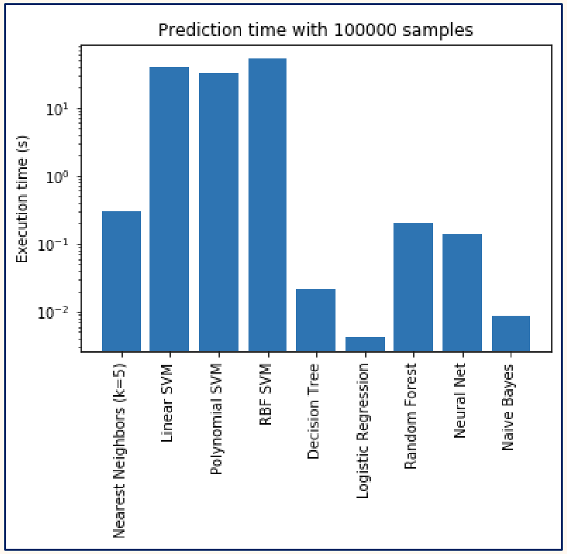
Comparison of Decision Surfaces
Circles within circles



Results may vary with hyper parameter tuning

Comparison of Execution Times





Times taken using GWDG Jupyter Hub and scikit-learn implementations of the algorithms.
Data randomly generated with using scikit-learn.datasets.make_moons (July 2018)

Strengths and Weaknesses

	Explanatory value	Consise representation	Scoring	Categorical features	Missing features	Correlated features
k-nearest Neighbor	o	-	-	-	+	-
Decision Tree	+	+	+	+	o	+
Random Forest	-	o	+	+	o	+
Logistic Regression	+	+	+	o	-	o
Naive Bayes	o	o	+	+	-	-
SVM	-	o	-	o	-	-
Neural Network	-	o	+	o	-	+

Outline

- Overview
- Classification Models
- Comparison of Classification Models
- **Summary**

Summary

- Classification is the task of assigning labels to objects
- Many evaluation criteria
  - Confusion matrix commonly used
- Lots of classification algorithms
  - Rule based, instance based, ensembles, regressions, …
- Different algorithms may be best in different situations