EFME 2010 LU Exercise 2

Report

Gruppe 15:

Fritz Daniel - 0507049 - 935

Hiller Elias - 0525787 - 935

Sonderegger Josef - 0501625 - 935

1. k-NN Classifier - Leave-one-out cross-validation

Zunächst wurden die Bilder "Apple", "Bat", "Beetle", "Bone" und "Key" eingelesen und deren Eigenschaften aus den regionprops in *STATS* gespeichert. Aus den Regionprops verwendeten wir formfactor, roundness, compactness, aspectratio, perimeter, majoraxis und minoraxis, welche zusammen unseren Featurevektor *S* bilden. Außerdem werden die Bilderklassen in integer umgewandelt (von 1 bis 5) und in *SC* gespeichert, was für die Zuordnung benötigt wird.

Beschreibung

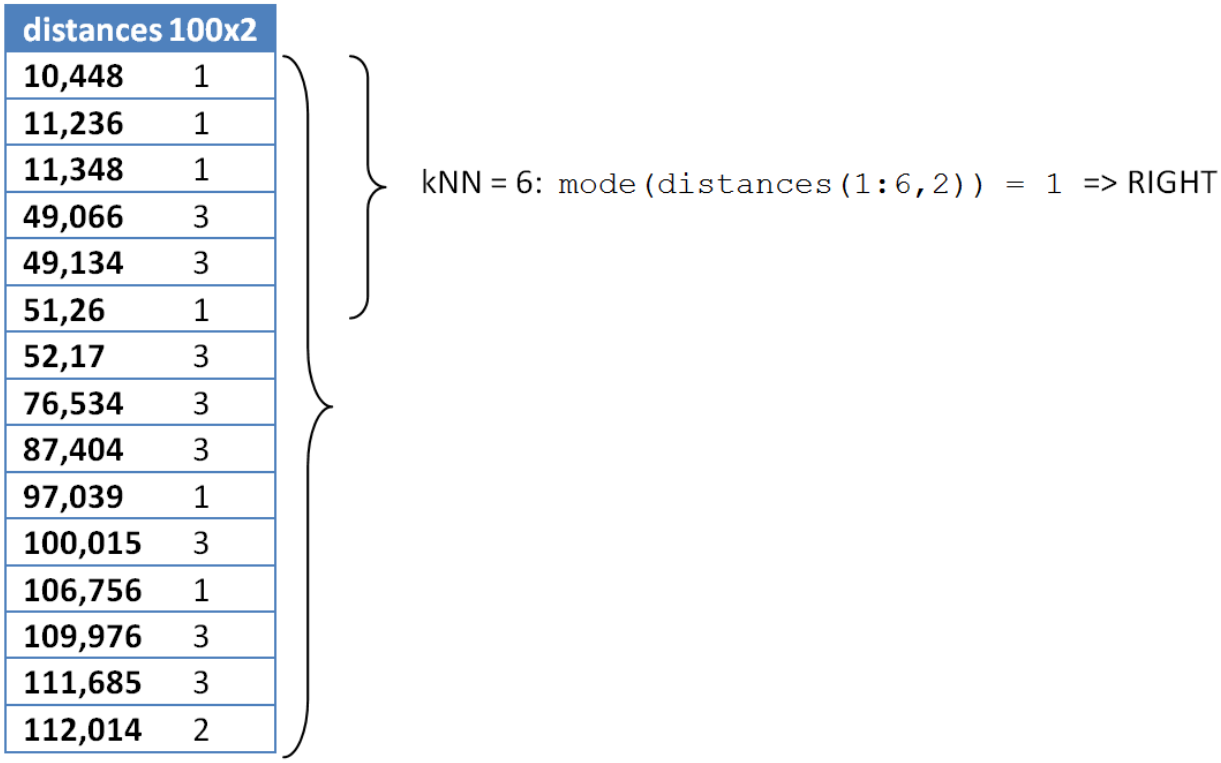
Der eigentliche kNN Algorithmus wurde mit der "leave-one-out-cross-validation" Methode implementiert. Diese Methode wird verwendet weil zu wenig Daten zur Verfügung stehen (für jede Klasse 20 verschiedene Bilder) und es dadurch nicht möglich ist eine eine gerechte Aufteilung in Trainings- und Testset zu machen. Bei der Leave-one-out cross-validation wird ein Bild als Testset, alle restlichen Bilder als Trainingsset verwendet. Jedes der Bilder wird einmal als Testset verwendet.

```
1. Train with x2, x3, x4, ..., xn, test with x1.
2. Train with x1, x3, x4, ..., xn, test with x2.
3. Train with x1, x2, x4, ..., xn, test with x3.
4. and so on ...
```

Um nun zu berechnen, welcher Klasse das Testset angehört, wird die euklidische Distanz zu jedem Bild im Trainingsset berechnet (erfolgt über den absoluten Betrag der aufsummierten Differenz der beiden Featurevektoren) und außerdem die zugehörige Bilderklasse im Vektor *eucl* gespeichert. Der Vektor wird aufsteigend nach der euklidischen Distanz sortiert und im Vektor *distance* gespeichert. Da es auch zum Vergleich von den gleichen zwei Bildern kommt und hier die Distanz 0 wäre, wird in diesem Fall die Distanz auf *realmax* gesetzt und somit stört der Wert beim sortieren nicht, da er immer an letzter Stelle sein wird. Jener Klasse, die am häufigsten in den *kNN* nächsten Nachbarn vorkommt, wird das Testbild zugeordnet. Für die Ermittlung der häufigsten Klasse wurde die Funktion *mode* in Matlab verwendet da unsere Klassen integer Zahlen sind und jeweils im sortierten *distances* Vektor stehen.

Beispiel:

Testbild entspricht der Klasse 1 (nur schematische Abbildung des Vektors *distances).*



Um das Ergebnis und die Qualität des Classifiers anschaulich zu präsentieren, berechnen wir uns mittels *error test* die Fehlerqoute für alle verschiedenen *kNN's*. Wenn ein Bild falsch klassifiziert wurde, wird eine 1 in *errors* geschrieben. Für die Fehlerqoute wird der Inhalt von *errors* in *sumerrors* spaltenweise aufsummiert (ergibt z.B. in Spalte 1 den gesamt Fehler für *kNN* = 1) und von 1 - *kNN* ausgegeben (siehe Abbildung 1).

Resultate

In der Abbildung geben wir die Fehlerquote für *kNN* von 1 bis 99 aus. Bis zu einem *kNN* von 34 bleibt die Fehlerqoute unter 30%, danach steigt sie rapide an auf etwa 70%. Ab einem Wert von 76 wird jedes Bild einer falschen Klasse zugewiesen. Wenn das *kNN* einen zu hohen Wert annimmt, kann es nicht mehr der richtigen Klasse zugewiesen werden. Bei einem *kNN* von 99 sind in den Nachbarn (*distances)* 19 Bilder der richtigen Klasse und je 20 Bilder von falschen Klassen, eine korrekte Zuweisung ist somit nicht mehr möglich.

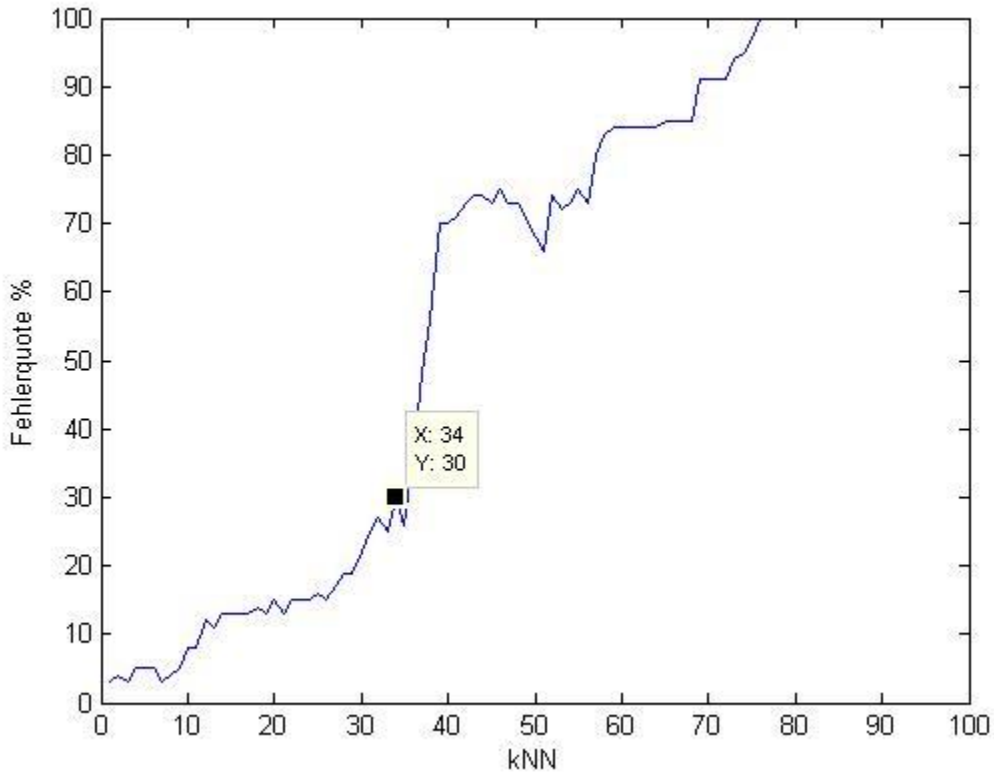


Abbildung 1: Fehlerquote in Abhängigkeit von *kNN*

Wir haben des weiteren die Werte Majoraxislength und Minoraxislength mit Orientation und Eccentricity getauscht um zu sehen, wie sich die Fehlerqoute dadurch verändert (Abbildung 2).
Es ist auffällig, dass bei einem k kleiner als 25 eine deutlich höhere Fehlerquote zu erkennen ist. Bei einem höheren k ist die Fehlerquote wie auch in Abbildung 1 sehr hoch. Um eine gute Klassifizierung zu erreichen ist also die Wahl der Features von essenzieller Bedeutung.

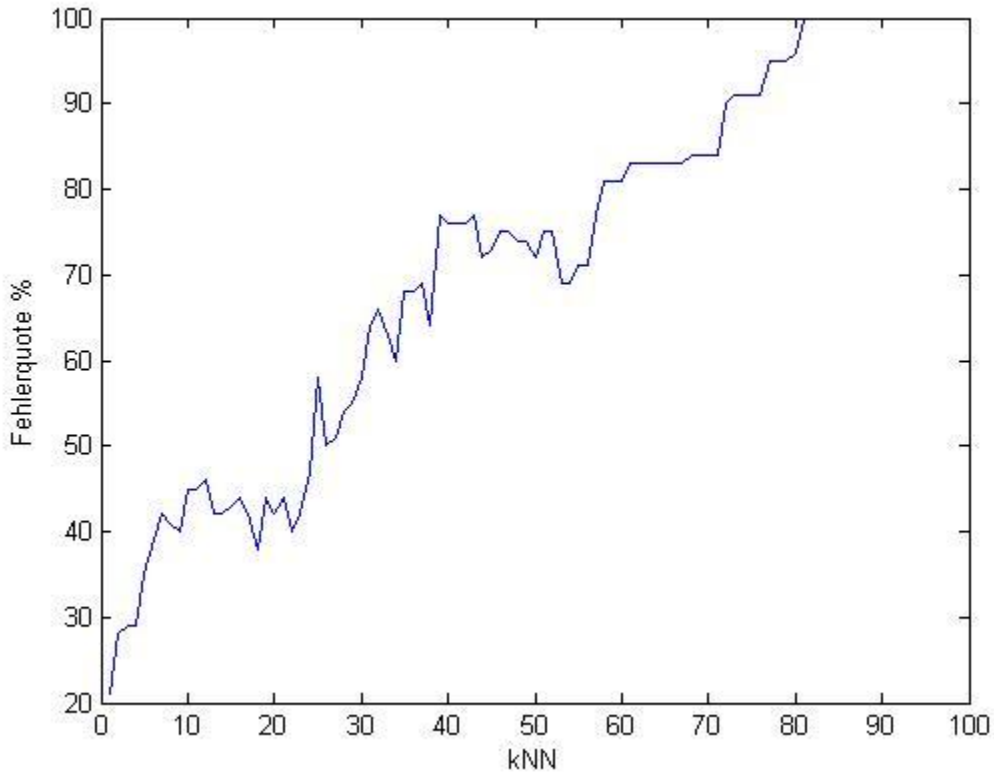


Abbildung 2: Fehlerquote in Abhängigkeit von *kNN*

Verglichen mit der ersten Übungsaufgabe ist diese Klassifizierung nicht so genau, allerdings haben wir damals die Features von Hand ausgewählt (deshalb 0% Fehlerquote), was bei einem realistischen Beispiel nicht möglich oder mit einem immensen und nicht portierbaren Aufwand verbunden wäre.

2. Content-based Image Retrieval - EFME Google

Beschreibung

Bei dieser Aufgabe wurden bereits vorhandene Featurevektoren als Trainings und Testset zur Verfügung gestellt. Ziel war es, diese Vektoren auf zweierlei Arten zu klassifizieren. Zum Einen, ob die Bilder etwas natürliches oder etwas vom Menschen gemachtes darstellen (nature und human), zum Zweiten welcher der sieben Klassen (grass, sky, foliage, brickface, cement, window, path) sie angehören.

Um dies zu bewerkstelligen berechnen wir für jedes der 2100 Bilder des Testsets die euklidische Distanz zu allen 210 Bildern im Trainingsset, unter Berücksichtigung der in der Datei zur Verfügung gestellten Featurevektoren.

Zunächst wird in einer Cell *eucl* den Distanzen die Klassennummer zugewiesen und schließlich nach Entfernung zum derzeitigen Bild sortiert *distances*. Daraufhin wird mittels

der Funktion `mode()` die meist-vorkommende Klasse im Bereich des kNN berechnet und mit dem tatsächlichen Wert verglichen.

Wie bereits oben beschrieben, und darum auch nur hier kurz der Vollständigkeit halber erwähnt, wird daraufhin die Anzahl der Fehler pro kNN, für beide Fälle (Mensch-Natur, Bildtyp) bis zum definierten Wert ausgerechnet und schließlich in einer Grafik ausgegeben.

Berechnung der Natur - Mensch Unterscheidung:
Variante 1:

Zu Beginn jeder Iteration wird zusätzlich zu jeder Distanz und der Definition des Klassentyps zusätzlich definiert, ob es sich um ein Naturbild handelt, oder um ein von Menschen gemachter Gegenstand (1 oder 2). Bei der Sortierung der Distanzen werden auch diese Werte mitsortiert und wieder mittels der Funktion `mode()` die meistvorkommende Klasse berechnet und mit der tatsächlichen des derzeitigen Bildes verglichen.

Variante 2:

Es kann auch wie bei der üblichen Klassifizierung vorgegangen werden und die meist-vorkommende Klasse im kNN Bereich bestimmt werden. Daraufhin wird überprüft, ob es sich bei dieser um ein Naturbild oder menschliches handelt.

Resultate

Da wir bei unserer Errorklassifikation alle kNN-Fehlerwerte von 1 bis zum vom Benutzer definierten Wert berechnen, kann bereits auf einen Blick (vorausgesetzt der Wert ist hoch genug) festgestellt werden, welche kNN-Werte geeignet, beziehungsweise ungeeignet sind:

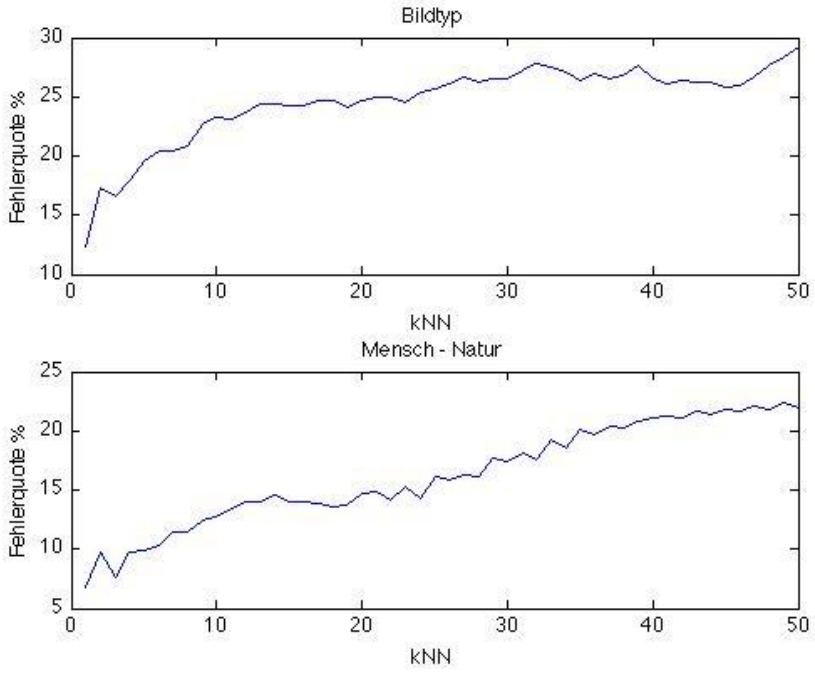


Abbildung 3: Fehlerquote in Abhängigkeit von kNN

Klassifikation des Bildtyps:

Wie aus Abbildung 3 zu entnehmen ist, ist die Fehlerquote bei der Klassifizierung des Bildtyps vor allem bei einem kNN von 1 mit 12,3% sehr gering. Bei kNN 2 und 3 ist sie schon um 5% höher, dabei liefert kNN 3 naturgemäß die besseren Ergebnisse, da durch die ungerade Anzahl an Werten eine bessere Klassifikation vorgenommen werden kann. Bei den restlichen Werten ist eine kontinuierliche Steigung der Fehlerquote wahrnehmbar. Generell hält sie sich bei einem kNN von 10 bis 45 aber in etwa im Bereich von 23% bis 28%. Da das Trainingsset 210 Bilder enthält sind maximal 209 Nachbarn möglich.

kNN 1: 12.33 %
kNN 3 : 16,62 %
kNN 21 : 24,90 %
kNN 49 : 28,42 %

Es kann also gesagt werden, dass der Klassifikator sich für einen Einsatz im realen Leben nicht sehr gut eignet, da die Fehlerquote generell mit durchschnittlich über 20% doch sehr hoch ist. Am besten eignet sich diese Klassifizierung mit einem kNN von 1 (Nearest Neighbour classifier).

Klassifikation durch Natur bzw. vom Mensch gemacht:

Betrachtet man die Ergebnise "Mensch - Natur" in Abbildung 3 (Variante 1), so sind die Ergebnisse schon wesentlich besser als die der "normalen" Klassifizierung. Wieder erhält man mit einem kNN von 1 und 3 die besten Ergebnisse. Mit steigendem kNN steigt aber auch hier die Fehlerquote stetig. Durch die geringere Anzahl an Klassen ist auch ist die Fehlerquote geringer als bei der normalen Klassifizierung.

kNN 1: 6,76 %
kNN 3 : 7,57 %
kNN 21: 14,86 %
kNN 49 : 22,43 %

In Abbildung 4 sind die Ergebnisse der oben vorgestellten "Variante 2" der Mensch-Natur Klassifizierung ersichtlich. Die Ergebnisse sind nochmal besser als die der ersten Variante.

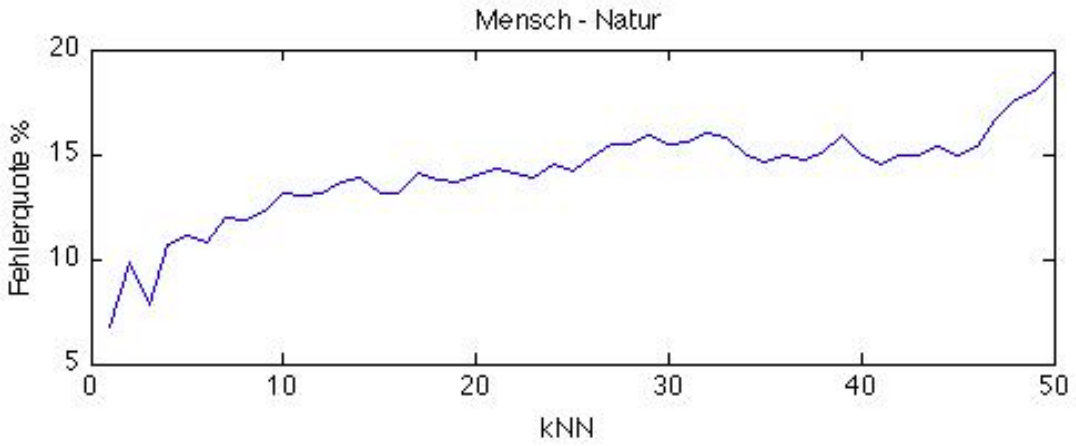


Abbildung 4: Mensch-Natur Klassifikator mit Variante 2

3. Mahalanobis Distance

Beschreibung

Ziel dieser Übung war es eine Klassifizierung anhand der Mahalanobis Distanz durchzuführen. Damit das erreicht werden kann, wird der Mittelwert aller Klassen sowie die Kovarianz benötigt. Als Trainings- und Testdaten sind dieselben Featurevektoren verwendet worden, wie für das vorhergehende Beispiel. Aus den Features wurden jene ausgewählt, die sinnvolle Werte beinhalten und dessen Kovarianzmatrix berechnung reell war. Somit wurde die Werte 1,2 und 6-12 ausgewählt. Klassifiziert werden soll wie mittels kNN, ob die Bilder natürlicher oder menschlicher Herkunft beziehungsweise ob sie einer der sieben Klassen angehören.
Es sollen verschiedene Kovarianzmatrizen berechnet werden können.

- Kovarianzmatrix einer Klasse
- diagonale Kovarianzmatrix
- mittlere Kovarianzmatrix aller Klassen
- Kovarianzmatrix über alle Klassen (wurde zusätzlich in Erwägung gezogen)

$$d(\underline{x},\underline{y}) = \sqrt{(\underline{x}-\underline{y})^T S^{-1}(\underline{x}-\underline{y})}.$$

Formel: Mahalanobis-Distanz

Mittels obiger Formel können die Matrizen berechnet werden, wichtig ist hierbei, dass der Mittelwert ein Zeilenvektor ist und für die Berechnung noch transponiert werden muss.

Mit der Variable CONTROL kann variiert werden, welche dieser Matrizen für die Distanzberechnung in Frage kommt. Danach erfolgt die Klassifizierung, wobei mittels *mode()* die am öftesten vorkommende Klasse gefunden wird. Bei einer Fehlklassifizierung wird errors an der jeweiligen Stelle auf 1 gesetzt. Bei der Klassifizierung für natürlich/menschlich werden die vorhergehenden Ergebnisse wiederverwendet

Resultate

Die aufsummierten Ergebnisse werden nur textuell angezeigt. Der auffälligste Vorteil dieser Methode ist die Performance, da der Mahalanobis Klassifikator wesentlich schneller ist als kNN. Des weiteren muss bei diesem Algorithmus nicht im vorhinein bereits ein k, das die besten Ergebnisse liefert herausgefunden werden.

In Abbildung 5 ist die Fehlerrate mit dem Mahalanobis Klassifikator für zwei beziehungsweise sieben Klassen ersichtlich. Die diagonale Kovarianzmatrix performt am schlechtesten, die anderen unterscheiden sich nicht so stark voneinander. Im Vergleich zum kNN Klassifikator ist bei der Mensch/Natur Klassifikation kein großer Unterschied zu einem niedrigen k, bei der Klassifizierung nach allen sieben Klassen ist auch auf Grund der Laufzeit ein Mahalanobis Klassifikator, mit Ausnahme der diagonalen Kovarianzmatrix zu empfehlen.

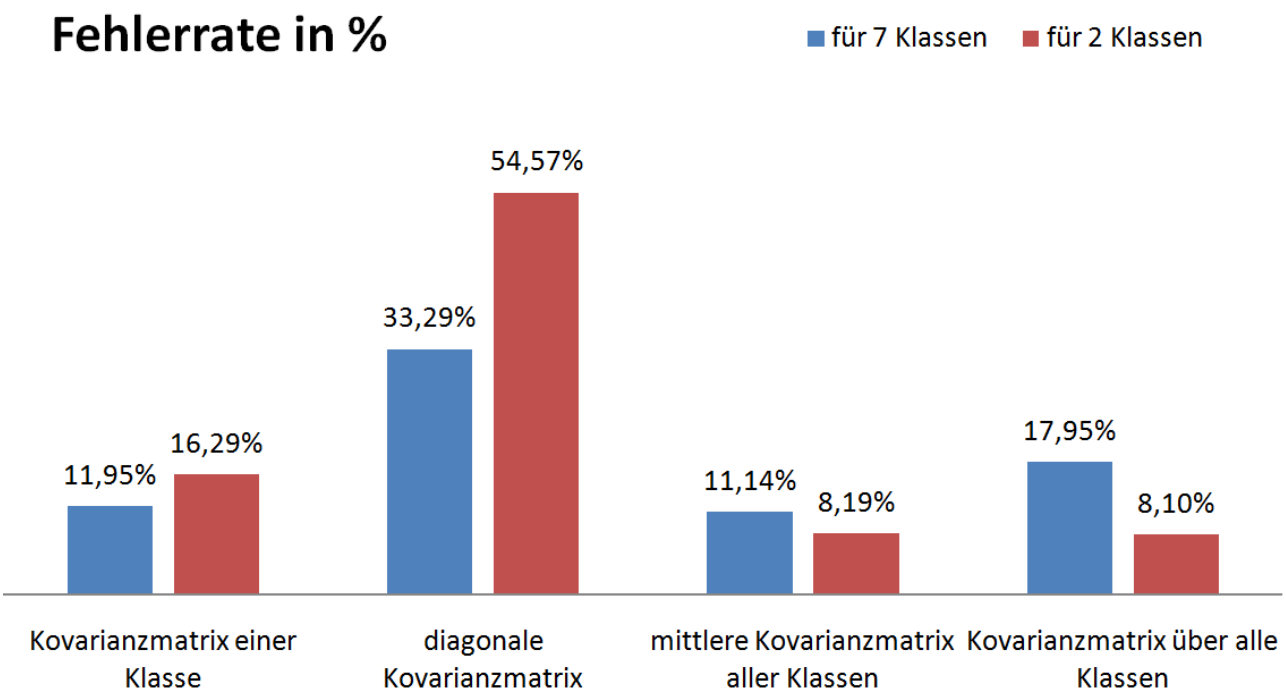


Abbildung 5: Fehlerraten Mahalanobis Klassifizierung