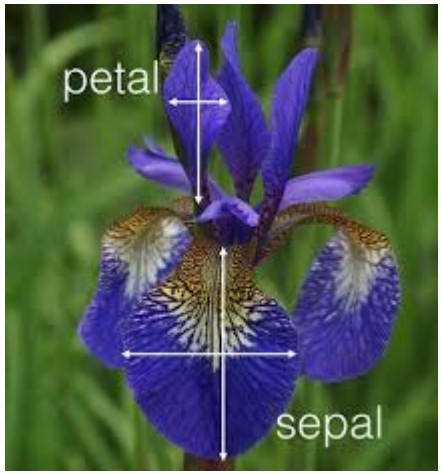
Alexander Hofmann

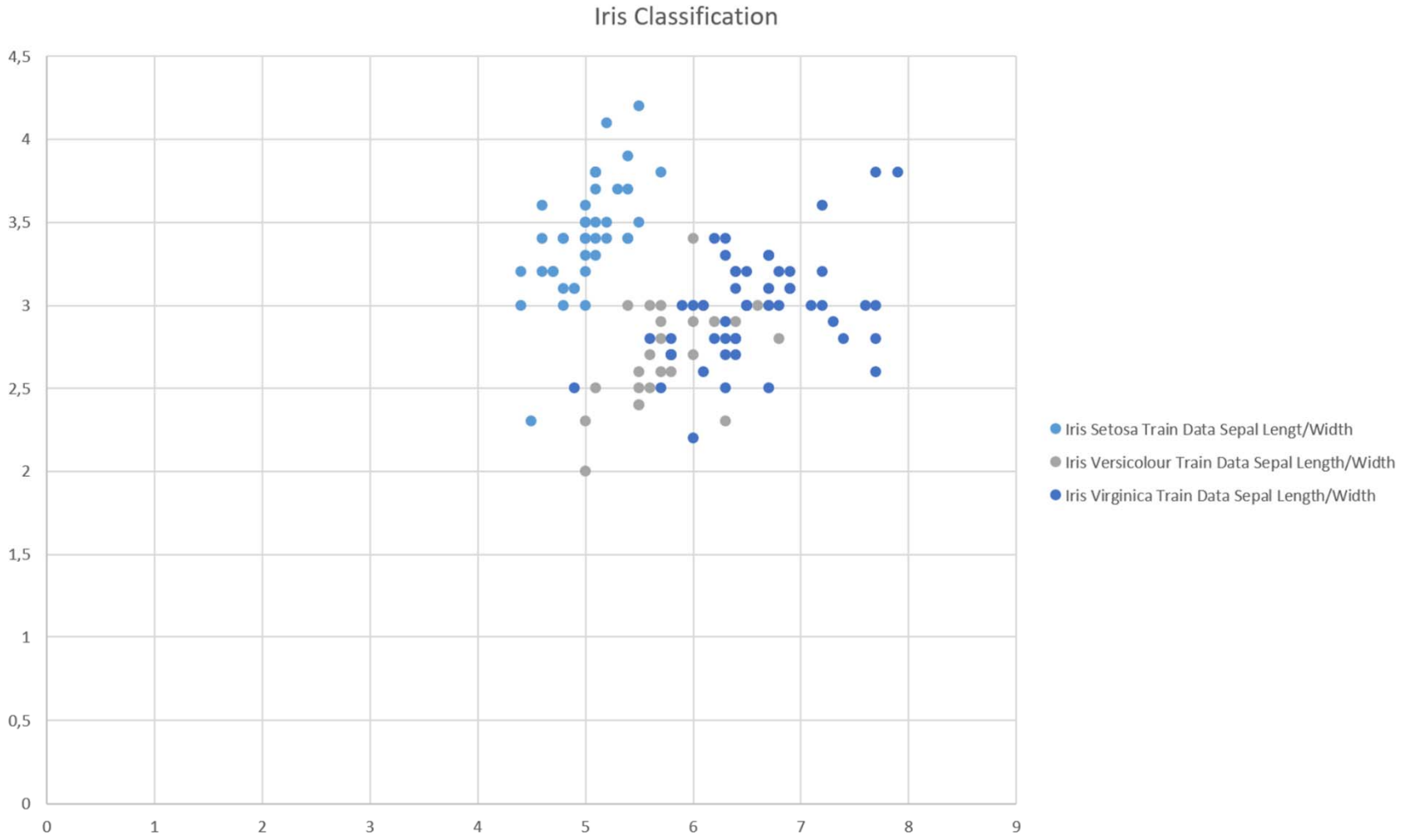
Machine Learning

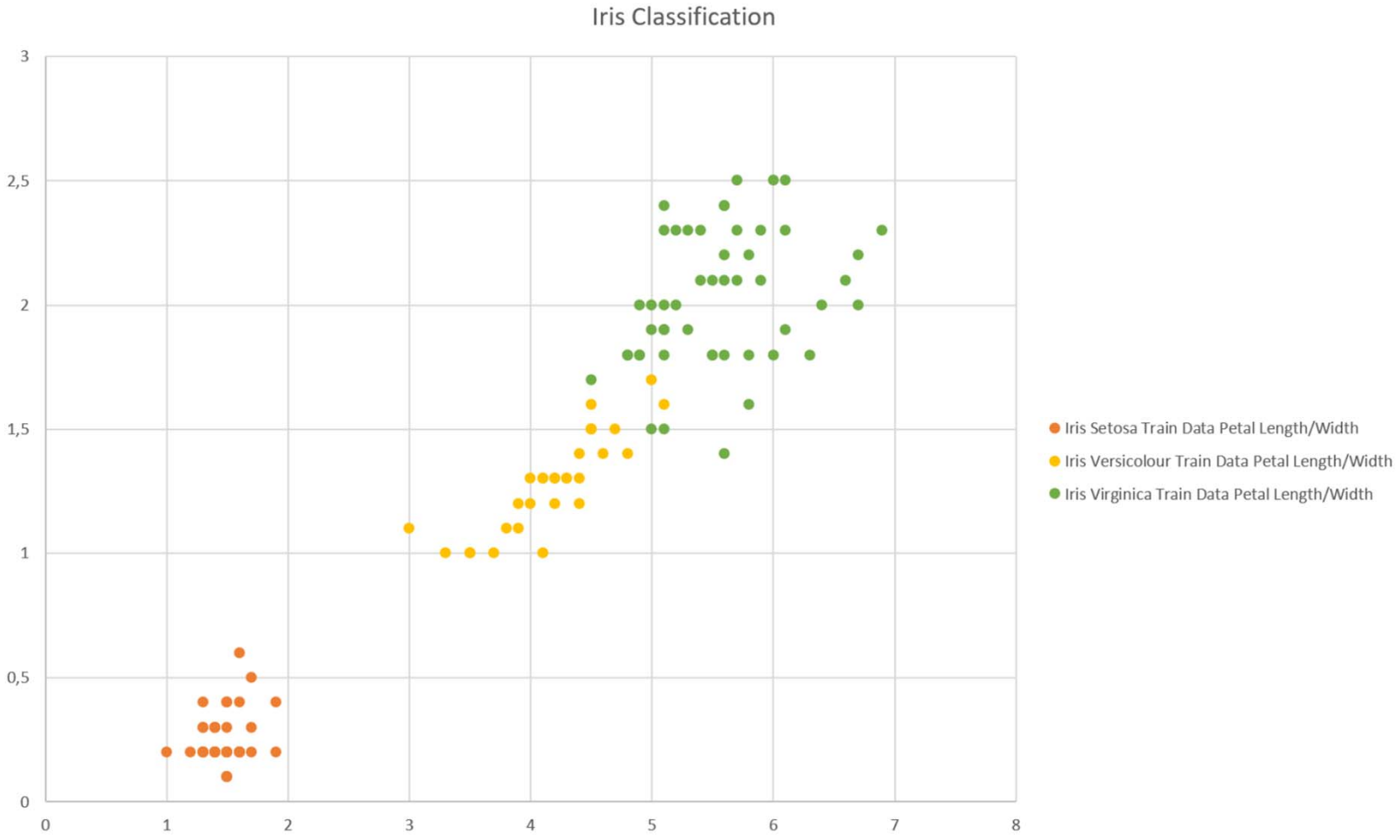
Algorithmen

- Nearest Neighbor oder K-Nearest Neighbor
- Beispiel Iris Datenbank
  - Arten von Iris:
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica



Classifier





Daten einlesen – File einlesen

```
String content = null;
try {
    content = new String(Files.readAllBytes(Paths.get("data/iris.txt")));
} catch (IOException e) {
    e.printStackTrace();
}
String[] lines = content.split("\r\n");
```

Objekte erzeugen

```
ArrayList<Iris> irises = new ArrayList<Iris>();
ArrayList<IrisSetosa> iSetosa = new ArrayList<IrisSetosa>();
ArrayList<IrisVersicolour> iVersicolour = new ArrayList<IrisVersicolour>();
ArrayList<IrisVirginica> iVirginica = new ArrayList<IrisVirginica>();

for (int i = 0; i < lines.length; i++) {
    String[] data = lines[i].split(",");
    float sepallength = Float.parseFloat(data[0]);
    float sepalwidth = Float.parseFloat(data[1]);
    float petallength = Float.parseFloat(data[2]);
    float petalwidth = Float.parseFloat(data[3]);
    Iris iris = null;
    if (data[4].equalsIgnoreCase("iris-setosa")) {
        iris = new IrisSetosa(sepallength, sepalwidth, petallength, petalwidth);
        iSetosa.add((IrisSetosa) iris);
    } else {
        if (data[4].equalsIgnoreCase("iris-versicolor")) {
            iris = new IrisVersicolour(sepallength, sepalwidth, petallength, petalwidth);
            iVersicolour.add((IrisVersicolour) iris);
        } else {
            if (data[4].equalsIgnoreCase("iris-virginica")) {
                iris = new IrisVirginica(sepallength, sepalwidth, petallength, petalwidth);
                iVirginica.add((IrisVirginica) iris);
            }
        }
    }
    if (iris == null) {
        System.out.println("wrong data line[" + i + "]:\n" + lines[i]);
    }
    irises.add(iris);
}
```

Daten aufteilen

- Daten werden in Lern- und Testdaten aufgeteilt
  - Marsland: Training, Testing and Validation Data
- Verschiedene Methoden
  - k-fold cross validation (gängiste: 10-fold cross validation)
  - Abhängig von der Anzahl der verfügbaren, vorklassifizierten Daten (Supervised learning!)
- Ziel: Overfitting vermeiden
- Algorithmen müssen auf ihre Fehlerfreiheit (Accuracy) und Genauigkeit überprüft werden
- Ergebnis ist eine „Accuracy Matrix“ oder auch „Confusion Matrix“

k-fold cross Validation

- Kreuzvalidierung
  - N Instanzen werden in $k \leq N$ gleich große Teilmengen $T_1, \ldots, T_k$ aufgeteilt
  - k Testdurchläufe
  - Bei jedem Durchlauf wird T_i für Tests nach der „Lernphase“ zurückgestellt
  - Unbedingt darauf achten, dass die Classifier gleich verteilt sind!
  - Bei jedem Durchlauf werden alle Werte der Confusion Matrix mitgezählt
  - Die endgültige Confusion Matrix für den Algorithmus enthält die Durchschnittswerte der Einzelauswertungen

Daten aufteilen

```
//Split Data into train and test data
ArrayList<Iris> test = new ArrayList<Iris>();
int r;
for (int i = 0; i < 5; i++) {
    r = (int) (Math.random() * (50-i));
    test.add(iSetosa.get(r));
    iSetosa.remove(r);
}
for (int i = 0; i < 5; i++) {
    r = (int) (Math.random() * (50-i));
    test.add(iVersicolour.get(r));
    iVersicolour.remove(r);
}
for (int i = 0; i < 5; i++) {
    r = (int) (Math.random() * (50-i));
    test.add(iVirginica.get(r));
    iVirginica.remove(r);
}
ArrayList<Iris> train = new ArrayList<Iris>();
train.addAll(iSetosa);
train.addAll(iVersicolour);
train.addAll(iVirginica);
```

Lern- oder Trainingsphase

- Bei kNN kann ein KD-Tree aufgespannt werden sofern wenige Attribute zum Einsatz kommen
  - Aufbau: $\sim O(N \log N)$
  - Nearest neighbor search: $\sim O(\log N)$
  - k nearest neigbor search: $\sim O(\mathrm{k} \log N)$
- Einfache Liste
  - Aufbau: $\sim O(N)$
  - k nearest neigbor search: $\sim O(N)$
  - Hinzu kommt der Aufwand zu Berechnung der Distanz

Distanzen zwischen 2 Instanzen

- Größen
  - n .. Anzahl der Attribute
  - a .. i. Attribut Instanz 1
  - b .. i. Attribut Instanz 2
- Euklidische: $d_E = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$
- Manhattan (City block): $d_C = \sum_{i=1}^{n}|a_i - b_i|$
- Minowski: $L_k(x, y) = \left(\sum_{i=1}^{d}|x_i - y_i|^k\right)^{\frac{1}{k}}$
  - Allgemeine Metrik, k = 1 .. Manhattan, k = 2 Euklidische
- Tangent distance
  - Vergleich von Bildern (z.B. Handschrifterkennung)

Euklidische Distanz

```
public class IrisEuklidianDistanceComparator implements Comparator<Iris> {
    private Iris candidate;

    public IrisEuklidianDistanceComparator(Iris candiate) {
        this.candidate = candiate;
    }
    public int compare(Iris test1, Iris test2) {
        double dist1 = 0.0d;
        double dist2 = 0.0d;
        dist1 = Math.sqrt(  Math.pow(test1.getSepalLength() - candidate.getSepalLength(), 2) +
                            Math.pow(test1.getSepalWidth() - candidate.getSepalWidth(), 2) +
                            Math.pow(test1.getPetalLength() - candidate.getPetalLength(), 2) +
                            Math.pow(test1.getPetalWidth() - candidate.getPetalWidth(), 2));
        dist2 = Math.sqrt(  Math.pow(test2.getSepalLength() - candidate.getSepalLength(), 2) +
                            Math.pow(test2.getSepalWidth() - candidate.getSepalWidth(), 2) +
                            Math.pow(test2.getPetalLength() - candidate.getPetalLength(), 2) +
                            Math.pow(test2.getPetalWidth() - candidate.getPetalWidth(), 2));
        if (dist1 > dist2) {
            return 1;
        }
        if (dist1 < dist2) {
            return -1;
        }
        return 0;
    }
}
```

Anwendung

```
for (Iris irisTest : test) {
    IrisEuklidianDistanceComparator iedc =
            new IrisEuklidianDistanceComparator(irisTest);
    Collections.sort(train, iedc);
    // the first k elements in train are the
    // k nearest neighbors!
}
```

Confusion Matrix

- Multiclass Classification Confusion Matrix

		Predicitons		
		Setosa	Versicolour	Virginica
Actual/Real	Setosa	35		
	Versicolour		29	6
	Virginica			35

- TP, true positive
- $Accuracy = \frac{\#TP + \#TN}{\#TP + \#FP + \#TN + \#FN}$

Evaluation

- Kreuzvalidierung
  - Mit jedem Testset einmal testen – gegen den Rest der Instanzen
  - Durchschnitt der Werte der Confusion Matrix bilden