



**WIR BEGEISTERN UNSERE KUNDEN MIT  
EMOTIONALEN FAHRERLEBNISSEN.**

PRÄZISE - CHARAKTERSTARK - INNOVATIV

# MASCHINELLES LERNEN AM BEISPIEL EINER ÜBERSTEUER-ERKENNUNG.

**SUPERVISED MACHINE LEARNING MIT MATLAB.**

**BMW  
GROUP**



**Rolls-Royce**  
Motor Cars Limited

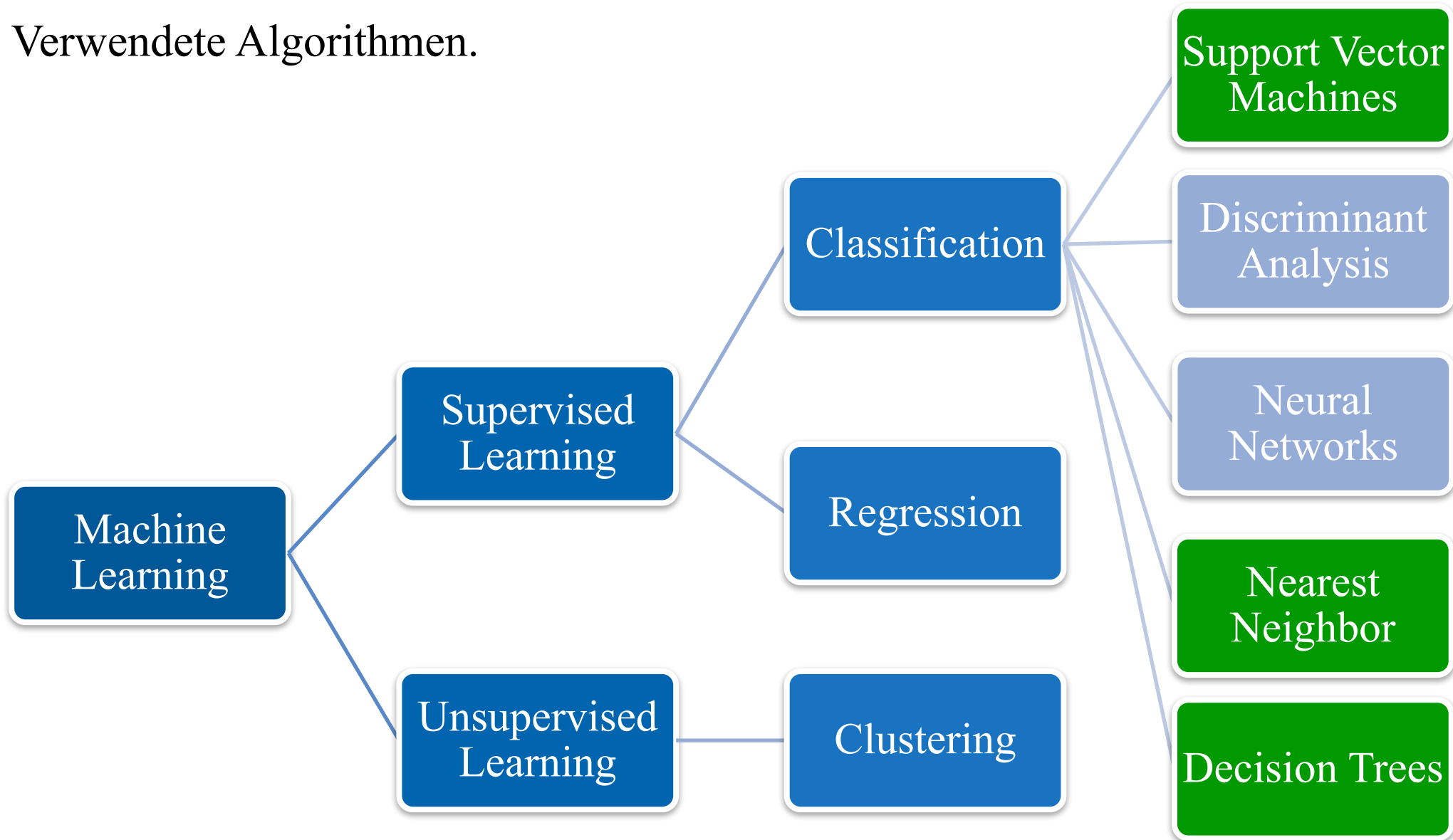
# MOTIVATION.

Evaluierung der „Statistics and Machine Learning“ Toolbox von MATLAB

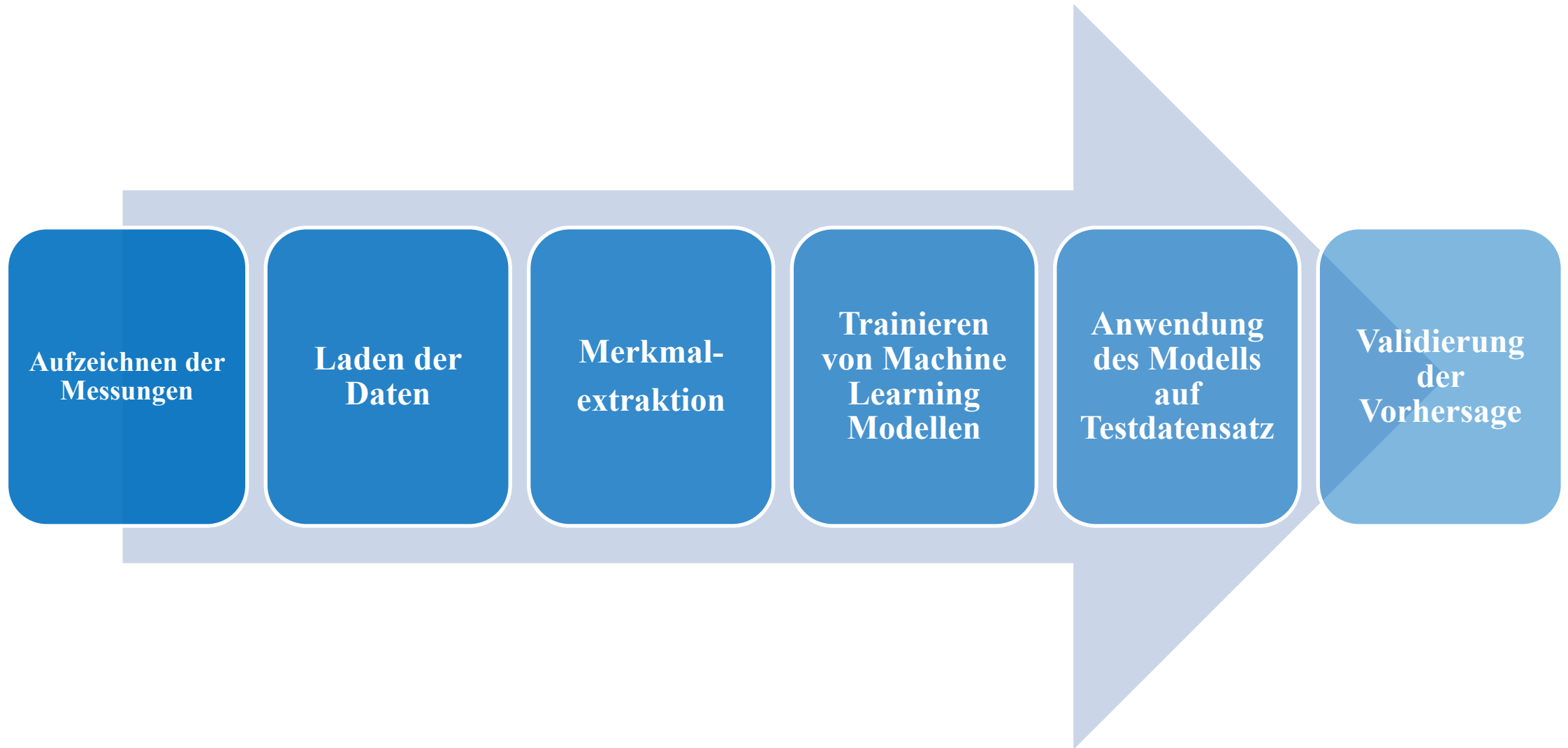
Große Anzahl an aufgezeichneten Fahrzeugmessungen (unlabeled) verfügbar

# MACHINE LEARNING.

## Verwendete Algorithmen.

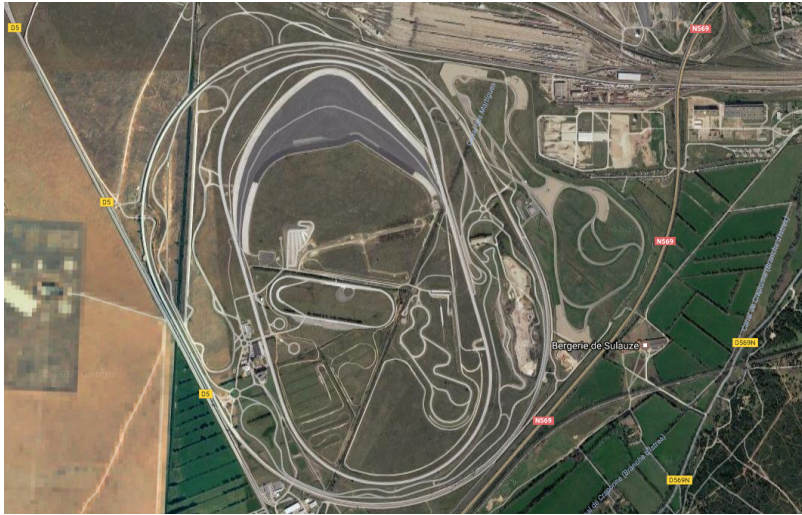


# ABLAUF.



# GENERIERUNG DER MESSDATEN. STRECKE

## Trainieren eines Modells



### Trainingsdatenset:

- Handlingskurs Miramas
- 259.000 Datenpunkte
- ≅ 43 Minuten

## Testen des erstellten Modells



### Testdatenset:

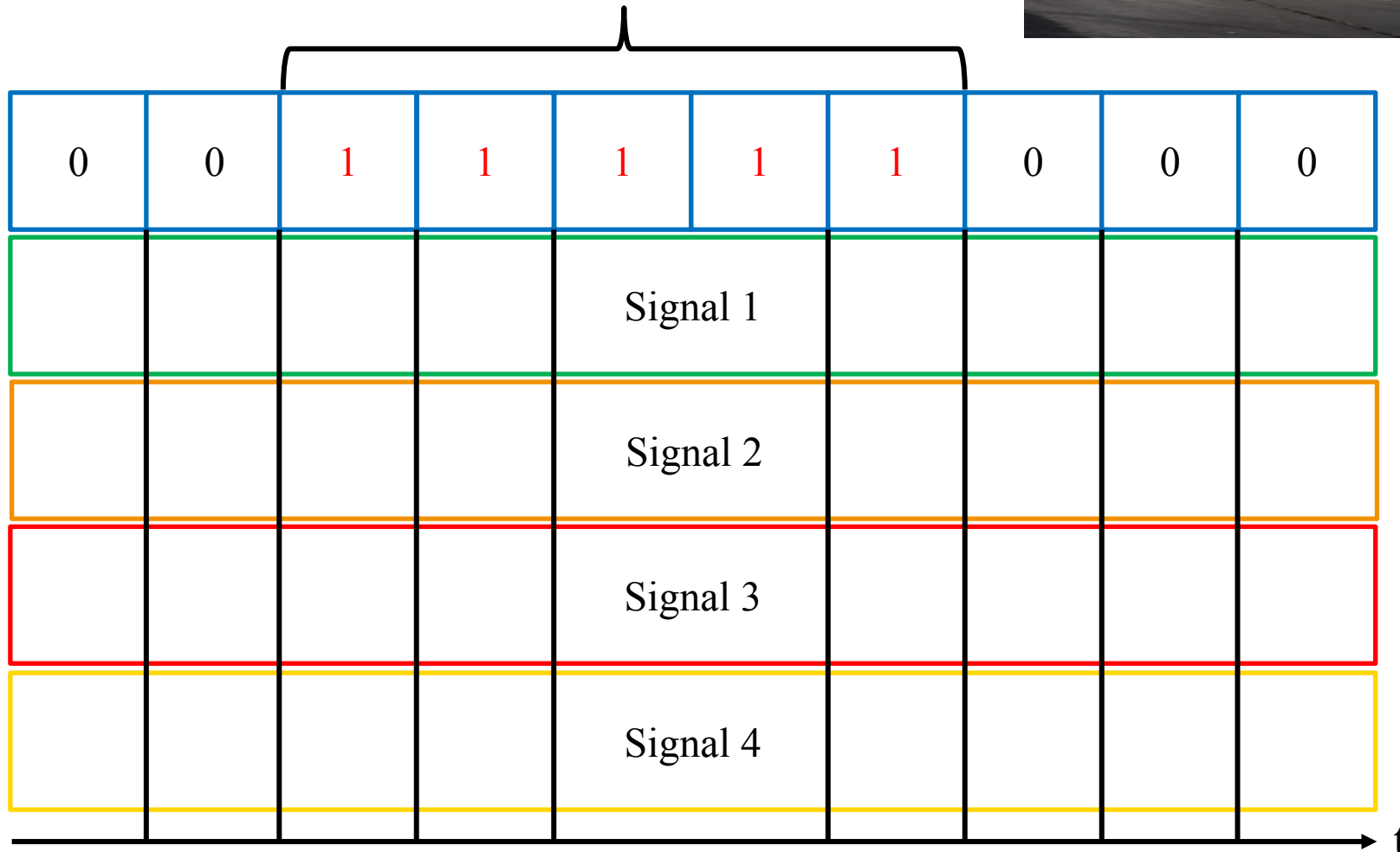
- Handlingskurs Aschheim
- 150.000 Datenpunkte
- ≅ 25 Minuten



# GENERIERUNG DER MESSDATEN.

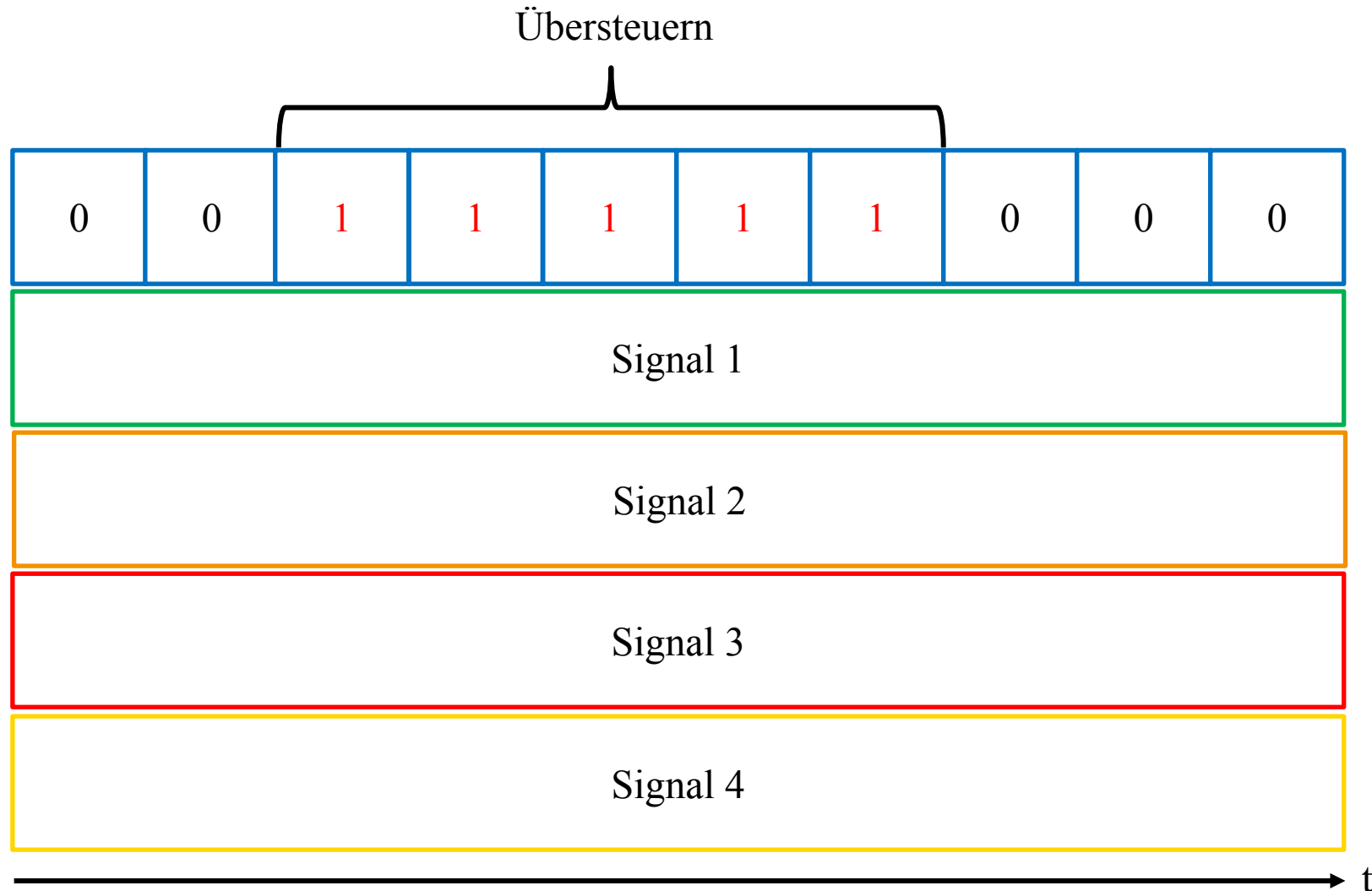
Einfügen eines Trigger-Signals.

Übersteuern



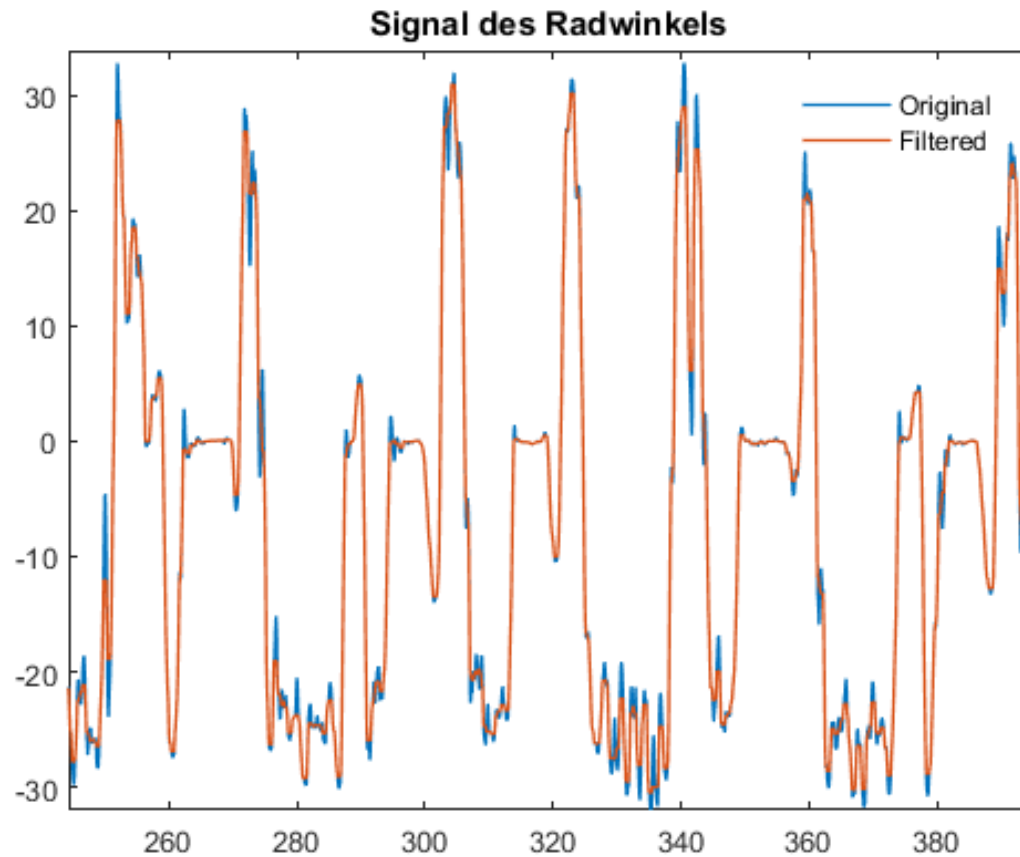
# GENERIERUNG DER MESSDATEN.

Einfügen eines Trigger-Signals.



# MERKMALEXTRAKTION.

Filter.



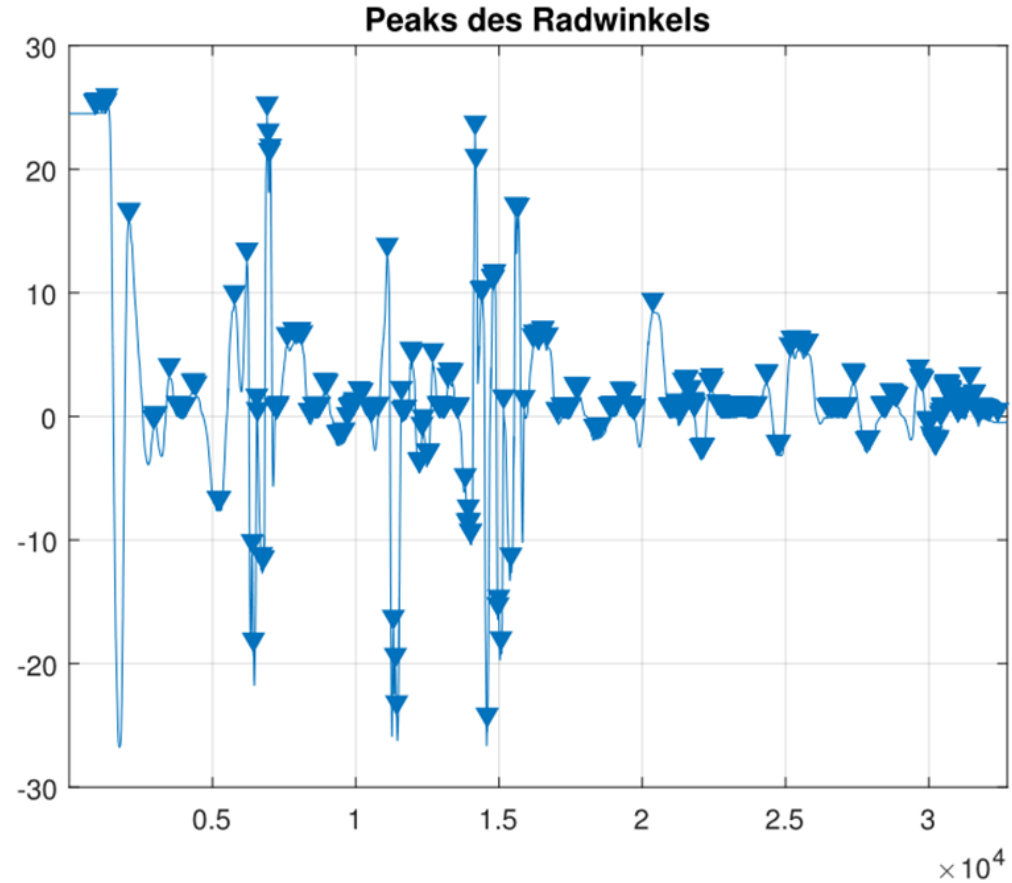
Rauschen des Signals unterdrücken

Trainings- und Testdatenset müssen  
mit dem gleichen Filter gefiltert werden



# MERKMALEXTRAKTION.

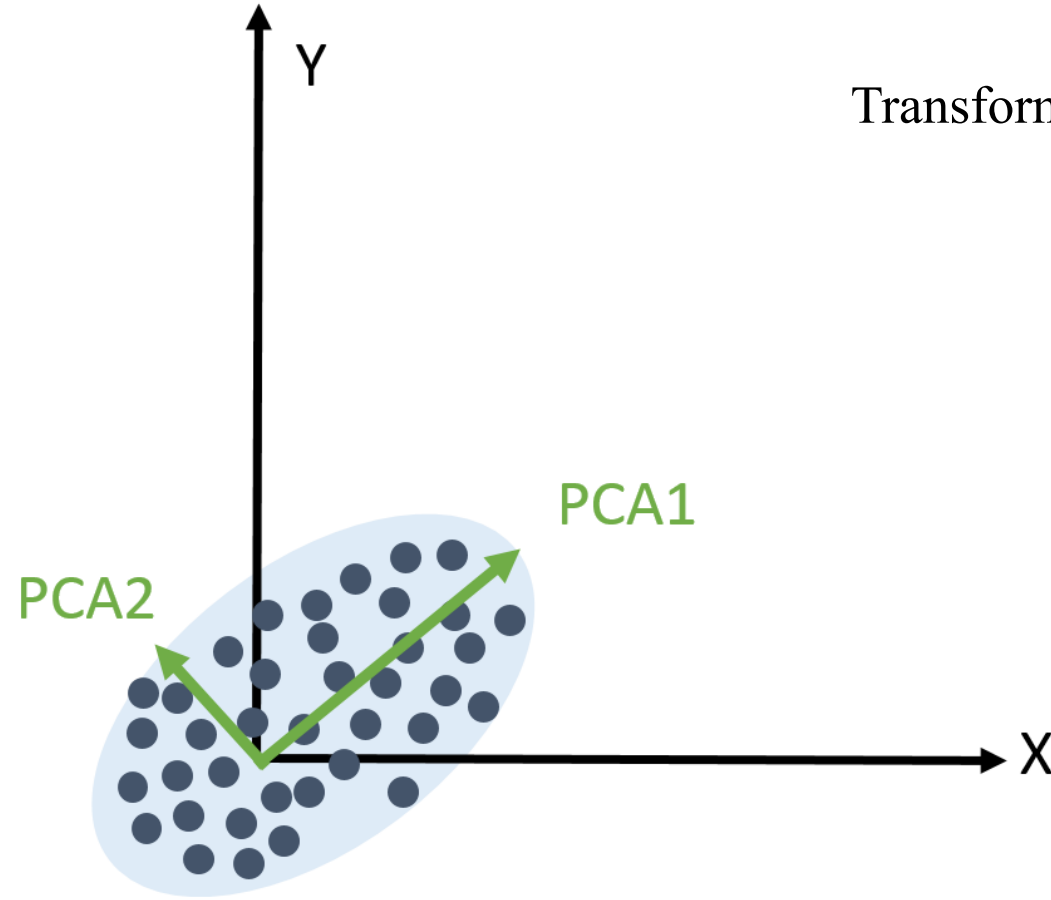
## Peak Analysis.



- Anwendung der Funktion FindPeaks
- Mindestdistanz zwischen den Peaks

# MERKMALEXTRAKTION.

## Principal Component Analysis (PCA).



Transformation in der Richtungen der Principal Components

# MODELLAUSWAHL.

## K-Fold Crossvalidierung.

|            | fold 1                                                                             | fold 2                                                                             | fold 3                                                                               | fold 4                                                                               | fold 5                                                                               |
|------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| training 1 |   |   |   |   |   |
| training 2 |   |   |   |   |   |
| training 3 |   |   |   |   |   |
| training 4 |   |   |   |   |   |
| training 5 |  |  |  |  |  |

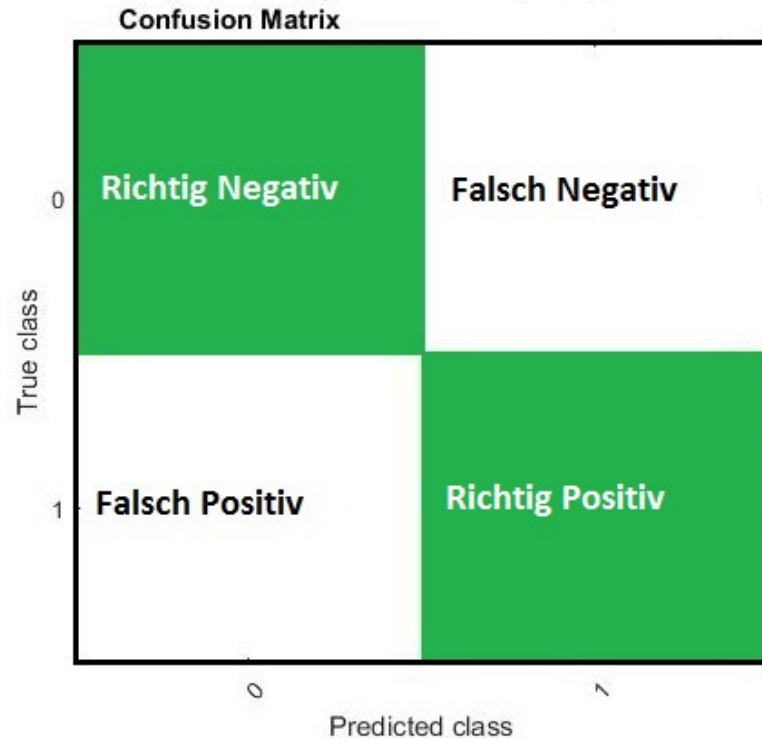
Blau = Trainingsdatenset  
Rot = Testdatenset

Ergebnis:  
Durchschnittlicher Fehler

Quelle: Machine Learning for Evolution Strategies, Kramer, 2016, S.39

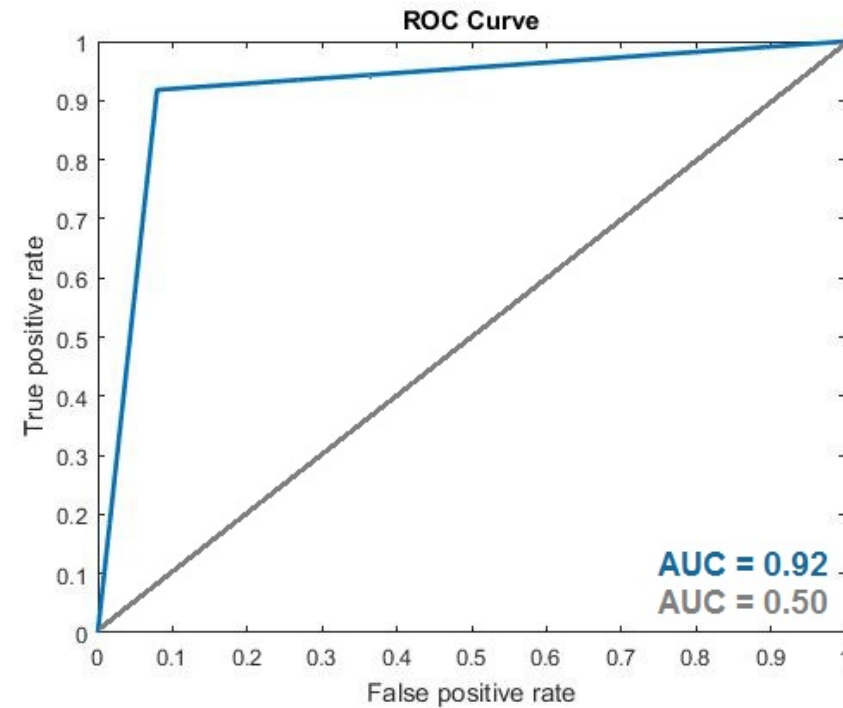
# MODELLAUSWAHL.

## Konfusionsmatrix.



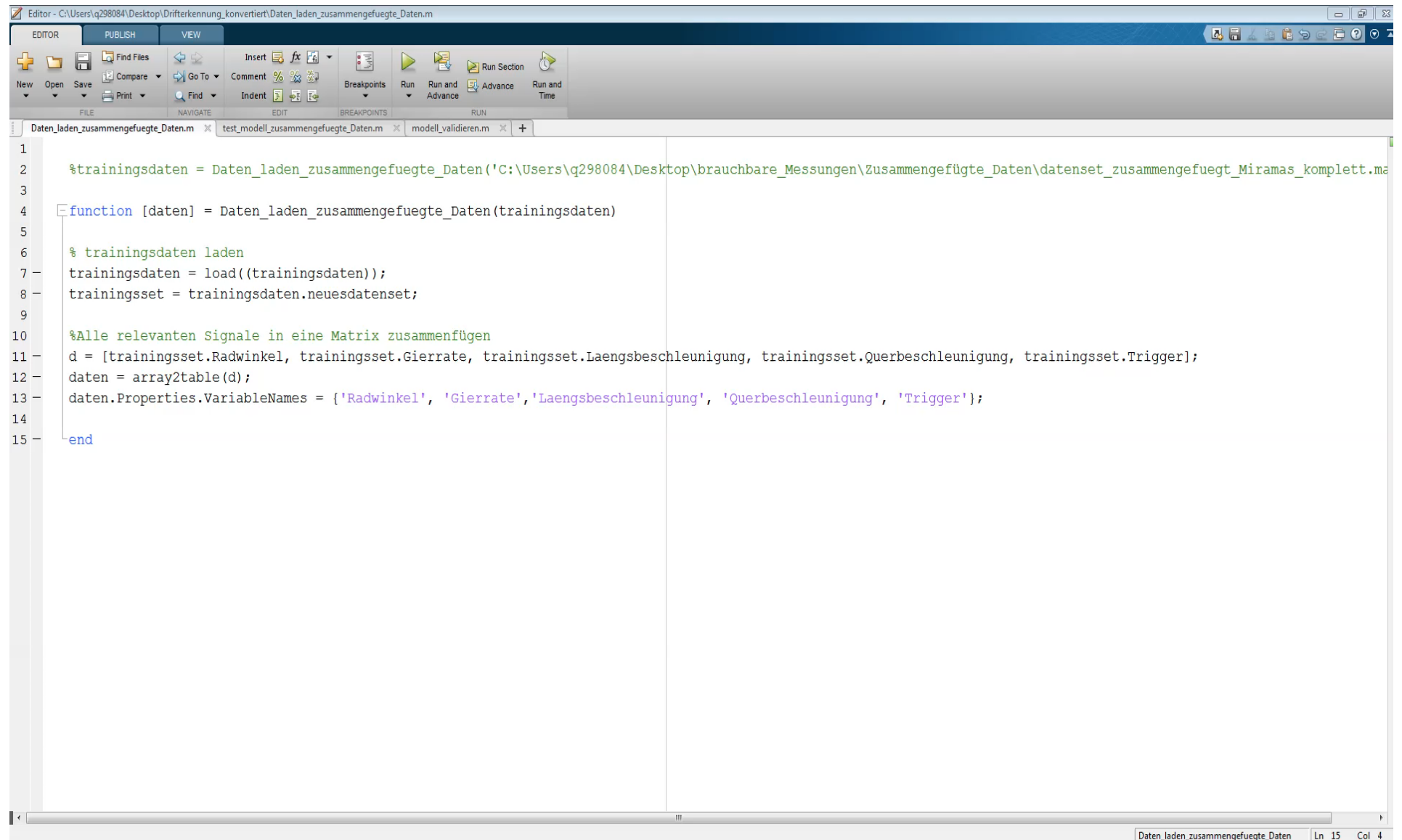
Ziel:  
100 % auf der grünen Diagonale

## Receiver-Operating-Characteristic-Kurve.



Ziel:  
AUC = 1

# VIDEO: VORGEHENSWEISE.



The image shows a MATLAB Editor window with the following content:

```
Editor - C:\Users\q298084\Desktop\Drifterkennung_konvertiert\Daten_laden_zusammengefuegte_Daten.m

EDITOR PUBLISH VIEW
New Open Save Find Files Go To Comment % Indent Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN

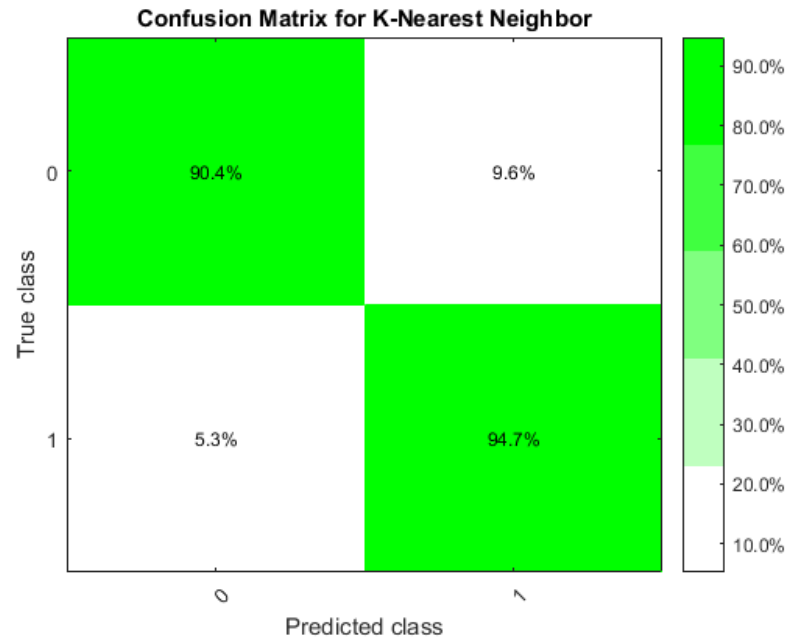
Daten_laden_zusammengefuegte_Daten.m test_modell_zusammengefuegte_Daten.m modell_validieren.m +

1
2 %trainingsdaten = Daten_laden_zusammengefuegte_Daten('C:\Users\q298084\Desktop\brauchbare_Messungen\Zusammengefuegte_Daten\dataset_zusammengefuegt_Miramas_komplett.ma
3
4 function [daten] = Daten_laden_zusammengefuegte_Daten(trainingsdaten)
5
6 % trainingsdaten laden
7 trainingsdaten = load((trainingsdaten));
8 trainingsset = trainingsdaten.neuesdatenset;
9
10 %Alle relevanten Signale in eine Matrix zusammenfügen
11 d = [trainingsset.Radwinkel, trainingsset.Gierrate, trainingsset.Laengsbeschleunigung, trainingsset.Querbeschleunigung, trainingsset.Trigger];
12 daten = array2table(d);
13 daten.Properties.VariableNames = {'Radwinkel', 'Gierrate', 'Laengsbeschleunigung', 'Querbeschleunigung', 'Trigger'};
14
15 end
```

The status bar at the bottom right indicates: Daten\_laden\_zusammengefuegte\_Daten Ln 15 Col 4

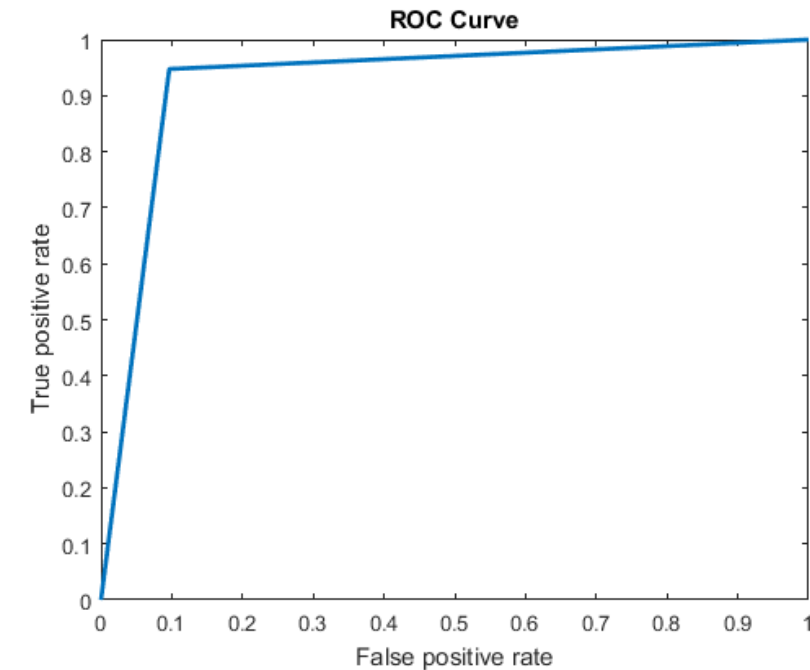
# ERGEBNISSE.

## Konfusionsmatrix: K-Nearest Neighbor & PCA Feature Extraktion



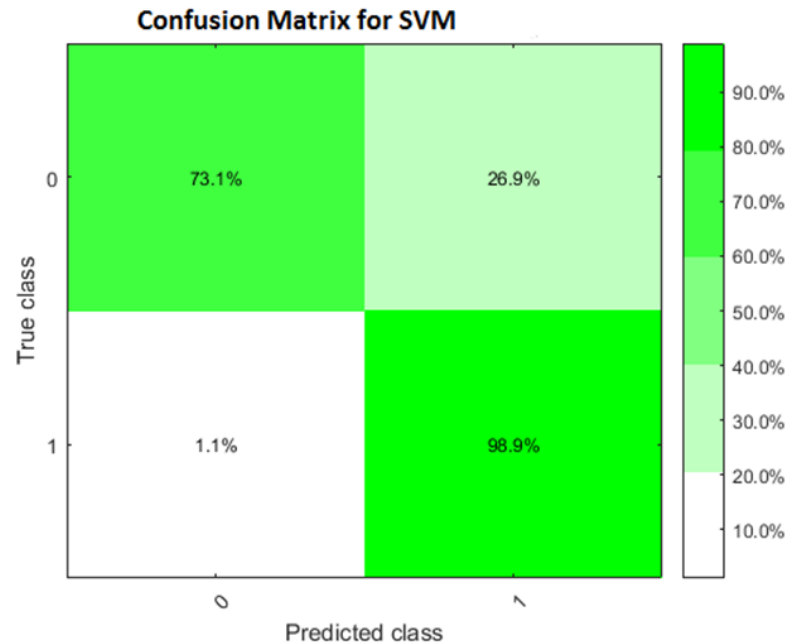
Performance of model :

|          | Predicted 0    | Predicted 1    |
|----------|----------------|----------------|
| Actual 0 | 90.35% (82943) | 9.65% (8856)   |
| Actual 1 | 5.26% (3162)   | 94.74% (57004) |



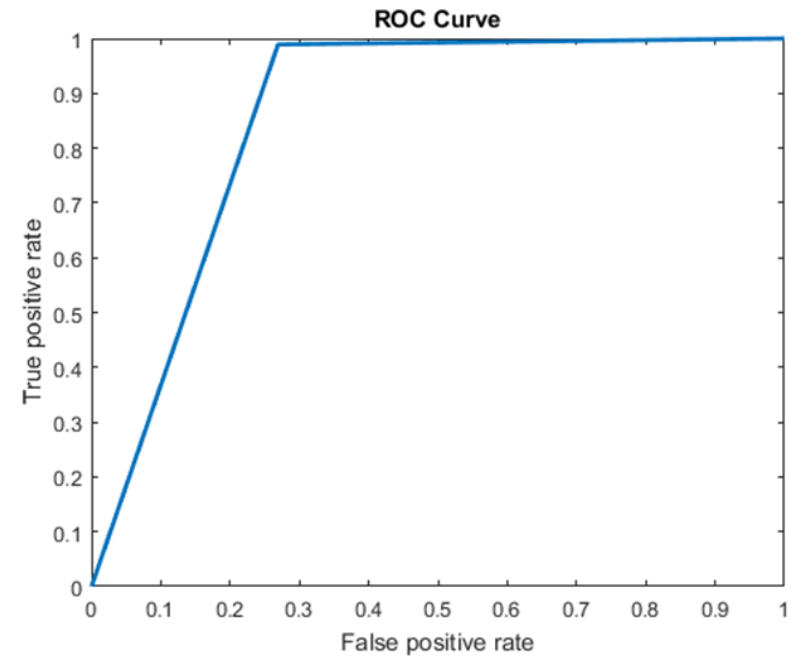
# ERGEBNISSE.

## Konfusionsmatrix: Support Vector Machine



Performance of model :

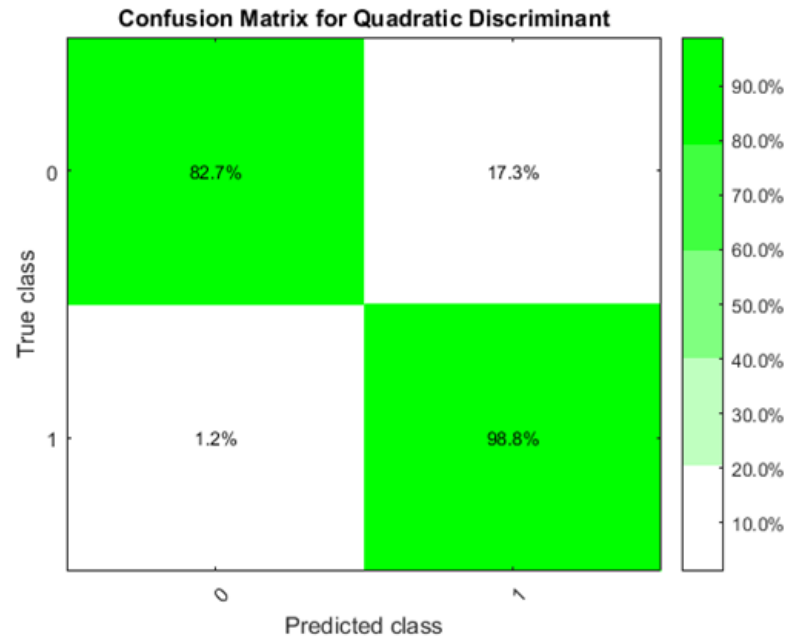
|          | Predicted 0    | Predicted 1    |
|----------|----------------|----------------|
| Actual 0 | 73.07% (67078) | 26.93% (24721) |
| Actual 1 | 1.08% (650)    | 98.92% (59516) |





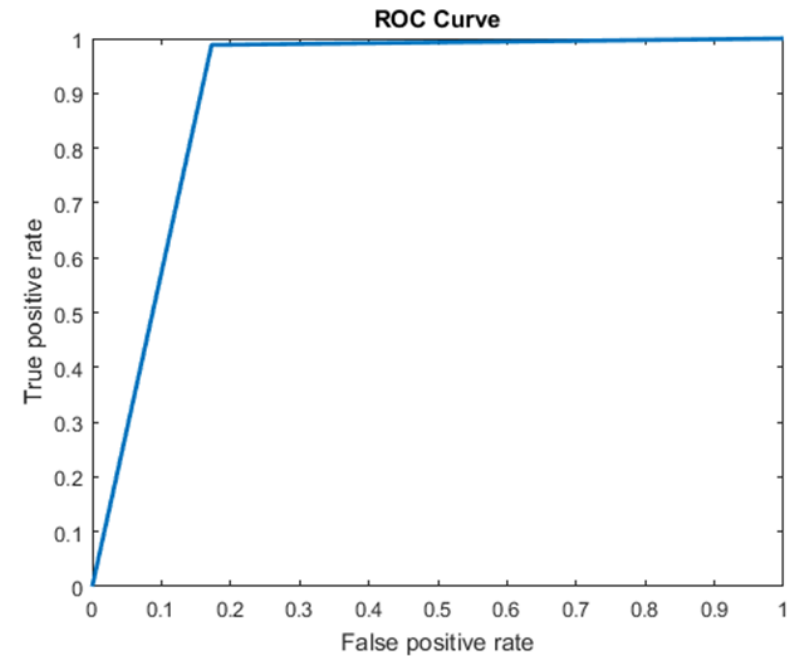
# ERGEBNISSE.

## Konfusionsmatrix: Quadratic Discriminant analysis model



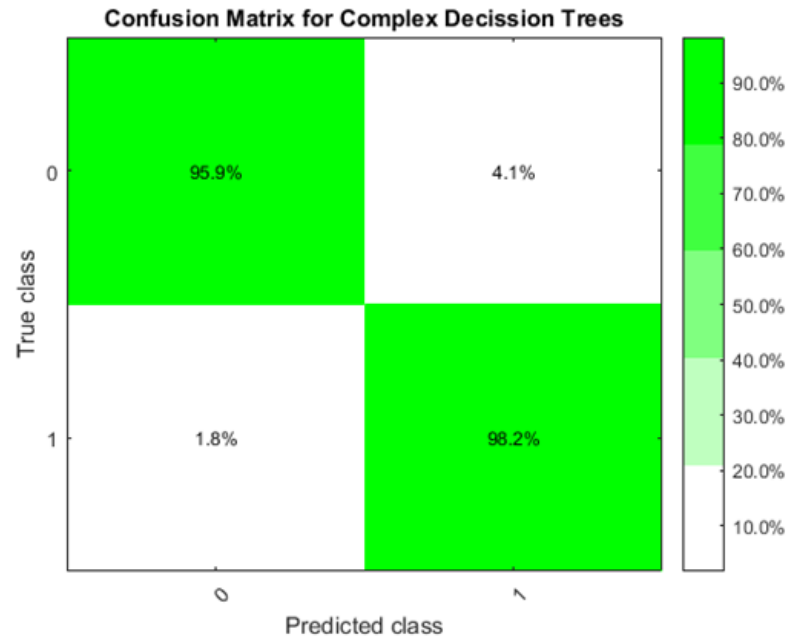
Performance of model :

|          | Predicted 0    | Predicted 1    |
|----------|----------------|----------------|
| Actual 0 | 82.73% (75946) | 17.27% (15853) |
| Actual 1 | 1.17% (701)    | 98.83% (59465) |



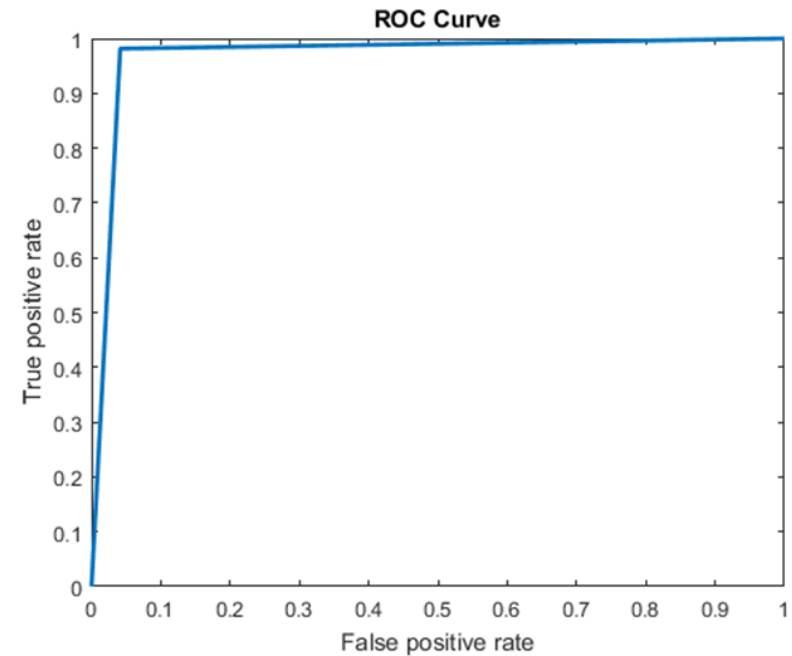
# ERGEBNISSE.

## Konfusionsmatrix: Complex Decision Trees



Performance of model :

|          | Predicted 0    | Predicted 1    |
|----------|----------------|----------------|
| Actual 0 | 95.86% (87996) | 4.14% (3803)   |
| Actual 1 | 1.84% (1109)   | 98.16% (59057) |



# HERAUSFORDERUNGEN.

Lernvorgang muss immer wieder vom Anfang der Messungen vollzogen werden,  
kein „Deltalernen“

Suche nach dem besten „Machine Learning“ Algorithmus

Sicherheitskritische Software erfordert hohe Zuverlässigkeit