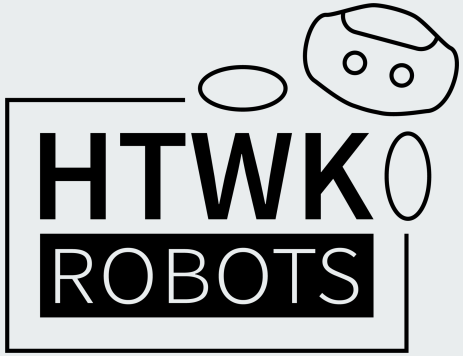
Pfifferkennung für NAO Robots



Nico Schramm (23INM)
Johann Straube (23INM)

GitHub Repository

C201.2: Mustererkennung
Prof. Grüttmüller

Fakultät Informatik & Medien
HTWK Leipzig

Gliederung

- Problemstellung
- Datenbasis
- Extraktion von Merkmalen
- Ansätze für Datenverarbeitung
- Fazit

Problemstellung

NAO Roboter

- Hersteller Aldebaran
- 57,4 cm hoch
- 5,4 kg schwer
- Zweck: Vermittlung von Robotik

© Aldebaran (Quelle), Zugriff am 24.06.2024

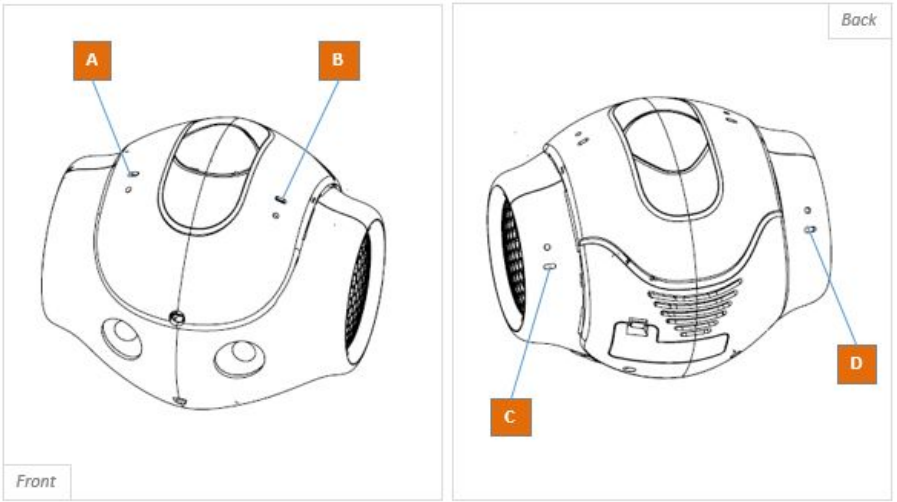
Standard Platform League (SPL)

- 7 gegen 7 Roboterfußball
- Teams verschiedener Universitäten
- Internationale Wettkämpfe
- Standardisierte Roboter für alle Teams
- Neue Regel: Roboter soll auf **Pfiff** des Schiedsrichters **reagieren**

HTWK Robots (Quelle), Zugriff am 22.06.2024

Die Ohren des NAO

- 4 Mikrophone
- Abtastrate: 44,1 kHz
- Frequenzbereich: 150 Hz – 12 kHz



Xiaofei Li et al. (CC-SA 4.0) (Quelle), Zugriff am 24.06.2024

Datenbasis

Audio-Dateien

- Aufnahmen aus unterschiedlichen Spielen
- Audio-Dateien im `flac` Format
- kompressionsfreies Format
  → keine Verluste

© Mike Wren (Quelle), Zugriff am 24.06.2024

Label-Dateien

- CSV Datei zu jeder Audio Datei
- Zeitstempel mit Label
- Vorhandene Labels:
  - `No_Whistle`
  - `Whistle`
  - `False_Whistle`

start	**end**	**label**
62.35092589842589	68.25101351351351	No_Whistle
68.97197278911564	69.7730612244898	Whistle
70.05104024354024	96.25142931392931	False_ Whistle

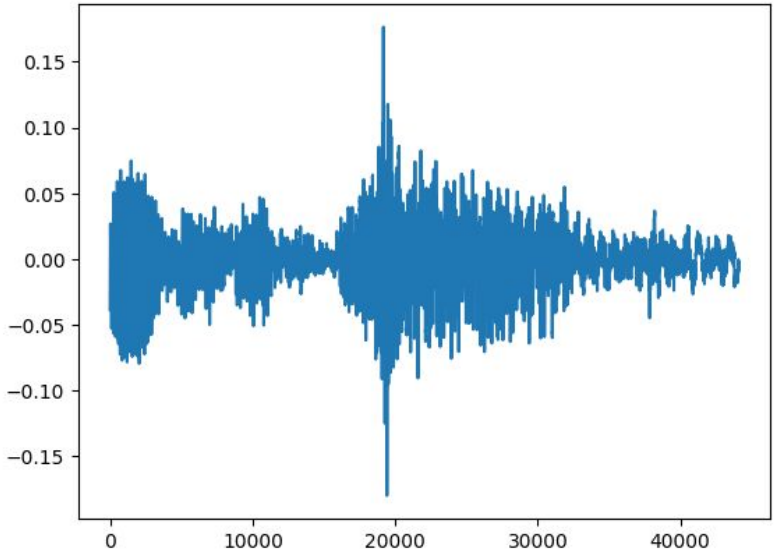
Einlesen der Dateien

Audio-Daten

- Einlesen mit `librosa.load` Funktion
- Generierung einer Waveform

Label-Daten

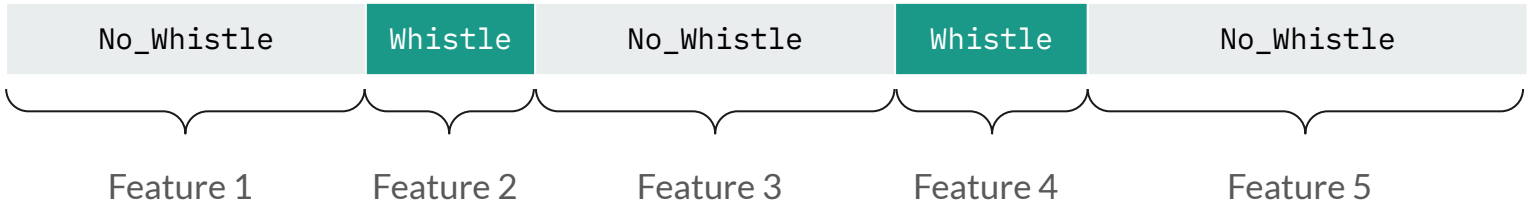
- Einlesen mit `pandas`
- Label auf Klassen abbilden
  - `Whistle` → `1`
  - `No_Whistle` → `-1`
  - `False_Whistle` → `1`



Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

Naiver Ansatz

- Schneiden der Audio Dateien entlang der Labelgrenzen
- inhomogene Dimensionen → Padding mit 0



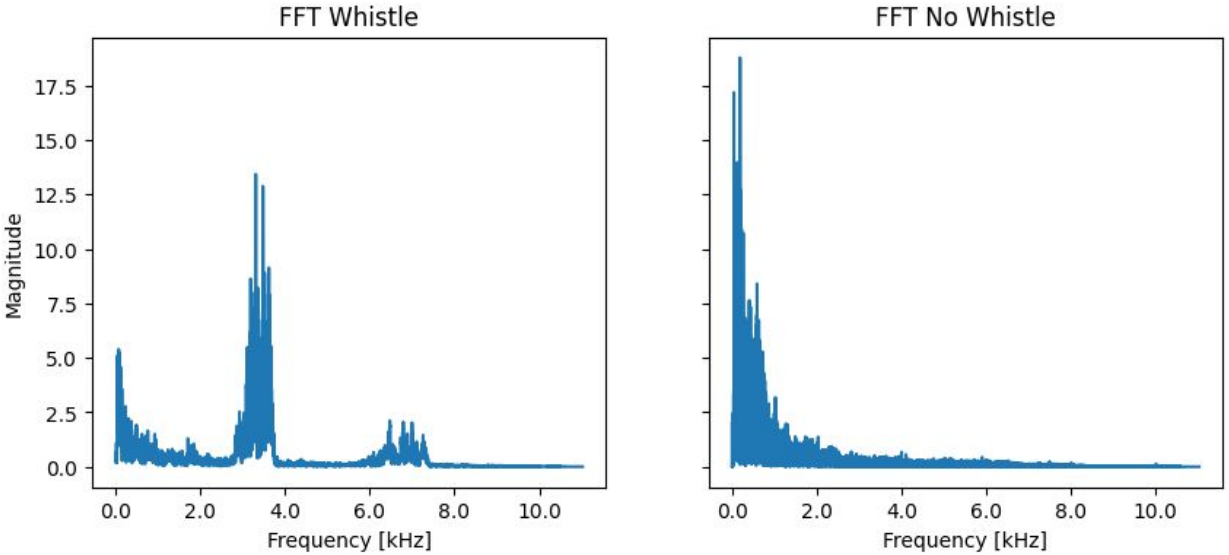
Extraktion von Merkmalen

1. Option: Fast Fourier Transformation (FFT)

- Schneller Algorithmus zur Berechnung der Fourier Transformation
- Zerlegung der Waveform in Frequenzanteile

Lucas Vieira (Quelle), Zugriff am 24.06.2024

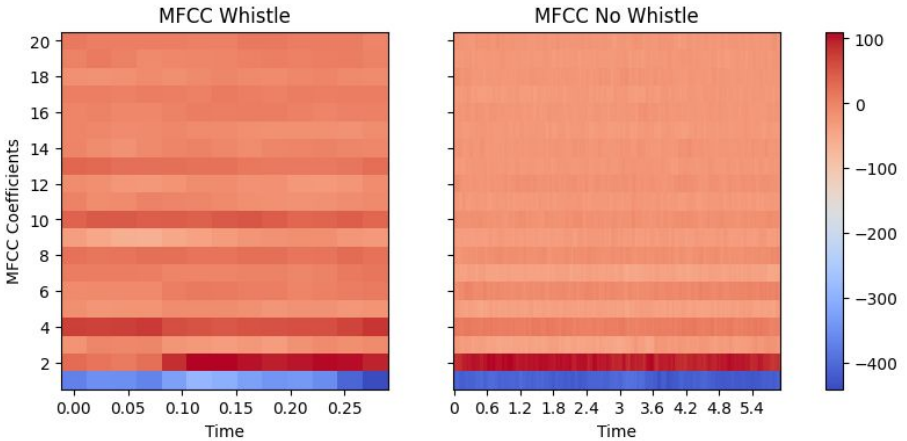
1. Option: Fast Fourier Transformation (FFT)



Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

2. Mel-frequency Cepstrum Coefficients (MFCC)

- Reduziert Audiosignale ohne Eigenschaften zu verlieren
- Entwickelt für Spracherkennung
- Extrahiert Oberschwingungen (*Harmonics*) und Seitenbänder (*Sidebands*) des Signalspektrums

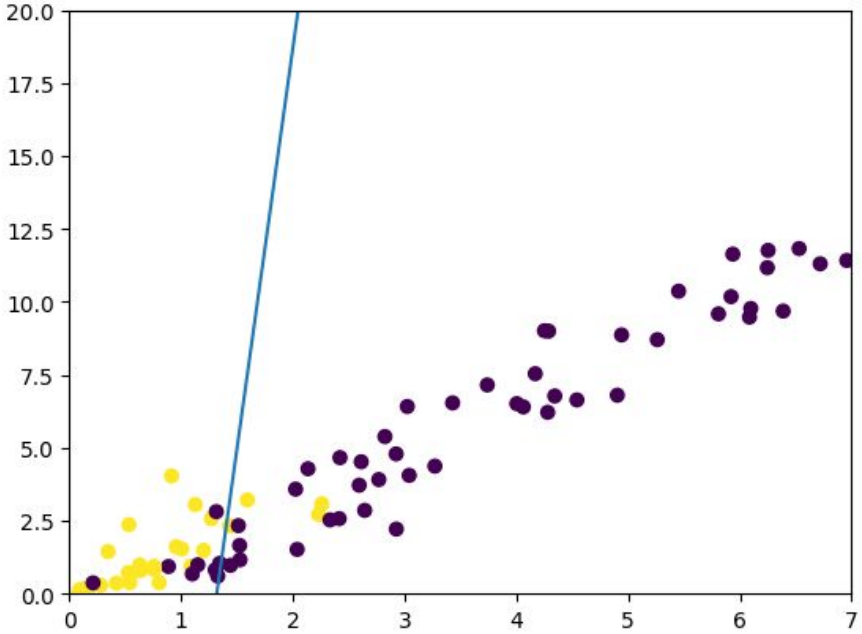


Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

Erster (naiver) Ansatz

Trennen durch Gradientenabstieg

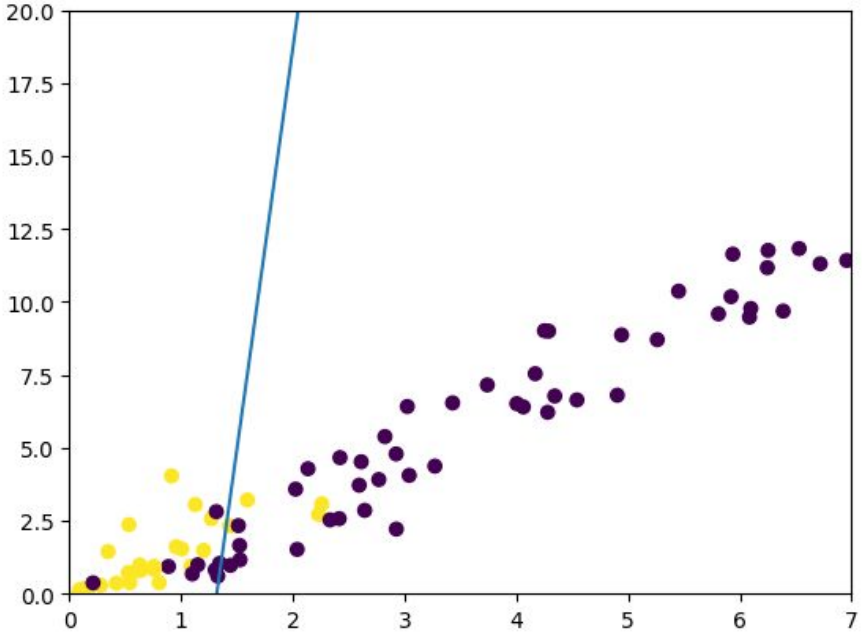
- Erstellung FFT
- Extraktion der Frequenzbereiche
  - X-Achse 2 – 2,5 kHz
  - Y-Achse 2,5 – 3 kHz
- Ausführung des Gradientenabstiegs mit eigener Implementierung



Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

Gradientenabstieg Ergebnis

- Lineare Trennung nicht möglich
- Dennoch annehmbare Werte:
  - `accuracy:  90,48%`
  - `precision: 77.14%`
  - `recall:    87.10%`
  - `f1:        0.8182`
- Gradientenabstieg mit mehr Dimensionen bringt nur wenig Verbesserungen



Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

Arbeit mit `scikit-learn`

- Features: MFCC oder FFT
- Trennen der Daten in Trainings- und Testdaten
- Training des Modells mit Trainingsdaten
- Auswertung erstellen mit Testdaten

Modelle mit `scikit-learn`

- Perzeptron
- Stochastisches Gradientenverfahren
  (*Stochastic Gradient Descent*, SGD)
- Support Vektor Maschine
  (SVM, *Support Vector Classifier*)
- später: Multi-layered Perzeptron (MLP)

Ergebnisse

Modell	Präzision	Recall	F1-Score
Perzeptron (MFCC)	93,26%	100%	0,97
SGD (MFCC)	93,26%	100%	0,97
SVM (MFCC)	95,40%	100%	0,98
SVM (10 Dateien FFT)	70,00%	100%	0,82

Schnipsel-Ansatz 1

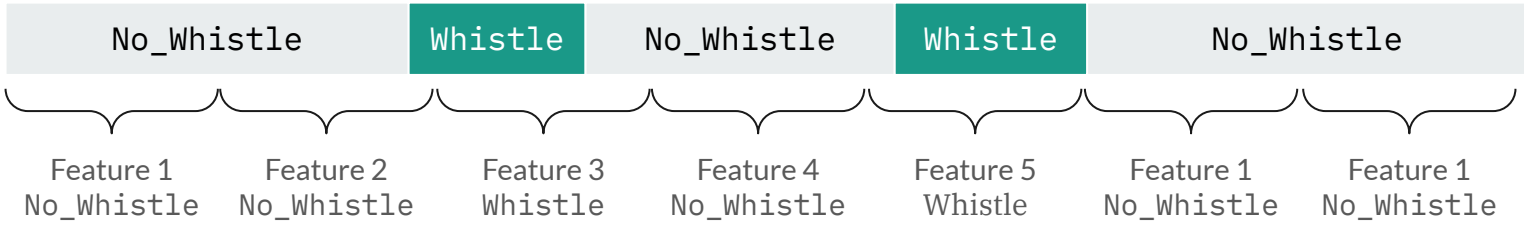
Audio-Dateien in gleiche Abschnitte schneiden

- Befürchtung: Modelle lernen **Länge** der Abschnitte
- Daher: Schneiden der Audio-Dateien **1-sekündige** Schnipsel
- Runden?

$$(\text{start},\ \text{end}) = (5.1,\ 5.4) \approx (5, 5)$$
$$(\text{start}, \text{end}) = (5.5,\ 5.9) \approx (6,\ 6)$$

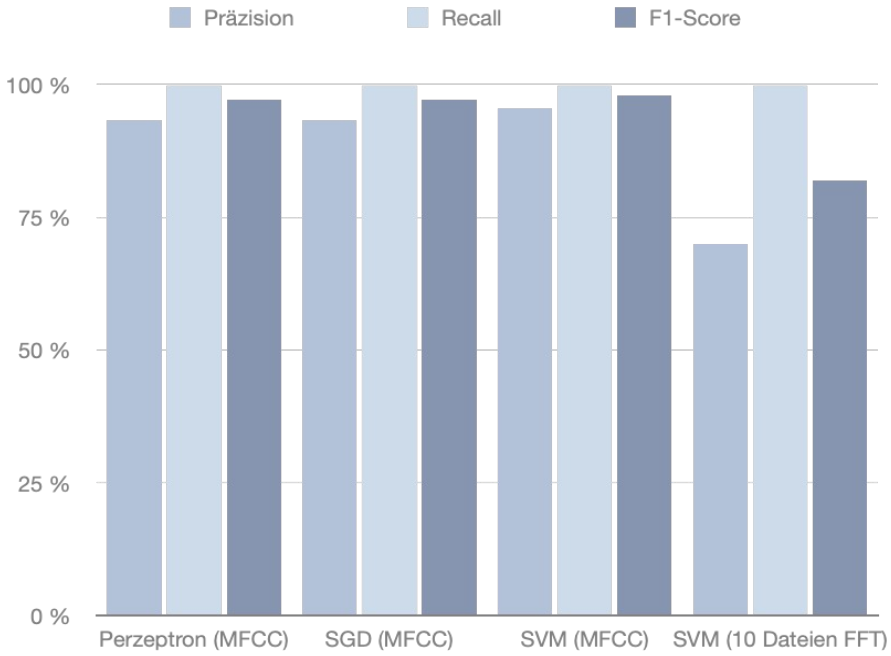
- `Whistle` Label wird vergeben, wenn “genug” Pfiff enthalten ist

Audio-Dateien in gleiche Abschnitte schneiden

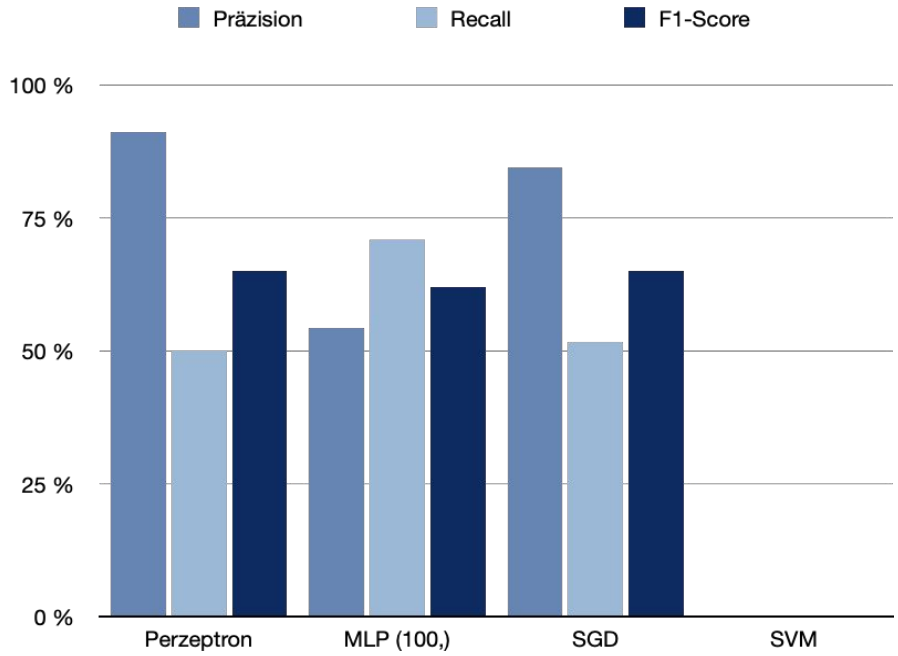


$$\mathbf{start}(t) = \begin{cases} \lceil t \rceil, & (t \bmod 1) > 1 - \theta \\ \max(0, \lfloor t \rfloor), & \text{sonst} \end{cases}$$

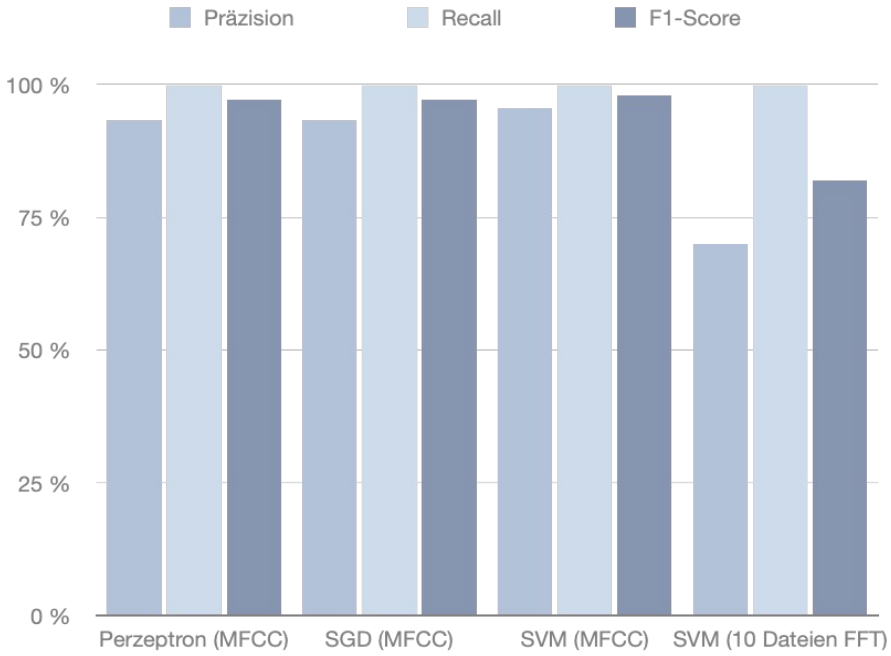
$$\mathbf{end}(t) = \begin{cases} \max(0, \lfloor t \rfloor), & (t \bmod 1) < \theta \\ \lceil t \rceil, & \text{sonst} \end{cases}$$



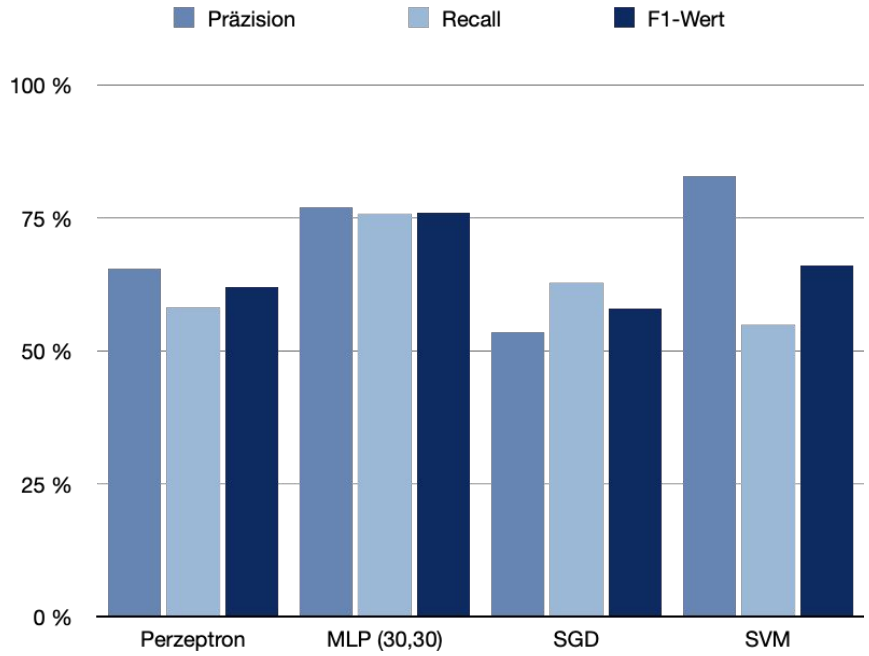
ungeschnitten



Schnipsel-Ansatz 1 MFCC



ungeschnitten



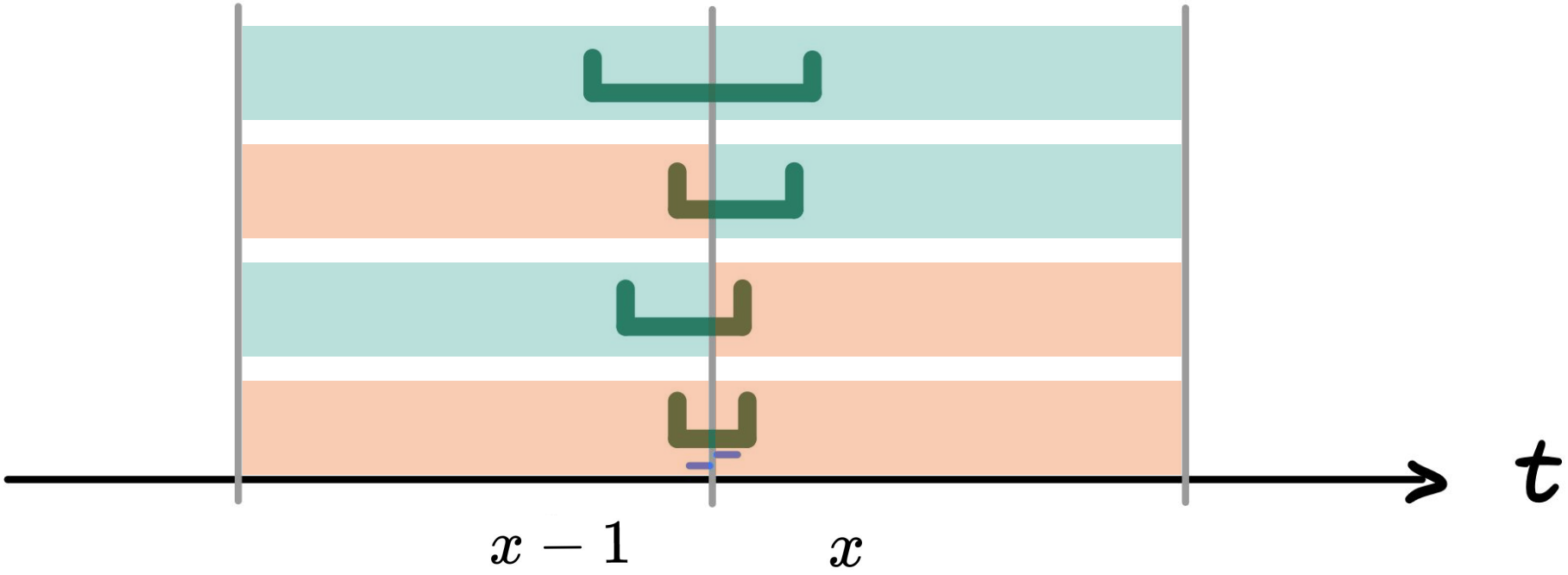
Schnipsel-Ansatz 1 FFT

Schnipsel-Ansatz 2

Audio-Dateien schneiden und Grenzfälle entfernen

Grenzfälle **nicht** übernehmen, da …

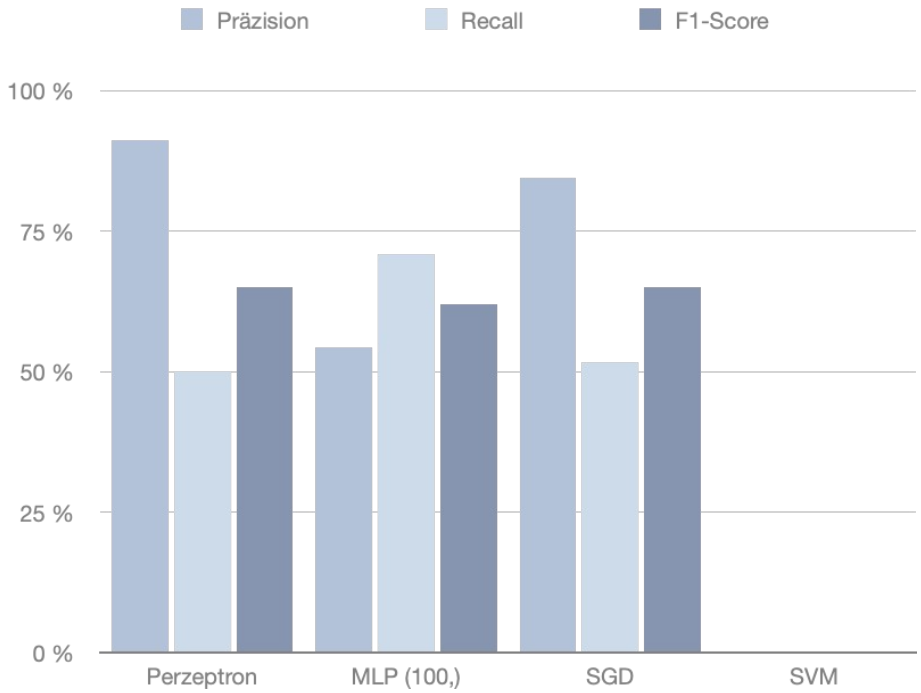
- Ergebnis der Klassifizierung in diesem Fall egal
  (davor / danach ist immer ein Abschnitt mit mehr Pfiff)
- Verwirrung des Modells (?)



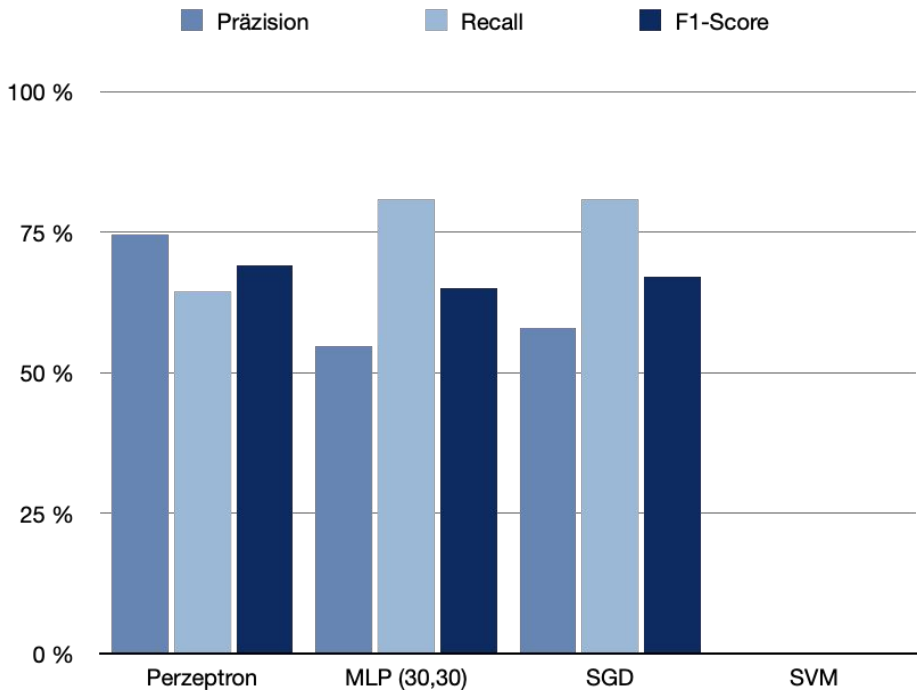
Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

Funktionsweise new_cut Ansatz

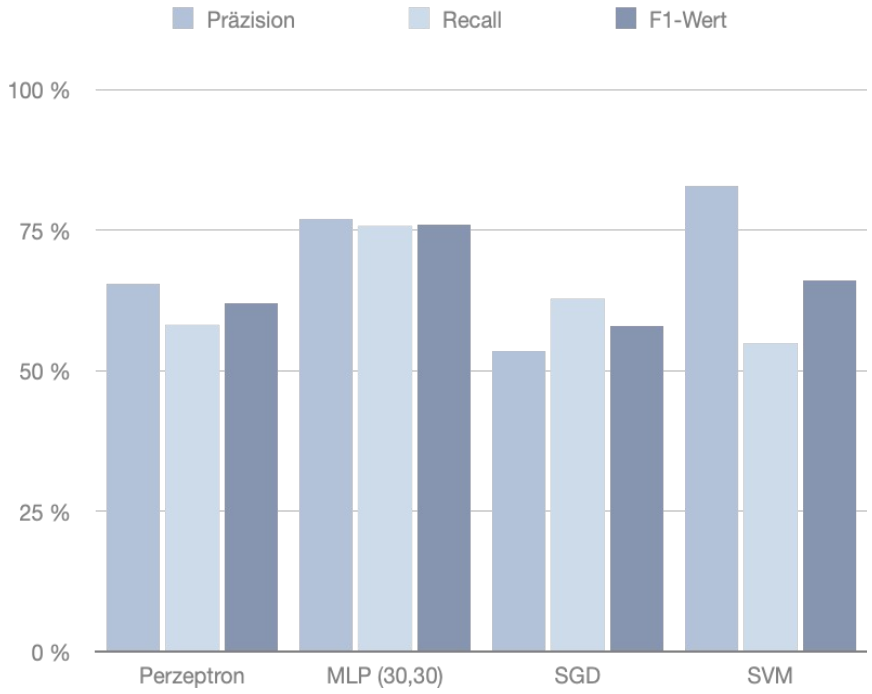
```
Whistle from 504.216 to 504.668
added to 504
added to 504

Whistle from 581.950 to 582.118
removed 581
added to 582
```



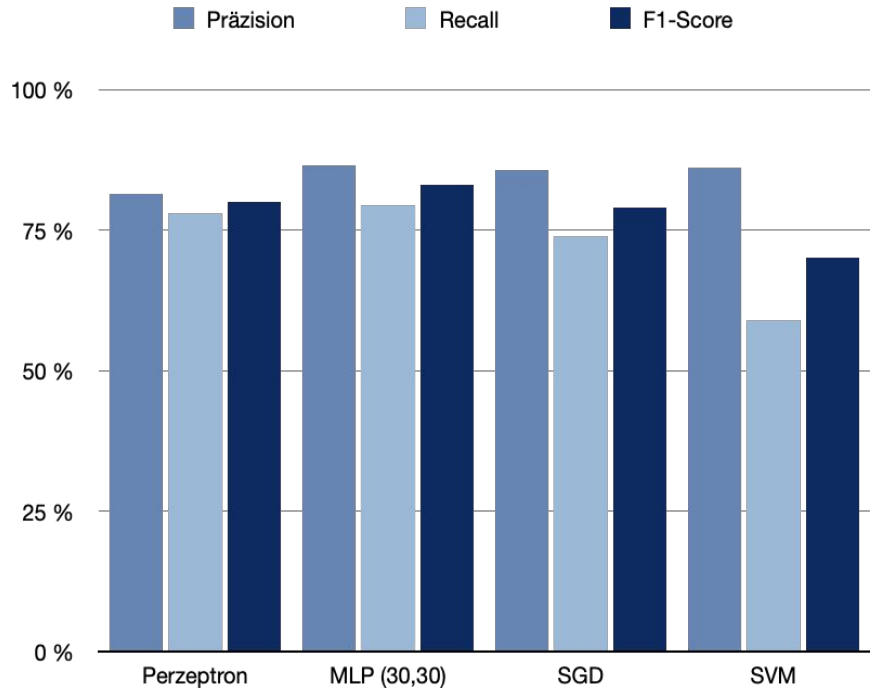
Schnipsel-Ansatz 1 MFCC



Schnipsel-Ansatz 2 MFCC



Schnipsel-Ansatz 1 FFT



Schnipsel-Ansatz 2 FFT

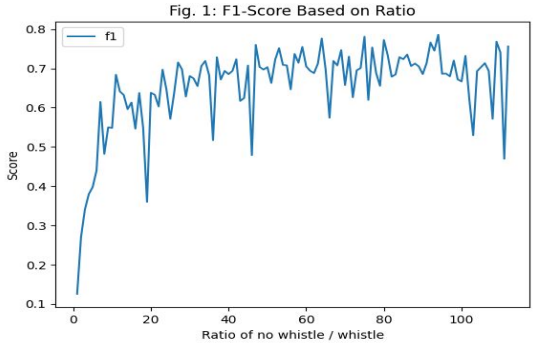
Verhältnis-Bias

Mengenverhältnisse der Labels

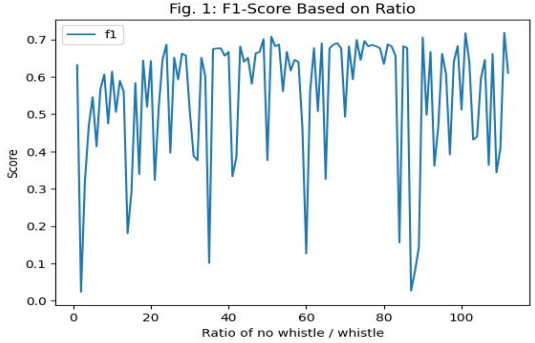
- Anzahl der `Whistle` und `No_Whistle` Labels stark unterschiedlich
- Befürchtung: Modelle bekommen starken **Bias**
  ⇒ Modelle mit **unterschiedlichen Mengenverhältnissen** testen
- Anschließend Wahl des Modelles mit den besten Ergebnissen
- Anmerkung: ausschließlich mit MFCC, da FFT vergleichsweise sehr lange rechnet

Modelle mit `scikit-learn`

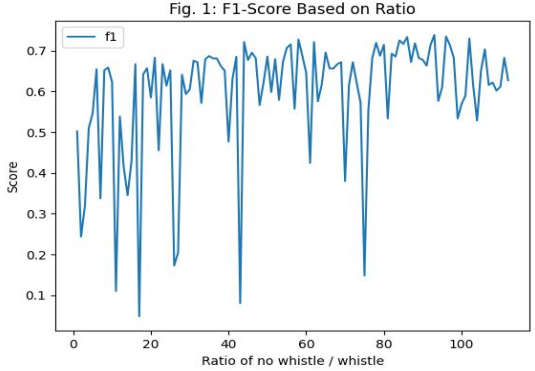
Modell (Feature: MFCC)	Verhältnis	Präzision	Recall	F1-Score
Perzeptron	87:1	100%	1,37%	0,03
MLP (30,30)	108:1	93,75%	41,10%	0,57
SGD	81:1	87,50%	38,36%	0,53
SVM	56:1	100%	2,74%	0,05



MLP

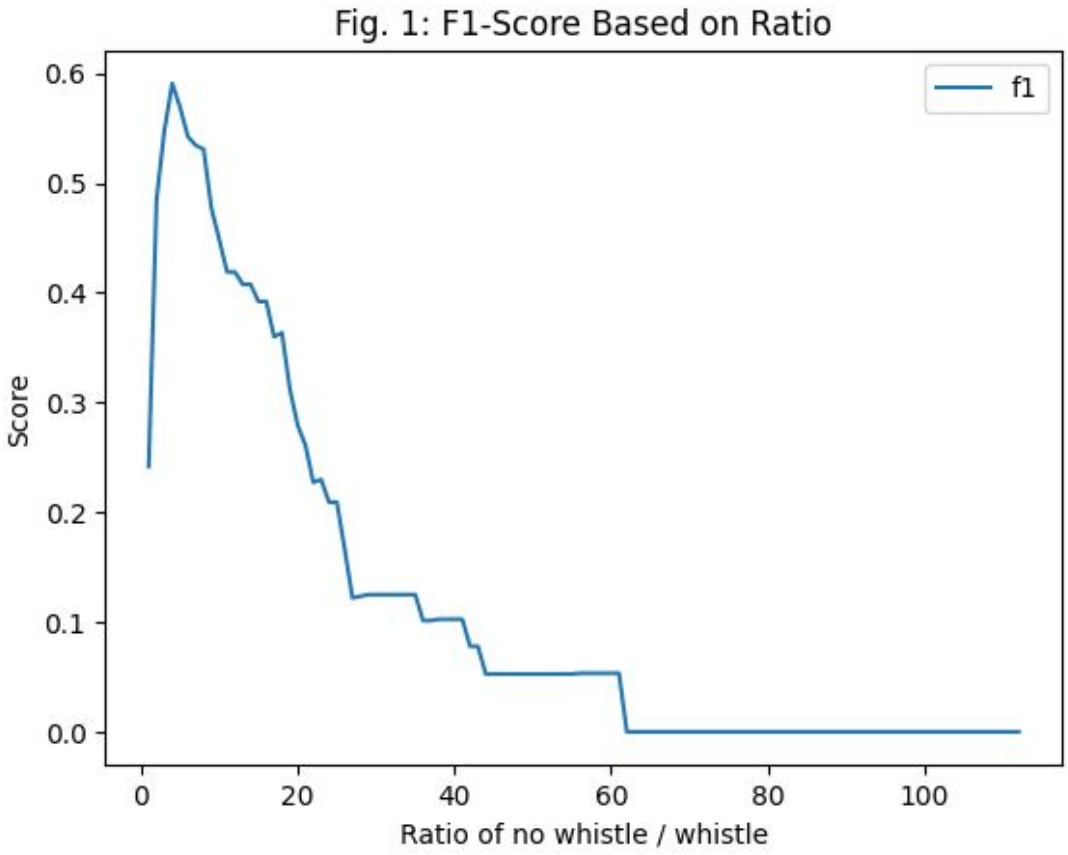


Perzeptron



SGD

Support Vektor Maschine

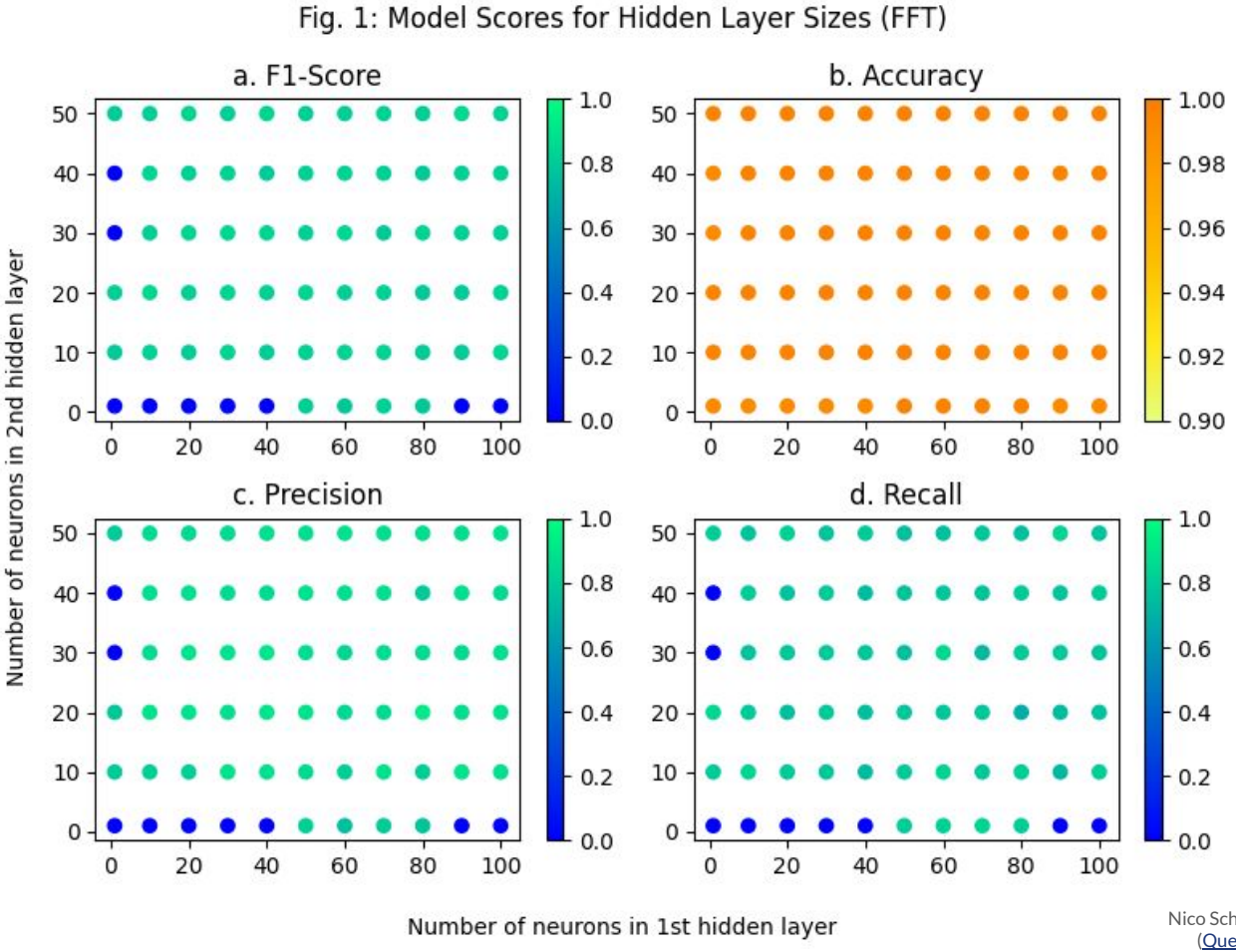


Hyperparameter-Optimierung Multi-layered Perzeptron

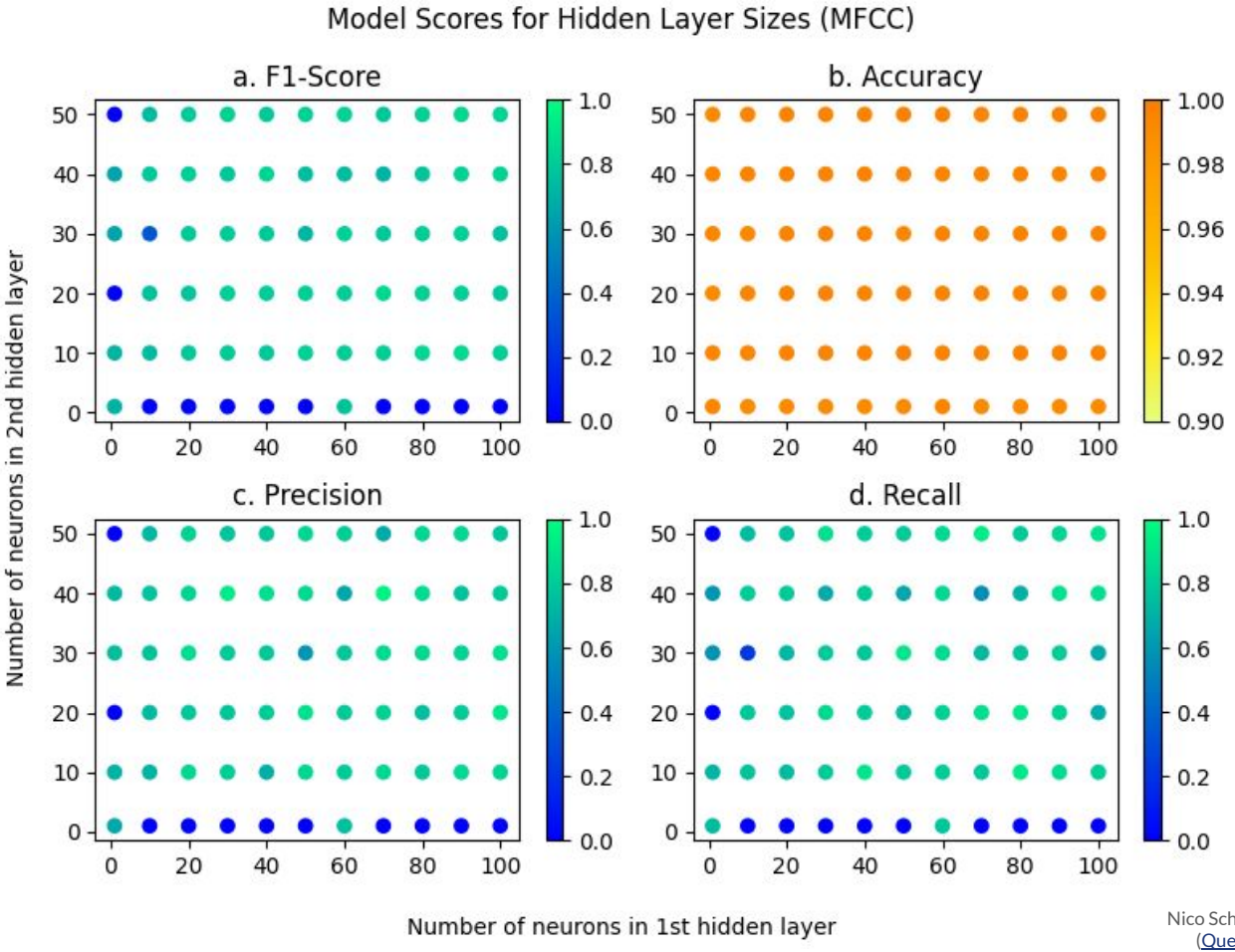
Hyperparameter-Optimierung des MLP

- Testen des MPL mit unterschiedlichen **Hidden-Layer** Größen

  {(1,1), (1,10), (1,20), ..., (1,50), (10,1), (10,10), ..., (10,50), ..., (100,50)}



Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

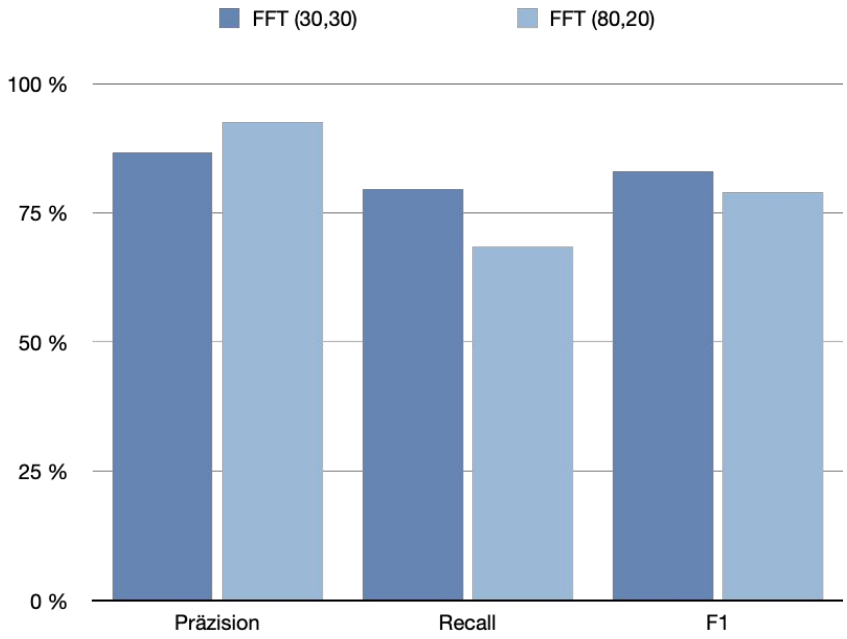


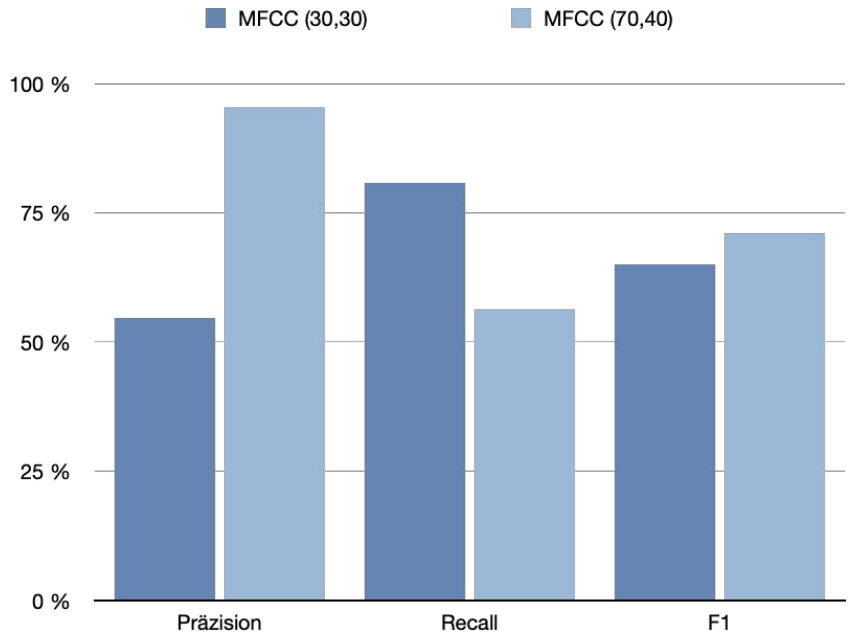
Nico Schramm & Johann Straube (MIT) (Quelle), Zugriff am 24.06.2024

Ergebnis der Hyperparameter-Optimierung des MLP

Feature	Größe	Präzision	Recall	F1-Score
FFT	(80,20)	92,59%	68,49%	0,79
MFCC	(70,40)	95,35%	56,16%	0,71

Ergebnis der Hyperparameter-Optimierung des MLP





Fazit

Fazit

- Schwieriger zu trennen als anfangs vermutet
- Einheitliche Länge der Audio-Daten wichtig
- Anzahl der mit "Pfiff" und "kein Pfiff" markierten Trainingsdaten sollte bei der Verwendung der SVM nicht zu unterschiedlich sein
- Größe der Hidden Layer im Multi-layered Perzeptron hat großen Einfluss auf die Leistung
- Leistungen des MFCC- im Vergleich zum FFT-Feature unterscheiden sich teilweise stark
- Bestes Ergebnis: MLP auf FFT-Basis im zweiten Schnipsel-Ansatz (P: 86,57%, R: 79,45%, F1: 0,83)

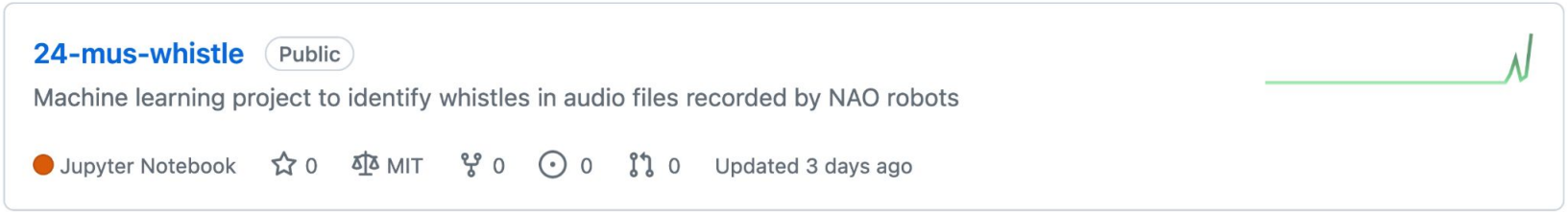
Ausblick

- Implementierung auf Roboter in C
- Performance-Analyse
- Unterscheidung zwischen `Whistle` und `False_Whistle`
- Weitere Hyperparameter-Optimierung
- Skalierung der Daten als mögliche Verbesserung (v.a. für Gradienten-Abstieg)
- Untersuchung weiterer Modelle

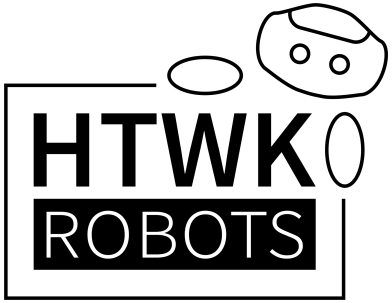
Repository & Dokumentation

Repository und weitere Dokumentation einsehbar unter

https://github.com/24-mus-whistle/24-mus-whistle



Pfifferkennung für NAO Robots



- Schwieriger zu trennen als anfangs vermutet
- Einheitliche Länge der Audio-Daten wichtig
- Anzahl der mit "Pfiff" und "kein Pfiff" markierten Trainingsdaten sollte bei der Verwendung der SVM nicht zu unterschiedlich sein
- Größe der Hidden Layer im Multi-layered Perzeptron hat einen Einfluss auf die Leistung
- Leistungen des MFCC- im Vergleich zum FFT-Feature unterscheiden sich teilweise stark
- Bestes Ergebnis: MLP auf FFT-Basis im zweiten Schnipsel-Ansatz (P: 86,57%, R: 79,45%, F1: 0,83)