Fakultät Wirtschaftsinformatik und Angewandte Informatik

Otto-Friedrich-Universität Bamberg

# Projekt „Das Schmerzgesicht“

Mark Gromowski (Matr. Nr. 1576483)
Benjamin Heibel (Matr. Nr. 1688930)

Projektbericht

WS 2015/16

31. März 2016

Betreuer: Michael Siebers

**Zusammenfassung**

Die vorliegende Projektarbeit entstand im Rahmen eines Projektes mit dem Titel „Das Schmerzgesicht“ des Lehrstuhls für Kognitive Systeme der Otto-Friedrich-Universität Bamberg und beschäftigt sich mit dem Thema der Schmerzerkennung in der Mimik des menschlichen Gesichts durch Maschinen. Als Basis der Untersuchung dienten Daten, die mithilfe des *Facial Action Coding Systems* aus Videosequenzen extrahiert wurden, in denen Probanden verschiedene Emotionen induziert wurden. Diese Daten wurden durch die Projektgruppe in eine geeignete Repräsentation verarbeitet und anschließend als Trainingsmenge für die Lernverfahren *Decision Tree Learning* und *KNN* aus dem Bereich des *Machine Learning* verwendet. Der Grund für die mittelmäßige *performance* der Schmerzerkennung könnte in ungeeigneten Lernverfahren oder aber einer ungeeigneten Attributauswahl für die Repräsentation der Daten liegen - für letztere wurden z. B. zeitliche Informationen wie die Dauer einer *action unit* oder die Reihenfolge der *action units* bewusst weggelassen. Zu einer umfassenderen Analyse sind die Projektarbeiten der weiteren Projektgruppen heranzuziehen, welche die zeitlichen Informationen zu den *action units* mit einbeziehen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Kapitel 1

# Einleitung

Im Kontext der fortschreitenden Digitalisierung der Gesellschaft ergeben sich fortlaufend neue Möglichkeiten der Kommunikation zwischen Menschen und Maschinen. Sogar die Erkennung menschlicher Emotionen durch Maschinen nimmt dabei immer deutlichere Züge an und eröffnet damit neue Möglichkeiten auf den verschiedensten Anwendungsfeldern.

Das Projekt des Lehrstuhls für Kognitive Systeme an der Otto-Friedrich-Universität Bamberg, in dessen Rahmen diese Projektarbeit entstanden ist, beschäftigte sich mit der Erkennung von menschlicher Schmerzempfindung als Kompetenz künstlicher Intelligenz. Ausgehend von Videosequenzen, welche die Mimik einer Reihe von Versuchsteilnehmern während der Induktion verschiedener Emotionen erfassten, und der anschließenden Codierung dieser Mimik in beobachtete *action units* des *Facial Action Coding Systems*, wurden verschiedene Techniken des *Machine Learning* angewandt, um eine möglichst korrekte automatische Klassifizierung einzelner Sequenzen in die verschiedenen Emotionen, insbesondere Schmerz, zu erarbeiten.

Die Arbeit dieser Gruppe untersuchte die Klassifizierung der vorliegenden Sequenzen anhand der beiden Lernverfahren *Decision Tree Learning* und *K nearest Neighbor (KNN)*. Zunächst musste eine geeignete Repräsentation der vorhandenen Daten als *Trainingsmenge* erarbeitet werden; anschließend bestand die Aufgabe darin, durch einen geeigneten Aufbau der Lernalgorithmen in *RapidMiner*, einer Software für maschinelles Lernen und *Data Mining*, eine möglichst hohe *performance* in der Klassifizierung verschiedener Emotionen, insbesondere Schmerz, zu erzielen. Der Anspruch an die *performance* des Lernverfahrens sollte dabei hoch gewählt werden, da es sich gerade bei Schmerz um ein sensibles Thema handelt, bei dem eine hinreichende Genauigkeit in der Erkennung hohe Priorität hat.

Der Arbeit liegt folgender Aufbau zugrunde: im zweiten Kapitel werden zunächst die grundlegenden Konzepte des *Machine Learning* und des *Facial Action Coding Systems* sowie die damit verbundenen zentralen Begriffe kurz skizziert. Das dritte Kapitel geht im Anschluss auf die verfügbaren Daten und die durch die Projektgruppe gewählte Repräsentation der Daten im Rahmen der Attributauswahl ein. Die hieraus resultierende Repräsentation bildet die

Grundlage für die folgenden Kapitel: im vierten Kapitel wird zunächst das Lernverfahren des *Decision Tree Learning* vorgestellt und der Aufbau und die Durchführung des Lernverfahrens mit der zuvor erstellten *Trainingsmenge* beschrieben; anschließend werden die Ergebnisse interpretiert; im fünften Kapitel werden die entsprechenden Schritte für das *KNN*-Verfahren durchgeführt, bevor ein abschließendes Fazit die Arbeit beendet.

# Kapitel 2

# Grundlegende Konzepte, Verfahren und Begriffe

## 2.1 Machine Learning

Maschinelles Lernen ist ein Oberbegriff für die Generierung von Wissen aus Erfahrung. Ein System lernt aus Beispielen und kann diese nach Beendigung der Lernphase verallgemeinern. Das heißt es erkennt Muster und Gesetzmäßigkeiten in den Lerndaten. So kann das System auch unbekannte Daten beurteilen.

## 2.2 Data Mining

Beim *Data Mining* geht es um die „Suche nach Mustern und Erkenntnissen in den Daten“. Dabei lassen sich zwei Ziele unterscheiden: die Gewinnung von durch Menschen interpretierbaren Erkenntnissen und das Schätzen unbekannter Werte (Zielvariablen) aus bekannten Werten (Eingangsvariablen). Ausgangspunkt sind in jedem Fall strukturierte Daten in Form einer Tabelle (liegen mehrere verknüpfte Tabellen oder unstrukturierte Daten vor, müssen diese zuvor mithilfe einer *Feature Extraction* in geeigneter Form transformiert werden. Typische Aufgabenstellungen des *Data Mining* sind *Regression*, *Klassifikation*, *Automatische Segmentierung* und *Assoziationsanalyse*.[4]

## 2.3 RapidMiner

*RapidMiner* ist ein Open-Source-Data-Mining-Werkzeug, welches aus dem an der Universität Dortmund entwickelten Data-Mining-Werkzeug *YALE* hervorgegangen ist. Die grafische Oberfläche ermöglicht dem Nutzer die Entwicklung von Lösungen für Data-Mining-Aufgaben, ohne Programmierkenntnisse vorauszusetzen. Dabei stehen dem Anwender sowohl Verfahren zum Einlesen und Transformieren der Eingangsdaten als auch alle gängigen Data-Mining-Algorithmen in verschiedenen Variationen zur Verfügung.[4]

## 2.4 Facial Action Coding System

Das *Facial Action Coding System* (kurz *FACS*) ist ein unter Psychologen weltweit verbreitetes Kodierungsverfahren zur Beschreibung von Gesichtsausdrücken. Ausgehend von der Annahme, dass jede Bewegung des Gesichts durch Muskelanspannung oder -entspannung hervorgerufen wird, werden sämtliche sichtbaren Veränderungen der Gesichtsmuskeln erfasst und damit „jede Gesichtsbewegung in anatomisch abgeleitete minimale Bewegungseinheiten (*action units*)“[2] zerlegt. Mit dieser Klassifikation ist es möglich, Gesichtsausdrücke schriftlich zu notieren – vergleichbar der Notation von verbalen Ausdrücken mit geschriebener Sprache. Im FACS gibt es insgesamt 44 solcher Einheiten, das Obergesicht umfasst 12, das Untergesicht 32 *action units*. Die *action units* im Untergesicht werden hinsichtlich der Richtung der Bewegungen unterteilt: so lassen sich horizontale, vertikale, schräge, kreisförmige und gemischte Aktionen unterscheiden.

## 2.5 Begriffe

### 2.5.1 Precision

Genauigkeit (engl. precision) ist die Wahrscheinlichkeit, mit der ein gefundenes Dokument relevant ist (Positiver Vorhersagewert).

### 2.5.2 Accuracy

Accuracy beschreibt die Wahrscheinlichkeit korrekter Vorhersagen.

### 2.5.3 Recall

Trefferquote (engl. recall) ist die Wahrscheinlichkeit, mit der ein relevantes Dokument gefunden wird.

### 2.5.4 Hauptkomponentenanalyse

Die Hauptkomponentenanalyse (engl. principal component analysis) ist eine mathematische Prozedur welche durch orthogonale Transformation ursprünglicher (möglicherweise korrelierter) Variablen eine neue (meist kleinere) Menge unkorrelierter Variablen erzeugt. Eine beispielhafte Transformation einer Punktmenge ist in 5.2 auf Seite 23 veranschaulicht.

### 2.5.5 Euclidean Distance

Der Euklidische Abstand (engl. Euclidean Distance) ist die einfache, direkte Strecke zwischen zwei Punkten in einem euklidischen Raum. Die Formel für den euklidischen Abstand zwischen zwei Punkten in einem n-dimensionalen Raum ist in 2.1 auf Seite 5 zu sehen.

$$\begin{aligned} \mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned}$$

Abbildung 2.1: Formel der euklidischen Distanz zwischen Punkten p und q

### 2.5.6 Voronoi Diagramm

Gegeben sind eine Anzahl an Punkten; das Voronoi Diagramm beschreibt die Bereiche, die sich am nächsten zu jedem gegebenen Punkt befinden. Diese Bereiche können als Kontrollzonen betrachtet werden.

### 2.5.7 Information Gain

Information Gain beschreibt die Wichtigkeit eines Attributes mithilfe von Entropie Berechnungen.

# Kapitel 3

# Daten und Attributauswahl

## 3.1 Ursprüngliche Daten

Die Grundlage der Untersuchung bilden Videosequenzen, bei denen Probanden wahlweise Schmerz oder Ekel oder aber gar kein Impuls (neutrale Klasse) induziert und die daraufhin gezeigte Mimik der Teilnehmer aufgezeichnet wurde. Diese Videosequenzen wurden im Anschluss ausgewertet und die extrahierten Informationen in vier Tabellen erfasst:

Die Tabelle *raw_sequences* enthält die *sequence_id* jeder einzelnen aufgenommenen Sequenz, die *subject_id* des an der Sequenz beteiligten Probanden und die *induction_id* der induzierten Emotion. Hinzu kommt die für diese Untersuchung nicht relevante Information *fold*.

Die Tabelle *raw_events* enthält eine *event_id* für jedes Event sowie die *sequence_id* der zugehörigen Sequenz - dabei können mehrere Events in einer Sequenz vorkommen. Jedes Event entspricht dem Auftreten einer *action unit*: als Informationen werden hierzu die Nummer der *au* selbst, die zeitlichen Informationen *start_time* sowie *duration*, und die gemessene *intensity*, mit der die *action unit* auftritt, erfasst.

Die Tabelle *raw_subjects* enthält für jeden an der Datenerfassung beteiligten Teilnehmer (ausgewiesen durch seine *subject_id*) die personenbezogenen Daten *name* (für diese Untersuchung irrelevant), *gender*, *age* und *height*.

Die Tabelle *raw_inductions* enthält für jede *induction_id* den zugehörigen *induction_name* der entsprechenden induzierten Emotion.

Auf der Grundlage dieser Tabellen sollte durch die Projektteilnehmer eine geeignete Repräsentation der vorliegenden Informationen in Form einer Tabelle von Trainingsbeispielen erarbeitet werden, auf welche ausgewählte Lernverfahren aus dem Bereich des *Machine Learning* sinnvoll angewendet werden können.

## 3.2 Attributauswahl

Für die Anwendung der Lernverfahren *Decision Tree Learning* und *KNN* auf die vorliegenden Daten wurde eine gemeinsame Auswahl von Attributen aus den vier Ursprungstabellen vorgenommen. Als Trainingsbeispiele sollen die einzelnen beobachteten Sequenzen verwendet werden. Dementsprechend ist die *ID* eines jeden Trainingsbeispiels die *sequence_id* der jeweiligen Sequenz, entnommen aus der Tabelle *raw_sequences*. Die Zielklasse, welche gelernt werden soll, also das in den Trainingsbeispielen ausgewiesene *label*, ist die induzierte Emotion, repräsentiert als *induction_name* aus der Tabelle *raw_inductions*: mögliche Zielklassen sind hier *pain* (Schmerz), *disgust* (Ekel) und *neutral* (keine induzierte Emotion).

Wichtigstes Element zur Schmerzerkennung sind die in den Videosequenzen beobachteten *action units* ($AU$'s): daher werden für jede in den vorliegenden Daten vorkommende $AU$ Die folgenden Informationen ausgewählt:

Die *frequency* einer $AU$ in einer Sequenz beschreibt die Summe aller Vorkommen dieser $AU$ in dieser Sequenz. Dabei wird untersucht, ob das Vorkommen einer $AU$ überhaupt ($frequency > 0$) sowie ein unterschiedlich häufiges Vorkommen der $AU$ entscheidend für die Klassifizierung der Sequenz sind.

Die *max_intensity* einer $AU$ in einer Sequenz beschreibt die höchste beobachtete Intensität dieser $AU$ in dieser Sequenz. Weitere Vorkommen derselben $AU$ mit geringeren Intensitäten werden dabei nicht in die Repräsentation mit aufgenommen: dieser Entscheidung liegt die Annahme zugrunde, dass das Vorkommen einer $AU$ mit der höchsten Intensität in seiner Aussagekraft die Vorkommen mit geringeren Intensitäten bereits einschließt und in diesen damit keine zusätzlichen Informationen enthalten sind. In Bezug auf die *max_intensity* wird untersucht, ob die Intensität einer $AU$ über das Vorkommen der $AU$ hinaus eine Rolle für die Klassifizierung der entsprechenden Sequenz spielt.

Über die auf diese Weise erfassten Informationen zu den *action units* hinaus existieren in den vorliegenden Daten weitere eventuell für die Klassifizierung relevante Informationen. Hierzu gehören insbesondere die *duration* (Dauer) der einzelnen Vorkommen der *action units* sowie die Kombinationen einzelner *action units* in Form von Aufeinanderfolgen (unter der Annahme, dass die Kombination und Reihenfolge mehrerer *action units* für die Klassifizierung relevant ist). Diese Informationen werden in der vorliegenden Untersuchung bewusst weggelassen, um ihre Relevanz im Vergleich mit anderen Projektgruppen, die diese Informationen in ihrer Repräsentation verwenden, genauer zu untersuchen.

Für jede der 31 in den Trainingsdaten vorkommenden *action units* zwischen *au1* und *au43* werden somit die Werte *frequency* und *max_intensity* als Attribute aufgenommen - eine Ausnahme stellt die *action unit au43* dar: da es für diese nicht sinnvoll erscheint, verschiedene Intensitäten zu messen, wird hier nur die *frequency* repräsentiert. Insgesamt ergeben sich so $30 \cdot 2 + 1 \cdot 1 = 61$ Attribute in Bezug auf die *action units*.

Zusätzlich werden für die an der Aufnahme der Sequenzen beteiligten Probanden die personenbezogenen Daten *subject_gender* (Geschlecht), *subject_age* (Alter) und *subject_height* (Größe) als Attribute in die Repräsentation der Trainingsdaten aufgenommen (die Werte werden den Attributen *gender*, *age* und *height* aus der Tabelle *raw_subjects* entnommen). Es erscheint zwar intuitiv unwahrscheinlich, dass solche personenbezogenen Informationen einen Einfluss auf die Schmerzerkennung in der gezeigten Mimik einer Person haben könnten, jedoch soll diese Annahme nicht im Voraus ausgeschlossen werden - stattdessen soll überprüft werden, ob sich diese zusätzlichen Informationen im Laufe der weiteren Untersuchung wider Erwarten als für die Klassifizierung relevant erweisen oder nicht. Zusammen mit den 61 auf die *action units* bezogenen Attributen ergeben sich damit 64 reguläre Attribute (neben der *ID* und dem *label*) für die Repräsentation der Trainingsdaten, auf die im Folgenden die eingangs genannten Algorithmen des *machine learning* angewandt werden.

# Kapitel 4

# Decision Tree Learning / ID3

Das erste der beiden untersuchten Lernverfahren ist das *Decision Tree Learning*, welches auf dem *ID3*-Algorithmus basiert. Die Anwendung des Verfahrens auf die Trainingsdaten wurde in *RapidMiner* realisiert. Im Folgenden soll zunächst das Konzept des *Decision Tree Learning* kurz vorgestellt werden; anschließend wird der Aufbau des Lernverfahrens in *RapidMiner* skizziert und der Ablauf der Untersuchung beschrieben; zuletzt werden die Ergebnisse interpretiert.

## 4.1 Grundlagen des Decision Tree Learning

Das *Decision Tree Learning* ist eine Technik des *Machine Learning*, bei der die gelernte Zielfunktion als Entscheidungsbaum repräsentiert wird. Jede neue Instanz wird dabei von der Wurzel aus bis zu einem Blatt des Baumes hinab gereicht, welches die der Instanz durch den Baum zugeordnete Klasse spezifiziert. In jedem Knoten, den die Instanz dabei passiert, wird sie auf ein bestimmtes Attribut getestet, wobei jeder von diesem Knoten ausgehende Zweig einem möglichen Attributwert für dieses Attribut entspricht - die Instanz wird dabei immer über denjenigen Zweig weitergereicht, welcher den Attributwert repräsentiert, den die Instanz für das entsprechende Attribut enthält, bis sie schließlich an einem Blatt des Baumes ankommt. Jeder Entscheidungsbaum lässt sich damit auch als *Disjunktion* von *Konjunktionen* von Attribut-Wert-Paaren repräsentieren, wobei jeder mögliche Pfad von der Wurzel zu einem Blatt einer *Konjunktion* von Wertzuweisungen an Attribute entspricht.[5]

Das *Decision Tree Learning* bietet einige Vorteile für das maschinelle Lernen: im Gegensatz zu anderen Algorithmen wird der gesamte *hypothesis space* uneingeschränkt nach einer möglichen Lösung durchsucht: es lassen sich also alle denkbaren Zielkonzepte lernen. Zudem ist das Lernverfahren auch für Außenstehende relativ gut intuitiv verständlich. Zugleich ist es dabei relativ robust gegenüber möglichen Fehlern wie falsch zugewiesenen Klassifikationen einzelner Trainingsbeispiele sowie falschen oder fehlenden Werten für einzelne Attribute.[5]

Dem *Decision Tree Learning* liegt in den meisten Fällen der *ID3*-Algorithmus oder eine Modifikation desselben zugrunde. Hierbei handelt es sich um einen *Greedy*-Algorithmus, der von jedem aktuellen Zustand der Durchführung aus immer den besten möglichen Folgezustand auswählt, ohne weitreichendere Konsequenzen in der Zukunft mit einzubeziehen. Der Algorithmus wurde 1986 von John Ross Quinlan entwickelt. Die Entscheidungsbäume werden hierbei immer *top-down*, also beginnend mit der Wurzel und schrittweise bis zu den Blättern aufbauend, konstruiert. Dabei wird schrittweise immer dasjenige Attribut für einen Knoten ausgewählt, welches die Menge der Trainingsbeispiele am besten klassifiziert. Für jeden möglichen Attributwert wird anschließend ein Zweig erstellt, an dessen Ende erneut ein Teilbaum aus der Menge der Trainingsbeispiele, für welche dieser Attributwert zutrifft, erstellt wird, wobei wieder das beste der bisher noch nicht verwendeten Attribute als nächster Knoten ausgewählt wird. Sind alle für die Erstellung eines Knotens zu berücksichtigenden Trainingsbeispiele derselben Klasse zugeordnet, kann statt einer weiteren Differenzierung die entsprechende Klasse als Blatt repräsentiert werden. Wenn die Trainingsbeispiele verschiedenen Klassen angehören, jedoch keine Attribute mehr zur weiteren Differenzierung vorhanden sind, wird dem Blatt stattdessen die häufigste Klasse innerhalb der Menge der Trainingsbeispiele in diesem Knoten zugewiesen.[5]

Zentraler Aspekt bei der Erstellung des Entscheidungsbaumes ist die Entscheidung, welches Attribut die Menge der Trainingsbeispiele am besten in Hinblick auf die Zielklassen unterteilt. Hierfür wird für jedes für einen Knoten mögliche Attribut der *information gain* berechnet: dieser beschreibt die Reduktion der *Entropie* innerhalb der Trainingsmenge durch die Aufteilung nach den entsprechenden Attributwerten - das Attribut mit dem höchsten *information gain* wird schließlich für den Knoten ausgewählt.[5]

## 4.2 Aufbau des Lernverfahrens

### 4.2.1 Aufbau in RapidMiner

Zunächst werden die Trainingsdaten in Form einer CSV-Datei mit dem *Read CSV*-Operator eingelesen. Anschließend findet eine Parameter-Optimierung statt (*Optimize Parameters*). Geschachtelt in der Parameter-Optimierung wird eine *X-Validation* durchgeführt. Diese ist unterteilt in einen *Training*- und einen *Testing*-Teil: der *Training*-Teil beinhaltet das Lernen des *Decision Tree* selbst mit dem Großteil der verfügbaren Trainingsdaten; im *Testing*-Teil wird das so gelernte Modell angewandt und mit den restlichen Trainingsbeispielen auf seine Korrektheit überprüft (*Apply Model*) - anschließend wird die *Performance* des Modells gemessen. Als Anzahl der Validierungen für die *X-Validation* wird 10 gewählt: die Menge der Trainingsdaten wird also in 10 gleich große Teilmengen unterteilt, von denen in 10 Wiederholungen immer 9 zum Training verwendet werden und die letzte zum Testen (so dass am Ende jede der 10 Teilmengen einmal als Testmenge verwendet wurde). Als Ausgabe des Programms werden ein über die gesamte Menge der Trainingsbeispiele gelernter Entscheidungsbaum, der dazugehörige *Performance-Vektor* sowie die gefundenen optimalen Parameterwerte für den Lernprozess ausgegeben.

### 4.2.2 Wahl der Parameter

Bei der Parameter-Optimierung werden für wichtige Einstellungen beim Lernen des Entscheidungsbaums verschiedene Werte getestet. Der Lernprozess wird für jede mögliche Kombination von abgefragten Werten für die Parameter wiederholt, um diejenige Werte-Kombination zu ermitteln, für die der Entscheidungsbaum mit der optimalen Performanz gelernt wird.

Für die folgenden Parameter werden Default-Werte verwendet:

*criterion*: *information_gain*

*maximal depth*: -1

*confidence*: 0.25

*number of prepruning alternatives*: 3

Für die folgenden Parameter werden verschiedene Werte getestet:

*minimal size for split*: Einzelwerte 2, 5, 10, 15, 25

*minimal leaf size*: Einzelwerte 2, 5, 10, 15

*minimal gain*: 10 Werte zwischen 0.0001 und 0.1 (quadratische Skalierung)

*no pre-pruning*: Werte *true* und *false*

*no pruning*: Werte *true* und *false*

## 4.3 Durchführung des Lernverfahrens

### 4.3.1 1. Ansatz: 3 Klassen, Standard-Performance

Im ersten Durchlauf wird das Lernverfahren auf die zuvor erstellten Trainingsdaten unverändert angewendet. Dabei ergibt sich folgende Ergebnismatrix:

**accuracy: 57.94% +/- 9.39% (mikro: 57.92%)**

| | true pain | true disgust | true neutral | class precision |
|---|---|---|---|---|
| pred. pain | 17 | 10 | 4 | 54.84% |
| pred. disgust | 25 | 83 | 28 | 61.03% |
| pred. neutral | 12 | 30 | 50 | 54.35% |
| class recall | 31.48% | 67.48% | 60.98% | |

Abbildung 4.1: Ergebnismatrix für *performance* mit 3 Klassen

Mit einer accuracy von 57.92% liegt hier ein eher schlechtes Ergebnis vor. Hinzu kommt das Problem, dass die *precision*- und *recall*-Werte für die Klasse *pain* (54.84% bzw. 31.48%) noch unter den Werten für die anderen beiden Klassen liegen. Die Differenzierung zwischen den Klassen *disgust* und *neutral*

funktioniert etwas besser, dies ist aber für die Aufgabenstellung des Projekts, die Schmerzerkennung, weniger relevant.

Um die *performance* des Lernverfahrens zu verbessern und vor allem für die Klasse *pain* bessere Werte zu erzielen, wird im Folgenden der Ansatz verfolgt, in der Klassifizierung nur noch zwischen Schmerz und nicht-Schmerz zu unterscheiden: dies wird durch die Zusammenfassung der Klassen *disgust* und *neutral* zu einer gemeinsamen Klasse *no_pain* realisiert.

### 4.3.2 2. Ansatz: 2 Klassen, Standard-Performance

Mit den im Folgenden nur noch zwei Klassen wird das Lernverfahren ein weiteres mal auf die Trainingsdaten angewendet. Die Ergebnismatrix verändert sich dadurch folgendermaßen:

**accuracy: 81.48% +/- 6.81% (mikro: 81.47%)**

| | true pain | true no_pain | class precision |
|---|---|---|---|
| pred. pain | 24 | 18 | 57.14% |
| pred. no_pain | 30 | 187 | 86.18% |
| class recall | 44.44% | 91.22% | |

Abbildung 4.2: Ergebnismatrix für *performance* mit 2 Klassen

Da jetzt nicht mehr zwischen Ekel und neutraler Klasse unterschieden wird, verbessert sich die *accuracy* deutlich auf 81.47%. Auch für die Klasse *pain* ergeben sich nun etwas bessere *precision*- und *recall*-Werte (57.14% bzw. 44.44%). Der *recall*-Wert liegt jedoch immer noch unter 50%, was für ein konkretes Anwendungsszenario weiterhin nicht akzeptabel ist. Beim praktischen Einsatz der Schmerzerkennung (z. B. im Kontext der Patientenbetreuung eines Krankenhauses) ist es wichtig, Schmerz zuverlässig identifizieren zu können; im Zweifelsfall sollte eher einmal zu oft als einmal zu wenig Schmerz prognostiziert werden, um Schäden für betroffene Personen durch nicht rechtzeitig erkannte Schmerzen zu vermeiden. Daher sollte einem hohen *recall*-Wert für die Klasse *pain* in der Untersuchung eine hohe Priorität eingeräumt werden. *RapidMiner* bietet dem Nutzer die Möglichkeit, bei der Erfassung der *performance* bestimmte Werte als *main criterion* festzulegen, denen bei der Durchführung des Lernverfahrens eine erhöhte Priorität zugewiesen wird, sofern für das Lernverfahren eine binominale Klassifizierung vorliegt - dies ist durch die Zusammenfassung der Klassen *disgust* und *neutral* der Fall. Um neben dem *recall* für die Klasse *pain* die *precision* nicht außer Acht zu lassen, wird als *main criterion* das *f-measure* ausgewählt, welches ein ausgewogenes Maß zwischen *precision* und *recall* darstellt und wie folgt berechnet wird:

$$f - measure = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

### 4.3.3 3. Ansatz: 2 Klassen (vertauscht), Fokus auf F-measure

Das in der *performance* ausgewiesene *main criterion* wird immer für eine bestimmte Klasse ausgewertet - im Falle des *f-measure* ist dies die *positive class*.

Diese ist momentan noch *no_pain*. Daher werden die Klassen zunächst neu zugewiesen. Da der Aufbau des Lernverfahrens sich darüber hinaus nicht verändert, sollten eigentlich auch die Ergebnisse gleich bleiben - dies ist jedoch nicht der Fall, wie ein Blick auf die veränderte Ergebnismatrix zeigt:

**accuracy: 81.46% +/- 5.12% (mikro: 81.47%)**

| | true no_pain | true pain | class precision |
|---|---|---|---|
| pred. no_pain | 192 | 35 | 84.58% |
| pred. pain | 13 | 19 | 59.38% |
| class recall | 93.66% | 35.19% | |

Abbildung 4.3: Ergebnismatrix für vertauschte Klassen

Die Verschlechterung der Ergebnisse, welche sich durch die Neuzuweisung ergibt, lässt sich aus dem Versuchsaufbau heraus nicht erklären, muss jedoch zunächst akzeptiert werden, um mit der Durchführung des Lernverfahrens fortzufahren - denn erst durch die veränderte Zuordnung lässt sich das *f-measure* für *pain* als *positive class* auswerten:

**f_measure: 41.68% +/- 16.99% (mikro: 44.19%) (positive class: pain)**

| | true no_pain | true pain | class precision |
|---|---|---|---|
| pred. no_pain | 192 | 35 | 84.58% |
| pred. pain | 13 | 19 | 59.38% |
| class recall | 93.66% | 35.19% | |

Abbildung 4.4: Ergebnismatrix mit *f-measure* für vertauschte Klassen

Um den beobachteten Wert für das *f-measure* von 44.19% weiter zu steigern, wird im nächsten Schritt das *f-measure* als *main criterion* für die *performance* ausgewählt - dies führt jedoch zu folgendem Ergebnis:

**accuracy: 79.15% +/- 1.83% (mikro: 79.15%)**

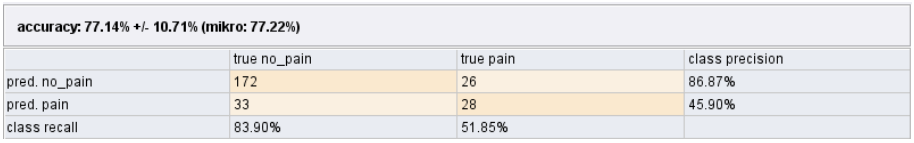
| | true no_pain | true pain | class precision |
|---|---|---|---|
| pred. no_pain | 205 | 54 | 79.15% |
| pred. pain | 0 | 0 | 0.00% |
| class recall | 100.00% | 0.00% | |

Abbildung 4.5: Ergebnismatrix für *f-measure* als *main criterion*

Dieses Vorgehen ist nicht zielführend, da nun alle Trainingsbeispiele als *no_pain* klassifiziert werden und *precision* und *recall* für *pain* jeweils 0.00% betragen - dadurch ist auch das *f-measure* nicht berechenbar. Im Folgenden wird daher als Alternative zum *f-measure* der *recall* der Klasse *pain* als *main criterion* ausgewählt.

### 4.3.4 4. Ansatz: 2 Klassen (vertauscht), Fokus auf Recall

Die Ergebnismatrix für *pain* als *positive class* mit dem *recall* als *main criterion* gestaltet sich wie folgt:



| accuracy: 77.14% +/- 10.71% (mikro: 77.22%) | | | |
|---|---|---|---|
| | true no_pain | true pain | class precision |
| pred. no_pain | 172 | 26 | 86.87% |
| pred. pain | 33 | 28 | 45.90% |
| class recall | 83.90% | 51.85% | |

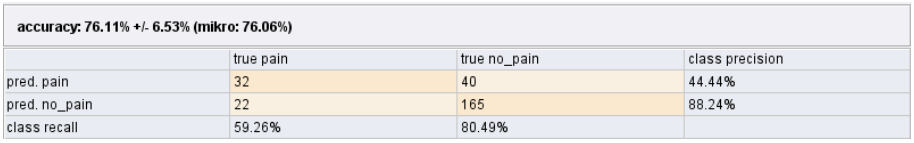
Abbildung 4.6: Ergebnismatrix für *recall* als *main criterion*

Tatsächlich wird hier eine Verbesserung des *recalls* für *pain* auf Kosten der *precision* erzielt. Dadurch verbessert sich auch der ursprünglich angestrebte Wert des *f-measure* auf 48.70%.

An dieser Stelle führen weitere Tests und Modifikationen des Versuchsaufbaus zu keiner weiteren Verbesserung der Ergebnisse. Eizig die Verschlechterung der Ergebnisse durch die Neuzuweisung der Klassen mit *pain* als *positive class* lässt sich für diese Konstellation noch rückgängig machen: das *main criterion recall* lässt sich zwar nur für die *positive class* einstellen, es gibt es jedoch ein äquivalentes *main criterion*, welches den *recall* für die *negative class* repräsentiert: die *specificity*. Dementsprechend wird im Folgenden die Neuzuweisung der Klassen rückgängig gemacht und die *specificity* für die Klasse *pain* (welche nun wieder die *negative class* ist) als *main criterion* ausgewählt.

### 4.3.5 5. Ansatz: 2 Klassen, Fokus auf Specificity

Für die ursprüngliche Klassenzuweisung mit *pain* als *negative class* und der *specificity* als *main criterion* ergibt sich folgende Ergebnismatrix:



| accuracy: 76.11% +/- 6.53% (mikro: 76.06%) | | | |
|---|---|---|---|
| | true pain | true no_pain | class precision |
| pred. pain | 32 | 40 | 44.44% |
| pred. no_pain | 22 | 165 | 88.24% |
| class recall | 59.26% | 80.49% | |

Abbildung 4.7: Ergebnismatrix für *specificity* als *main criterion*

Die *specificity*, also der *recall* für die *negative class pain*, verbessert sich, allein durch das Rückgängig-Machen der die Ergebnisse verschlechternden Neuzuweisung der Klassen, noch einmal auf 59.26%, während die *precision* für *pain* sowie die allgemeine *accuracy* gleichzeitig kaum abnehmen. Darüber hinaus lässt sich jedoch wie bei der vorigen Klassenzuweisung keine weitere Verbesserung der Ergebnisse erzielen. Damit ergeben sich auch für die beste mögliche Anordnung nur mittelmäßige Ergebniswerte.

## 4.4 Auswertung der Ergebnisse

Die *performance* für das *Decision Tree Learning* mit den durch unsere Gruppe ausgewählten Attributen führt nur zu mittelmäßigen Ergebnissen; besonders für die Klasse Schmerz selbst liegen nur geringe *precision*- und *recall*-Werte vor,

die sich auch durch Modifikationen des Versuchsaufbaus nur bedingt verbessern lassen.

Auch der Blick auf die konkreten Entscheidungsbäume wirft Fragen auf: die Basis ist bei allen Bäumen noch relativ gleich aufgebaut: das Attribut *au7_max_intensity* stellt bei allen Bäumen die Wurzel und damit das Attribut mit dem höchsten Informationsgehalt dar; in den meisten Fällen folgen die Attribute *au17_frequency* und *au4_max_intensity* als Nachfolger-Knoten. Darüber hinaus ergeben sich jedoch sehr unterschiedlich stark differenzierte Baumstrukturen, deren Knoten von Baum zu Baum unterschiedliche Attribute enthalten - es lässt such also anhand der Entscheidungsbäume keine eindeutige gemeinsame Struktur und Attributauswahl für die Schmerzerkennung herausarbeiten.

Bei der Betrachtung der komplexeren Bäume fällt zudem auf, dass viele Knoten die personenbezogenen Daten Größe, Alter und Geschlecht enthalten, die intuitiv eigentlich keinen Einfluss auf die Schmerzerkennung haben dürften. Dennoch verschlechtert sich die *performance* des Lernverfahrens, wenn diese Werte bei der Durchführung weggelassen werden. Hierfür lassen sich zwei Interpretationsmöglichkeiten finden: entweder sind die personenbezogenen Daten bei der Schmerzerkennung wichtiger als man intuitiv vermuten könnte - für diese These lassen sich jedoch praktisch keine Erklärungsansätze finden; zudem sind Größe, Alter und Geschlecht bei der Aufstellung der Attribute mit dem höchsten Informationsgehalt, die im Rahmen des KNN-Verfahrens im folgenden Kapitel erstellt wurde, nicht vertreten. Wahrscheinlicher ist die Erklärung, dass für das Lernverfahren eine für die Schmerzerkennung unzureichende Attributauswahl vorliegt oder aber das Lernverfahren des *Decision Tree Learning* selbst für die Schmerzerkennung ungeeignet ist.

Auch bei der Betrachtung der gefundenen optimalen Parameter für die verschiedenen Lerndurchläufe lassen sich kaum Gemeinsamkeiten finden, so dass sich wie auch bei den Entscheidungsbäumen keine klare Struktur abzeichnet.

Insgesamt hat das Verfahren des *Decision Tree Learning* mit der gegebenen Attributauswahl deutliche Probleme mit der Schmerzerkennung. Die Baumstruktur und die Auswahl der optimalen Parameter erscheinen im Vergleich der verschiedenen Ansätze eher willkürlich. Ein Grund für die schlechten Ergebnisse könnte die Tatsache sein, dass bei der Attributauswahl die zeitlichen Informationen wie die *duration* oder die Aufeinanderfolge der *action units* bewusst weggelassen wurden - dies ist im Vergleich mit den Ergebnissen der anderen Projektgruppen genauer zu überprüfen. Ob für die vorliegende Attributauswahl mit einem anderen Lernverfahren bessere Ergebnisse erzielt werden können, soll im folgenden Kapitel untersucht werden.

# Kapitel 5

# K-nächste-Nachbarn Algorithmus

## 5.1 Grundlagen

Der k-nächste-Nachbarn Algorithmus (engl. k-nearest-neighbor algorithm, kurz KNN) ist ein Klassifikationsverfahren. Die Grundidee des Algorithmus ist, dass naheliegende Objekte ähnliche Merkmalsausprägungen besitzen. Entsprechend geschieht die Zuordnung von nicht klassifizierten Instanzen basierend auf der Menge der k umliegenden (bereits klassifizierten) Nachbarinstanzen. Die Distanz als Maß zur Nähe des nächsten Nachbarn wird unter anderem als Euclidean Distance, Minkowski Distance oder Manhattan Distance gemessen.

Drei Phasen werden durchlaufen:
Erste Phase: Berechnung der Distanz zu allen Nachbarn
Zweite Phase: Bestimmung der k nächsten Nachbarn
Dritte Phase: Zuweisung der Klasse abhängig vom häufigsten Klassenvorkommen der in der zweiten Phase bestimmten Nachbarn

KNN erlangt im Vergleich mit anderen Algorithmen seiner Klasse relativ gute Ergebnisse. Gleichzeitig gibt es einen größenvariablen Hypothesenraum, flexible Entscheidungsgrenzen, und es ist keine Lernphase nötig. Vom Typ ist KNN ein lazy-learning Algorithmus und die Generalisierung des Trainingbeispiels findet erst statt wenn ein neues Objekt klassifiziert werden muss.
Nachteile sind der hohe Rechenaufwand und die hohe Klassifizierungszeit. Es gilt irrelevante und korrelierte Eigenschaften zu eliminieren, denn diese haben eine große Auswirkung auf das Ergebnis.
Wichtig ist die Bestimmung des Parameters k. Bei einem k von 1 gleicht der Ergebnisgraph einem Voronoi-Diagramm. Wird das k zu klein gewählt, kann Rauschen in den Trainingsdaten das Ergebnis verschlechtern. Ist das k jedoch zu groß gewählt, werden unter Umständen zu weit entfernte Objekte zur Bestimmung benutzt, was das Ergebnis entsprechend verschlechtern kann. Um Klassifizierungsprobleme zu vermeiden, sollte k stets ungerade gewählt werden.

## 5.2 Anordnungen und Ergebnisse

Im Folgenden werden sieben Anordnungen vorgestellt, bei denen die Parameter optimiert werden. Das jeweils beste Ergebnis wird präsentiert.

### 5.2.1 Anordnung 1

In der ersten Anordnung findet eine Crossvalidation (deutsch: Kreuzvalidierung) mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 260. Für den Algorithmus werden gemischte Maße als Einheit und eine gemischte euklidische Distanz gewählt. Einheit und Distanz wurden so gewählt, weil nominale Werte zusätzlich zu numerischen Werten in den Daten vorkommen.

### 5.2.2 Ergebnis von Anordnung 1

Bei einem k-Wert von 14 beträgt die accuracy = 51.74% +/- 10.76% .

### 5.2.3 Anordnung 2

Anordnung 2 basiert auf der Idee, dass Ekel und Neutrale Klasse als eine Klasse verarbeitet werden. Somit unterteilen sich die Zielklassen nur noch in Schmerz oder kein-Schmerz.

In dieser Anordnung findet eine Crossvalidation mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 260. Für den Algorithmus werden gemischte Maße als Einheit und eine gemischte euklidische Distanz gewählt. Zusätzlich Merge (Vereinigung) von Ekel und Neutraler Klasse.

### 5.2.4 Ergebnis von Anordnung 2

Bei einem k-Wert von 11 beziehen sich folgende Werte auf die Klasse kein-Schmerz.
Accuracy: 79.57% +/- 4.48%
Precision: 80.95% +/- 3.29%
Recall: 97.05% +/- 3.29%
Folgende Werte beziehen sich auf die Schmerz Klasse.
precision: 53.85%
recall: 12.96%

### 5.2.5 Anordnung 3

Anordnung 3 basiert auf der Idee, dass Attribute nach ihrem Information Gain gewichtet werden.

In dieser Anordnung findet eine Crossvalidation mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 260. Für den Algorithmus werden gemischte Maße als Einheit und eine gemischte euklidische Distanz gewählt. Zusätzlich findet eine Gewichtung nach Infomation Gain (mit normalisierten Gewichten) statt. Die Attribute werden nach dem Information Gain skaliert.

### 5.2.6 Ergebnis von Anordnung 3

Bei einem Wert von k = 26 beträgt die accuracy = 60.29% +/- 7.51% .

| | |
|---|---|
| au7_frequency | 1.0 |
| au7_max_intensity | 1.0 |
| au4_frequency | 0.8483239497896657 |
| au4_max_intensity | 0.8483239497896657 |
| au6_max_intensity | 0.6507619849257551 |
| au6_frequency | 0.5873113570372117 |
| au10_frequency | 0.529505474438484 |
| au10_max_intensity | 0.529505474438484 |
| au43_frequency | 0.41984116210033157 |
| au12_frequency | 0.3276917293507933 |
| au12_max_intensity | 0.3276917293507933 |
| au19_frequency | 0.31790684975298833 |
| au19_max_intensity | 0.31790684975298833 |
| au28_frequency | 0.3156896142950439 |
| au28_max_intensity | 0.3156896142950439 |
| au15_frequency | 0.2973363880136888 |
| au15_max_intensity | 0.2973363880136888 |
| au1_max_intensity | 0.2963858855308812 |
| au1_frequency | 0.22795536238740358 |
| au24_frequency | 0.21793026782766609 |
| au24_max_intensity | 0.21793026782766609 |

Abbildung 5.1: Top 21 Attribut Gewichte nach Information Gain

### 5.2.7 Anordnung 4

Anordnung 4 basiert auf der Idee, dass unwichtige Attribute bezüglich des Information Gain weglassen werden.

In dieser Anordnung findet eine Crossvalidation mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 260. Für den Algorithmus werden gemischte Maße als Einheit und eine gemischte euklidische Distanz

gewählt. Zusätzlich findet eine Gewichtung nach Infomation Gain (mit normalisierten Gewichten) statt. Die Attribute werden nach dem Information Gain ausgewählt (Anzahl der ausgewählten Gewichte werden optimiert im Bereich von 1-64 Werten bei 64 Schritten).

### 5.2.8 Ergebnis von Anordnung 4

Bei einem Wert von k = 35 beträgt die accuracy = 64.86% +/- 7.59% . 21 Attribute wurden als besonders wichtig für die Auswertung gefunden, diese sind in 5.1 auf Seite 21 zu sehen.

Die wichtigsten Action Units nach Information Gain:
7 Zusammenziehen des inneren Teils des Ringmuskels um die Augen
4 Zusammenziehen der Augenbrauen
6 Zusammenziehen des äußeren Teils des Ringmuskels um die Augen
10 Anheben der oberen Lippe
43 Absenken des oberen Augenlides (43E = Augen geschlossen)
12 Anheben der Mundwinkel wie beim Lächeln oder Lachen
28 Lippen zwischen die Zähne saugen
15 Herabziehen der Mundwinkel
1 Heben der Augenbrauen innen
24 Zusammenpressen der Lippen

### 5.2.9 Anordnung 5

In der fünften Anordnung findet eine Crossvalidation mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 70. Für den Algorithmus werden numerische Einheiten und eine euklidische Distanz gewählt. Einheit und Distanz werden so gewählt, weil nur noch numerische Werte in den Daten vorkommen. Die Hauptkomponentenanalyse arbeitet nur mit numerischen Werten. Bei der Analyse wird dimensionality reduction (deutsch: Dimensionalitätsreduktion) optimiert für die Werte none, fixed number, keep variance. Bei none wird die Dimensionalität nicht reduziert. Für fixed number wird die Zahl der Komponenten im Bereich von 1-63 in Zehnerschritten quadratisch optimiert. Für keep variance wird nach dem variance threshold im Bereich von 0.0-1.0 in Zehnerschritten quadratisch optimiert. Als Folge für die Umstellung der Einheiten und der Distanz ergibt sich eine Änderung der Daten. Zwei Sequenzen mit fehlenden Werten (fehlende Werte sind bei der Hauptkomponentenanalyse nicht erlaubt) und das Attribut Geschlecht (da nicht numerisch) werden gestrichen.

### 5.2.10 Ergebnis von Anordnung 5

Ergebnis nach dimensonality reduction.
None:
Bei einem Wert von k = 13 beträgt die accuracy = 52.55% +/- 10.51% .
Fixed number:
Bei einem Wert von k = 1 und 6 Komponenten beträgt die accuracy = 58.38% +/- 11.72% .
Keep variance:

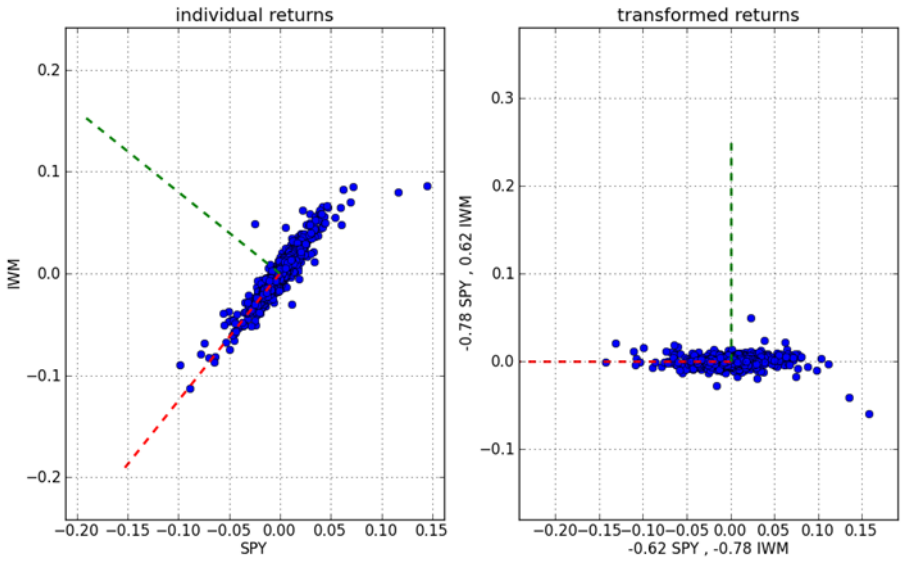



Abbildung 5.2: Grafisches Beispiel einer Hauptkomponentenanalyse

Bei einem Wert von k = 13 und einem variance threshold von 0.0 beträgt die accuracy = 52.55% +/- 10.51% .

### 5.2.11 Anordnung 6

Ähnlich wie in Anordnung 2 basiert Anordnung 6 auf der Idee, dass Ekel und Neutrale Klassen als eine Klasse verarbeitet werden. Somit unterteilen sich die Zielklassen nur noch in Schmerz oder kein-Schmerz. Zusätzlich findet eine Hauptkomponentenanalyse statt.

In dieser Anordnung findet eine Crossvalidation mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 70. Es gibt eine Vereinigung von Ekel und Neutraler Klasse. Für den Algorithmus werden numerische Einheiten und eine euklidische Distanz gewählt. Bei der Analyse wird dimensionality reduction optimiert für die Werte none, fixed number, keep variance. Bei none wird die dimensonality nicht reduziert. Für fixed number wird die Zahl der Komponenten im Bereich von 1-63 in Zehnerschritten quadratisch optimiert. Für keep variance wird nach dem variance threshold im Bereich von 0.0-1.0 in Zehnerschritten quadratisch optimiert.

### 5.2.12 Ergebnis von Anordnung 6

Ergebnis nach dimensonality reduction. Die Auswertung erfolgt für die kein-Schmerz Klasse.
None:
Bei einem Wert von k = 24 beträgt accuracy = 80.17% +/- 1.94%, precision = 80.09% +/- 1.93% und recall= 100.00% +/- 0.00%.

Fixed number:
Bei einem Wert von k = 10 und 54 Komponenten beträgt accuracy = 81.31% +/- 5.22%, precision = 81.85% +/- 3.89% und recall= 98.50% +/- 3.20%.
Keep variance:
Bei einem Wert von k = 24 und einem variance threshold von 0.0 beträgt accuracy = 80.17% +/- 1.94%, precision = 80.09% +/- 1.93% und recall= 100.00% +/- 0.0%.

### 5.2.13 Anordnung 7

Anordnung 7 basiert auf der Idee, dass Attribute bezüglich der Hauptkomponentenanalyse gewichtet werden.

In dieser Anordnung findet eine Crossvalidation mit 10 Teilen statt. Parameter k wird optimiert mit Werten im Bereich von 1 bis 70. Für den Algorithmus werden numerische Einheiten und eine euklidische Distanz gewählt. Bei der Hauptkomponentenanalyse wird dimensionality reduction optimiert für die Werte none, fixed number, keep variance. Bei none wird die dimensonality nicht reduziert. Für fixed number wird die Zahl der Komponenten im Bereich von 1-63 in Zehnerschritten quadratisch optimiert. Für keep variance wird nach dem variance threshold im Bereich von 0.0-1.0 in Zehnerschritten quadratisch optimiert. Anschließend werden die Werte nach den (normalisierten) Gewichten der Hauptkomponentenanalyse gewichtet.

### 5.2.14 Ergebnis von Anordnung 7

Ergebnis nach dimensonality reduction.
None:
Bei einem Wert von k = 13 beträgt accuracy = 52.55% +/- 10.51%.
Fixed number:
Bei einem Wert von k = 1 und 6 Komponenten beträgt accuracy = 58.38% +/- 11.72%.
Keep variance:
Bei einem Wert von k = 13 und einem variance threshold von 0.0 beträgt accuracy = 52.55% +/- 10.51%.

### 5.2.15 Fazit

Übersicht:
Anordnung, accuracy, k, Komponentenzahl
A1: 51.74% +/- 10.76%, k=14
A2: 79.57% +/- 4.48%, k=11
A3: 60.29% +/- 7.51%, k=26
A4: 64.86% +/- 7.59%, k=35
A5: 58.38% +/- 11.72%, k=1, 6
A6: 81.31% +/- 5.22%, k=10, 54
A7: 58.38% +/- 11.72%, k=1, 6

Die Schmerzerkennung ist nicht sehr überzeugend, den besten accuracy Wert (= 81.31% +/- 5.22%) bekommt man nach Anordnung 6. Auch in Anordnung 2 gibt es eine hohe accuracy. Diese Anordnungen teilen die Tatsache, dass es nur zwei Zielklassen zur Unterscheidung gibt. Den nächstbesten Wert findet man bei Anordnung 4, wo nur die nach dem Information Gain wichtigsten Attribute zur Auswertung genutzt werden. Mögliche Ursachen für dieses Ergebnis:
Das Verfahren ist nicht geeignet für die Schmerzerkennung. Die gewählten Attribute sind nicht aussagekräftig genug (Beispiel: Größe, Alter, Geschlecht ...)/ eventuell zu viele irrelevante/falsche Attribute. Schmerzvorhersage allein anhand des Gesichts ist unmöglich.

# Kapitel 6

# Fazit & Ausblick

Die durch die Projektgruppe vorgenommene Attributauswahl liefert sowohl für das *Decision Tree Learning* als auch für das *KNN*-Verfahren nur unzureichende Ergebnisse. Hierfür können verschiede Interpretationen angeführt werden: zum einen liegen möglicherweise mit den Verfahren *Decision Tree Learning* und *KNN* für die Schmerzerkennung ungeeignete Lernverfahren vor; ein weiteres Problem könnte die durch die Projektgruppe vorgenommene Attributauswahl sein, der aufgrund der Auslassung bestimmter Bestandteile der ursprünglichen Daten wichtige Informationen für eine zuverlässige Schmerzerkennung fehlen könnten (hier sind vor allem die zeitlichen Informationen zu den *action units* wie Dauer oder Abfolge zu nennen, welche in dieser Arbeit im Gegensatz zu den Arbeiten der anderen Projektgruppen nicht verwendet wurden). Auch Rauschen in den vorliegenden Trainingsdaten könnte die Ergebnisse zusätzlich verschlechtern.

Um die Ursache für die schlechten Ergebnisse näher zu erforschen, sind weitere Untersuchungen erforderlich: Zum einen sind die vorhandenen Daten mit anderen Lernverfahren zu verarbeiten, um eine Aussage über die Qualität der Lernverfahren für die Schmerzerkennung an sich treffen zu können; auch die bereits verwendeten Lernverfahren mit einer anderen Attributauswahl als Trainingsgrundlage sollten erneut überprüft werden, um Fehler oder fehlende relevante Informationen in der Repräsentation der Datenbasis auszuschließen - hierfür bietet sich eine übergreifende Analyse der Ergebnisse der verschiedenen Projektgruppen im Rahmen des Projektes „Das Schmerzgesicht“ an, da andere Gruppen zum Teil andere Informationen in die Repräsentation der Daten mit einbeziehen (z. B. zeitliche Informationen zu den *action units* wie Dauer oder Abfolge). Über alle Projektarbeiten hinweg ließe sich zudem überprüfen, ob in den vorhandenen Daten Rauschen vorhanden ist, welches die Ergebnisse unabhängig von Repräsentation und Lernverfahren beeinträchtigt - die betroffenen Instanzen sind dabei herauszufiltern und das Lernverfahren mit den übrigen, unbeeinträchtigten Daten erneut durchzuführen.

Insofern sich durch eine Kombination der zuvor beschriebenen Ansätze Ergebnisse mit einer hinreichend hohen *performance* finden lassen, ließe sich die Schmerzerkennung aus der Mimik des menschlichen Gesichts durch Maschinen in zahlreichen praktischen Anwendungsszenarien einsetzen: ein zentrales

Feld wäre der medizinische Bereich (z. B. frühe und zuverlässige Schmerzerkennung im Krankenhaus), weitere Anwendungsbereiche finden sich im Straßenverkehr, in der Robotik oder im Entertainment-Bereich (Filme, Videospiele etc.). Dementsprechend ist die weitere Forschung auf dem Gebiet der maschinellen Schmerzerkennung sowohl unter wissenschaftlichen als auch unter praktischen Gesichtspunkten erstrebenswert.

# Literaturverzeichnis

[1] Cherkassky, V. and Mulier, F. (2007). *Learning from data*. John Wiley & Sons.

[2] Ekman, P. (1988). *Gesichtsausdruck und Gefühl*. Junfermann-Verlag.

[3] Flach, P. (2012). *Machine Learning*. Cambridge University Press.

[4] Hofmann, R. (2010). Data mining mit dem rapidminer. In Haneke, U., Trahasch, S., Hagen, T., and Lauer, T., editors, *Open Source Business Intelligence*, chapter 3.7, pages 214–231. Carl Hanser Verlag.

[5] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

# Anhang A

# Ergebnisse für Decision Tree Learning

## A.1 Optimale Parameter

### A.1.1 1. Ansatz: 3 Klassen, Standard-Performance

- *minimal_gain* = 0.005
- *no_pre_pruning* = false
- *no_pruning* = false
- *minimal_size_for_split* = 2
- *minimal_leaf_size* = 10

### A.1.2 2. Ansatz: 2 Klassen, Standard-Performance

- *minimal_gain* = 0.0013
- *no_pre_pruning* = false
- *no_pruning* = true
- *minimal_size_for_split* = 25
- *minimal_leaf_size* = 5

### A.1.3 Vertauschte Klassen, Standard-Performance

- *minimal_gain* = 0.0445
- *no_pre_pruning* = false
- *no_pruning* = false
- *minimal_size_for_split* = 15
- *minimal_leaf_size* = 5

### A.1.4 3. Ansatz: 2 Klassen (vertauscht), Fokus auf F-Measure

- *minimal_gain* = 0.1
- *no_pre_pruning* = false
- *no_pruning* = true
- *minimal_size_for_split* = 2
- *minimal_leaf_size* = 2

### A.1.5 4. Ansatz: 2 Klassen (vertauscht), Fokus auf Recall

- *minimal_gain* = 0.0309
- *no_pre_pruning* = true
- *no_pruning* = true
- *minimal_size_for_split* = 15
- *minimal_leaf_size* = 5

### A.1.6 5. Ansatz: 2 Klassen, Fokus auf Specificity

- *minimal_gain* = 0.0001
- *no_pre_pruning* = true
- *no_pruning* = true
- *minimal_size_for_split* = 15
- *minimal_leaf_size* = 10

## A.2 Entscheidungsbäume

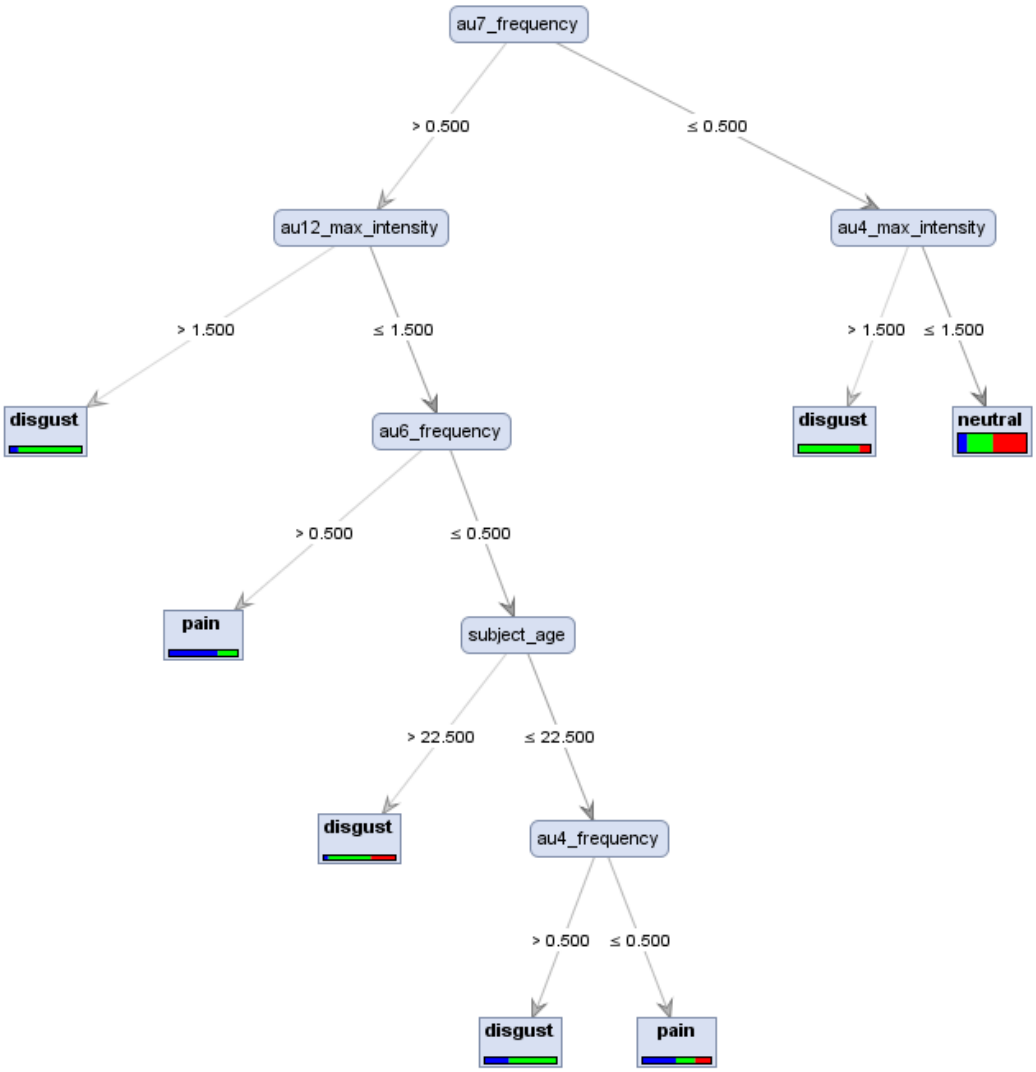



Abbildung A.1: Entscheidungsbaum für 3 Klassen

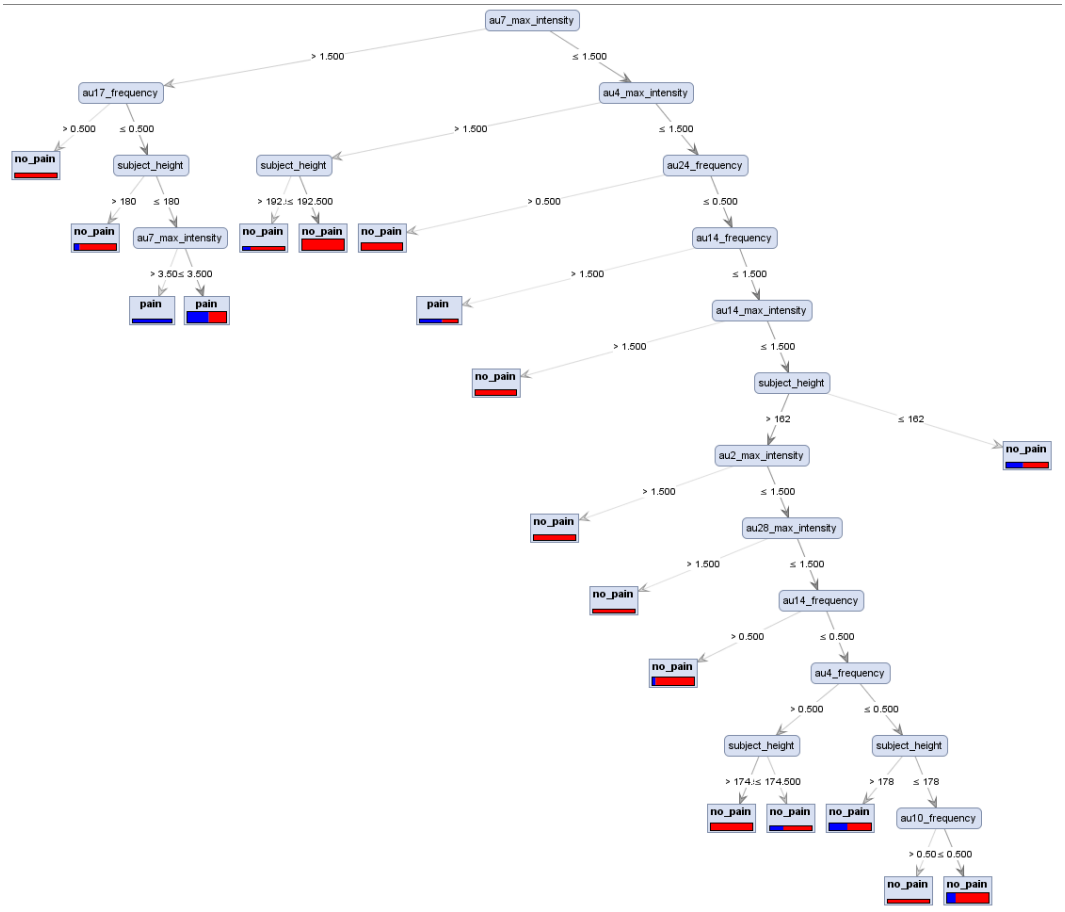



Abbildung A.2: Entscheidungsbaum für 2 Klassen

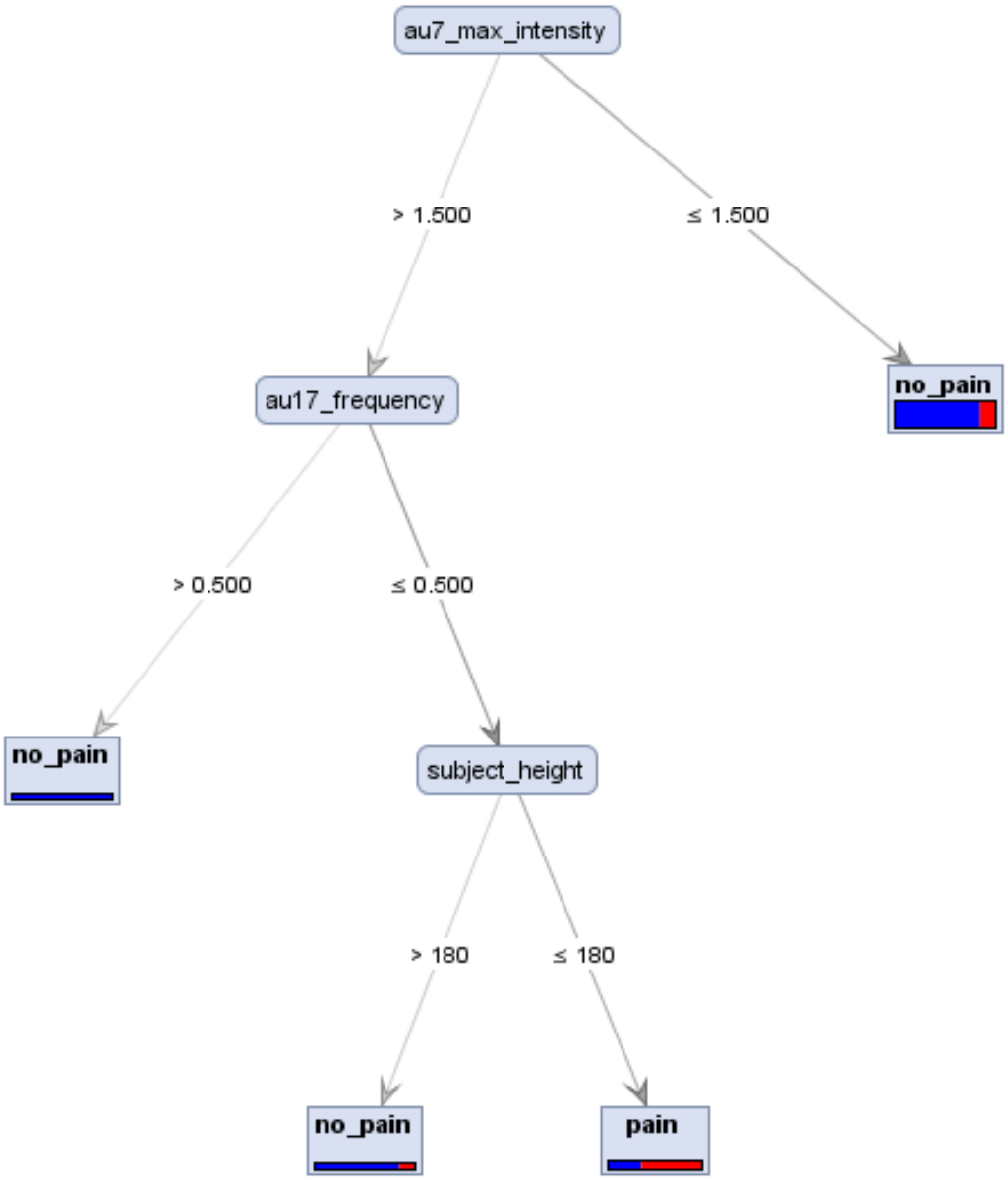



Abbildung A.3: Entscheidungsbaum für vertauschte Klassen


Abbildung A.4: Entscheidungsbaum für *f-measure* als *main criterion*

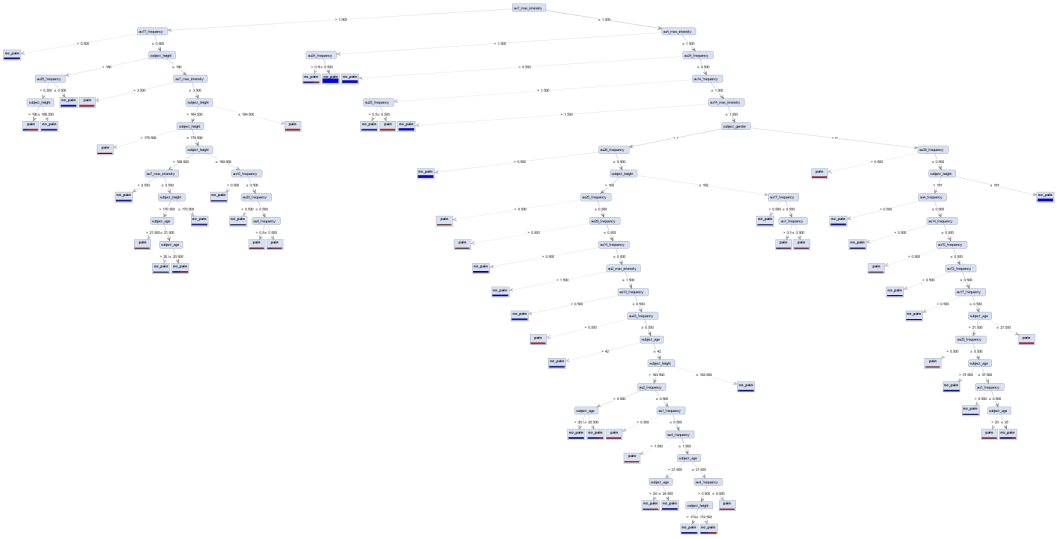


Abbildung A.5: Entscheidungsbaum für *recall* als *main criterion*

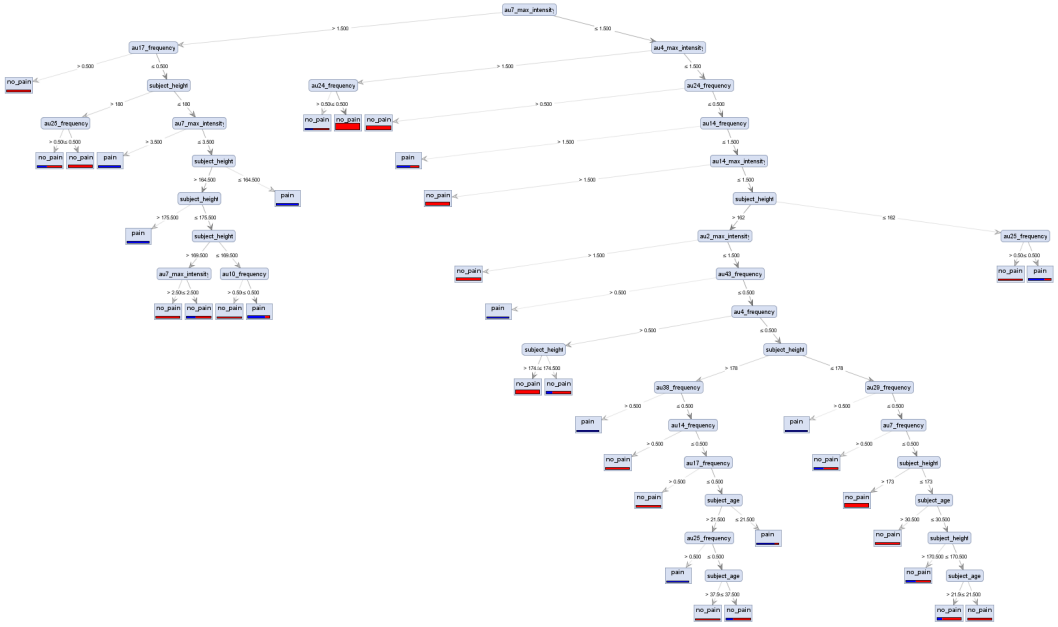



Abbildung A.6: Entscheidungsbaum für *specificity* als *main criterion*