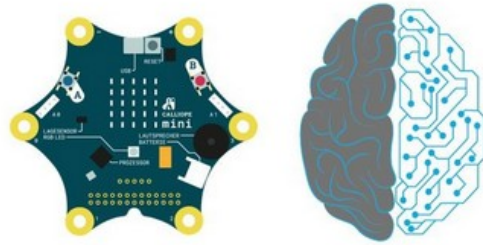


# KI@Calliope – Pipeline 1 Prototype Alpha2

## Christian Schiller, 05.12.2019



**Wissen zu Künstlicher Intelligenz  
spielerisch in die Schulen bringen:  
Autonomes Fahren mit dem Calliope  
mini**

# Benötigte Software

- PuTTY (o.ä.)  
Data Capturing
- Python
  - Anaconda3 (empfohlen)  
Python Distribution&Environment manager
  - Jupyter Notebook  
IDE (Data Preparation)
  - Orange3  
IDE (Machine Learning)



# Schritte - Überblick

## 1) Trainingsdaten von einem Calliope Mini via USB mittels PuTTY loggen

- Beispiel Logdatei = 10min Spielzeit des Autorennspiels:  
car\_race/orange3/putty-logs/carrace20191129002133.log (Rohdaten)  
car\_race/orange3/10minutesplay.log (Header entfernt)

## 2) Datenvorverarbeitung – Jupyter Notebook

- Geloggte Trainingsdaten mittels eines Jupyter Notebooks in Machine-Learning-geeignetes Format überführen
- Beispiel Ergebnis vorverarbeitete Daten basierend auf rohen Logdaten:  
car\_race/orange3/10minutesplay.csv

## 3) Machine Learning – Orange3

- Vorverarbeitete Daten in Orange3 Pipeline laden. Orange3 Workflow-Datei:  
car\_race/orange3/Pipeline1-Alpha.ows
- Beispiel Ergebnis trainiertes ML-Modell:  
car\_race/orange3/alpha-model.pkcls

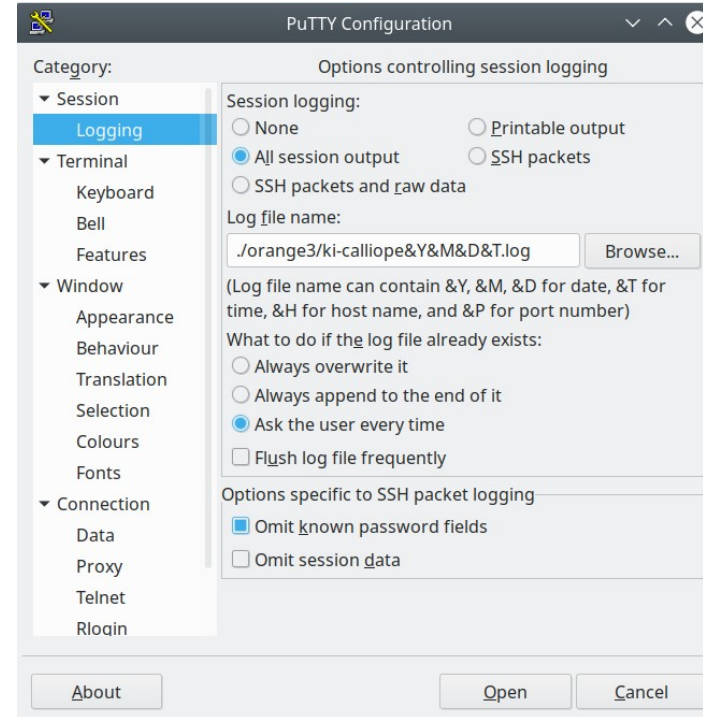
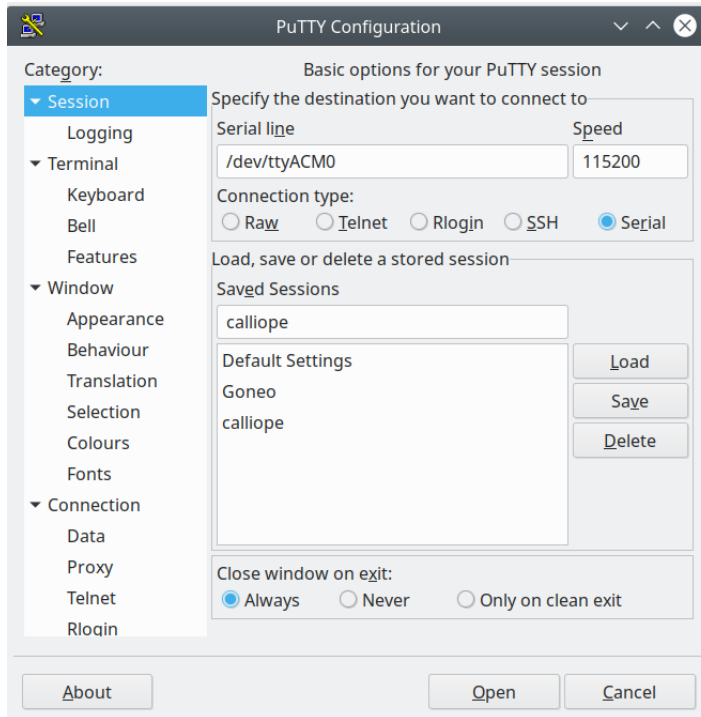
## 4) Evaluation

- Trainiertes Modell in eine 1:1 Python-Nachimplementierung des Calliope-Rennspiels laden
- Messen, wie lange das „autonome“ Auto durchhält.

} noch nicht implementiert

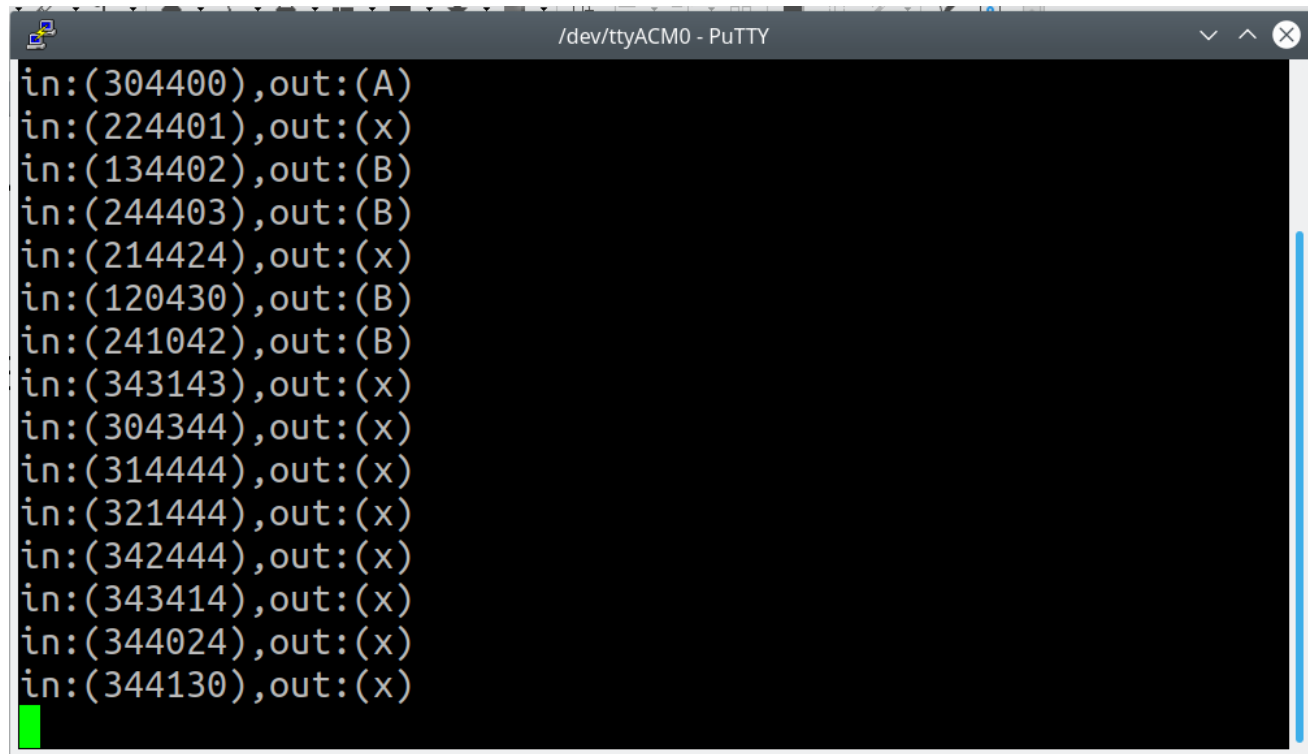
# Trainingsdaten von einem Calliope Mini via USB mittels PuTTY loggen

- Example configuration on Ubuntu 18.04LTS:  
Session and logging settings



# Trainingsdaten von einem Calliope Mini via USB mittels PuTTY loggen

- Example of logging during playing the car\_race game



The image shows a screenshot of a PuTTY terminal window. The title bar at the top reads "/dev/ttyACM0 - PuTTY". The terminal area has a black background with white text. It displays a series of 15 lines of log data, each consisting of an input value in parentheses followed by an output value. The input values are: (304400), (224401), (134402), (244403), (214424), (120430), (241042), (343143), (304344), (314444), (321444), (342444), (343414), (344024), and (344130). The output values are: (A), (x), (B), (B), (x), (B), (B), (x), (x), (x), (x), (x), (x), (x), and (x). A green cursor is visible at the bottom left of the terminal window.

```
in:(304400),out:(A)
in:(224401),out:(x)
in:(134402),out:(B)
in:(244403),out:(B)
in:(214424),out:(x)
in:(120430),out:(B)
in:(241042),out:(B)
in:(343143),out:(x)
in:(304344),out:(x)
in:(314444),out:(x)
in:(321444),out:(x)
in:(342444),out:(x)
in:(343414),out:(x)
in:(344024),out:(x)
in:(344130),out:(x)
```

# Datenvorverarbeitung Jupyter Notebook

- Run Jupyter notebook  
car\_race/orange3/00 Preprocess  
data.ipynb
- Set input logfile name
- Set output file name
- Example output CSV  
file:

10minutesplay.csv							
PlayerPos	Car1Pos	Car2Pos	Car3Pos	Car4Pos	Car5Pos	Action	
2	0	0	0	0	0	x	
2	0	0	0	0	0	x	
2	0	0	0	0	0	x	
2	1	0	0	2	0	x	
2	2	1	0	3	0	x	
2	4	2	0	4	0	x	
2	4	3	1	4	1	B	
3	4	4	3	0	3	x	
3	4	4	4	1	4	x	
3	4	4	4	2	4	A	
2	4	4	4	4	1	x	
2	4	4	4	4	2	A	
1	4	4	4	4	3	B	
2	1	1	4	0	4	x	
2	3	2	0	1	4	x	
2	4	2	1	2	4	x	

jupyter 00 Preprocess data Last Checkpoint vor 16 Stunden (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python (myenv) C

### Load raw data

```
In [3]: import os
```

```
In [39]: def load_data(path):  
    """  
    Load Dataset from File  
    """  
    input_file = os.path.join(path)  
    with open(input_file, "r") as f:  
        data = f.read()  
    return data
```

```
In [45]: raw_data = load_data('./10minutesplay.log')
```

### Preprocess raw data

```
In [41]: import pandas as pd
```

```
In [49]: preprocessed_data = None  
preprocessed_data = pd.DataFrame(columns=['PlayerPos', 'Car1Pos', 'Car2Pos', 'Car3Pos', 'Car4Pos', 'Car5Pos', 'Action'])
```

```
In [51]: for i in raw_data.split('\n'):  
    PlayerPos = i[4:5]  
    Car1Pos = i[5:6]  
    Car2Pos = i[6:7]  
    Car3Pos = i[7:8]  
    Car4Pos = i[8:9]  
    Car5Pos = i[9:10]  
    Action = i[17:18]  
    preprocessed_data = preprocessed_data.append({'PlayerPos': PlayerPos,  
        'Car1Pos': Car1Pos,  
        'Car2Pos': Car2Pos,  
        'Car3Pos': Car3Pos,  
        'Car4Pos': Car4Pos,  
        'Car5Pos': Car5Pos,  
        'Action': Action},  
        ignore_index=True)
```

```
In [52]: preprocessed_data.head()
```

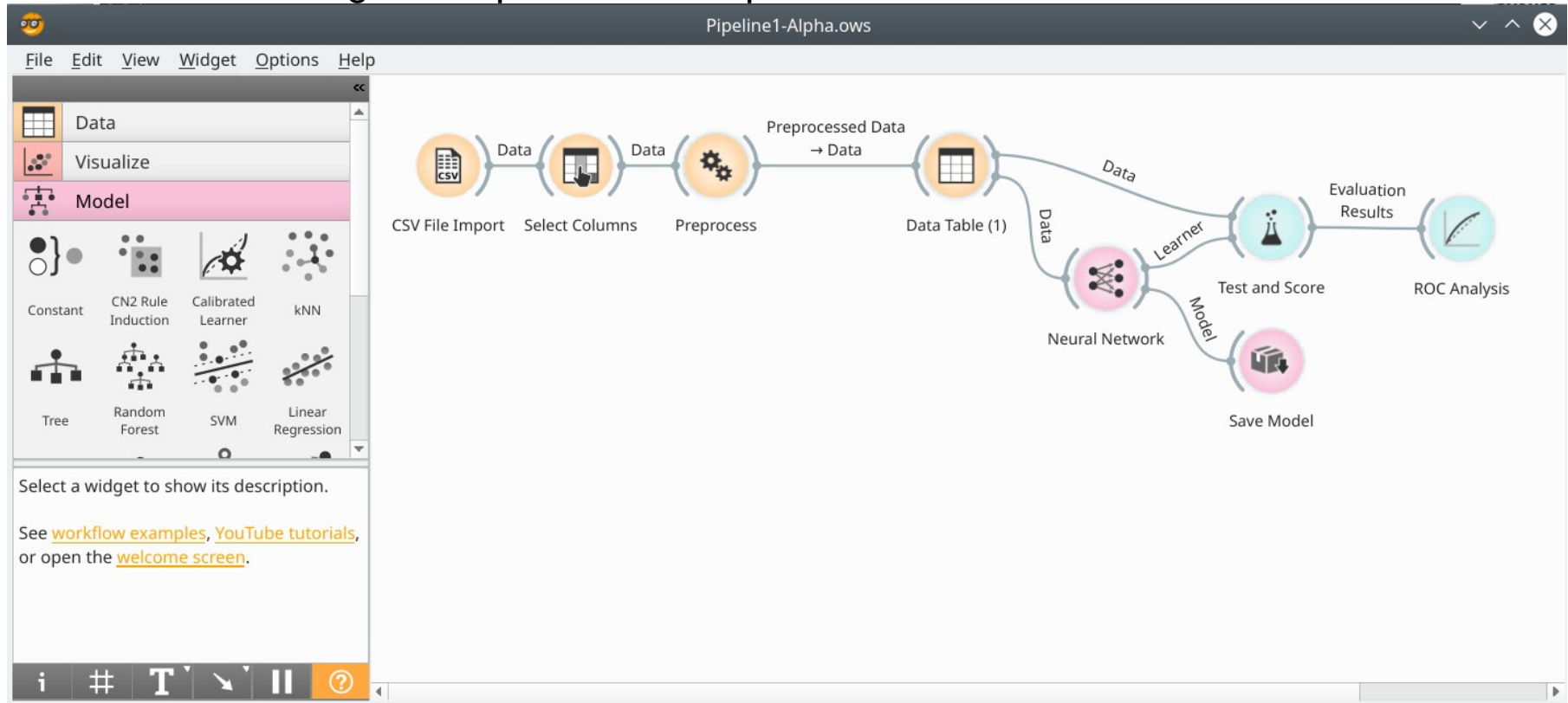
```
Out[52]:
```

	PlayerPos	Car1Pos	Car2Pos	Car3Pos	Car4Pos	Car5Pos	Action
0	2	0	0	0	0	0	x
1	2	0	0	0	0	0	x
2	2	0	0	0	0	0	x
3	2	1	0	0	2	0	x
4	2	2	1	0	3	0	x

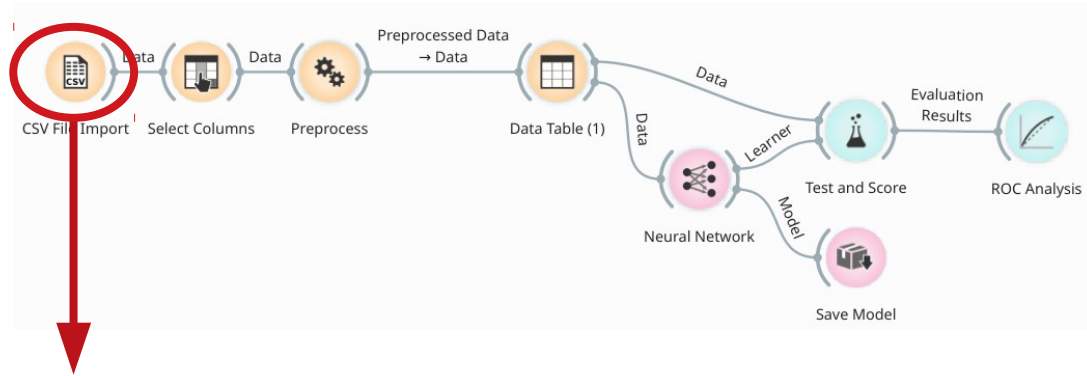
```
In [53]: preprocessed_data.to_csv('10minutesplay.csv', encoding='utf-8', index=False)
```

# Machine Learning – Orange3

- Open Orange3 Workflow  
car\_race/orange3/Pipeline1-Alpha.ows



# Machine Learning – Orange3



Import Options

Encoding: Unicode (UTF-8)

Cell delimiter: Comma

Quote character: "

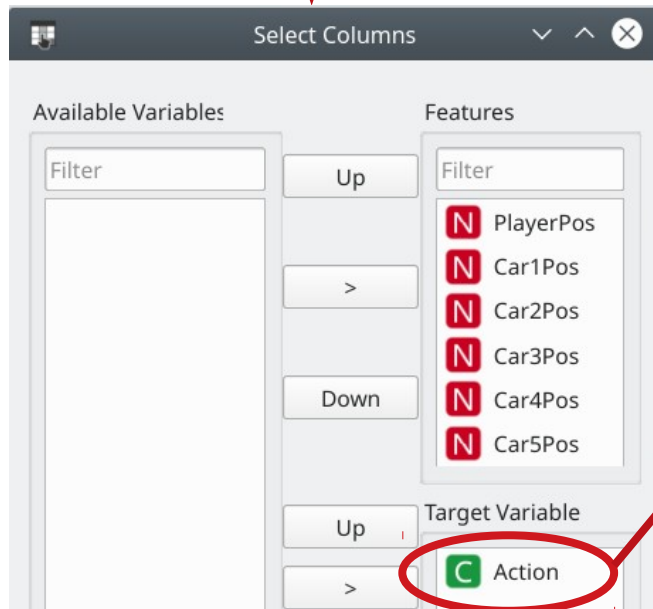
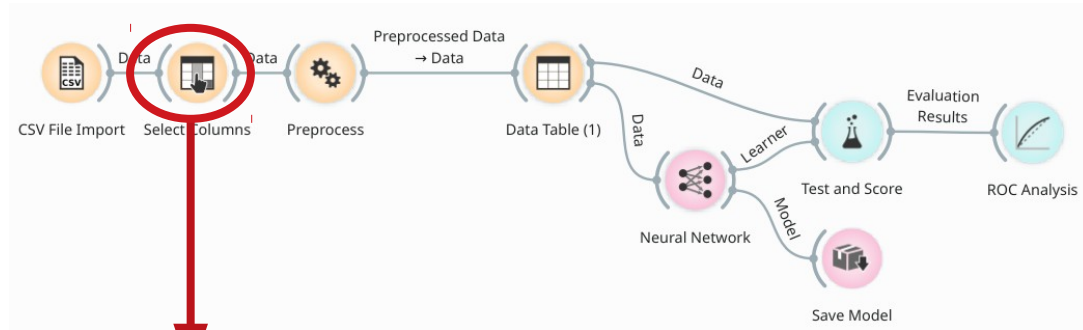
Number separators: Grouping: Decimal: .

Column type:

	N 1	N 2	N 3	N 4	N 5	N 6	C 7
	PlayerPos	Car1Pos	Car2Pos	Car3Pos	Car4Pos	Car5Pos	Action
1							
2	2	0	0	0	0	0	x
3	2	0	0	0	0	0	x
4	2	0	0	0	0	0	x
5	2	1	0	0	2	0	v

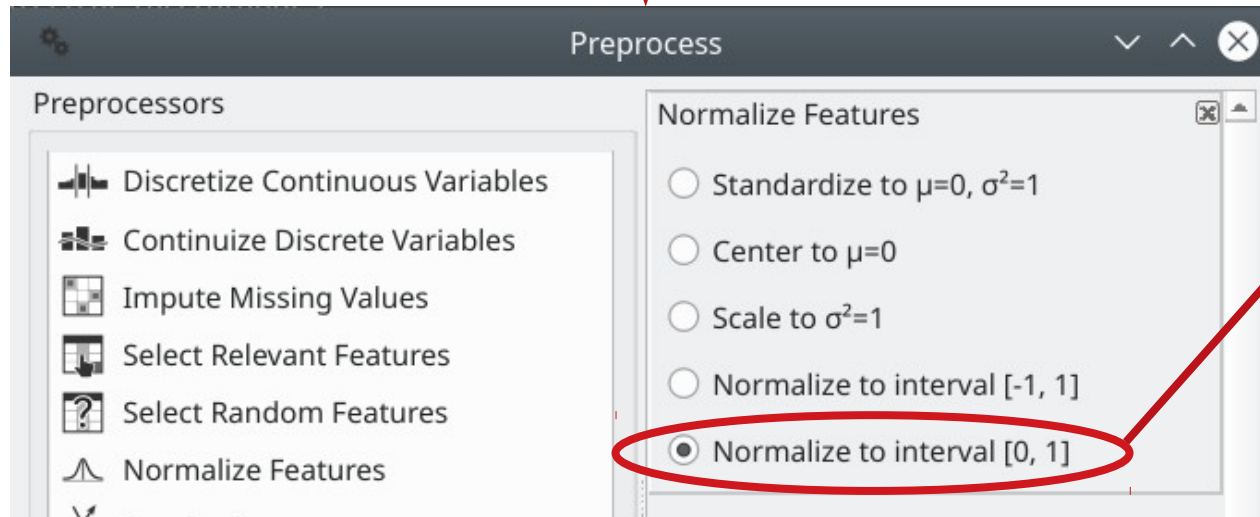
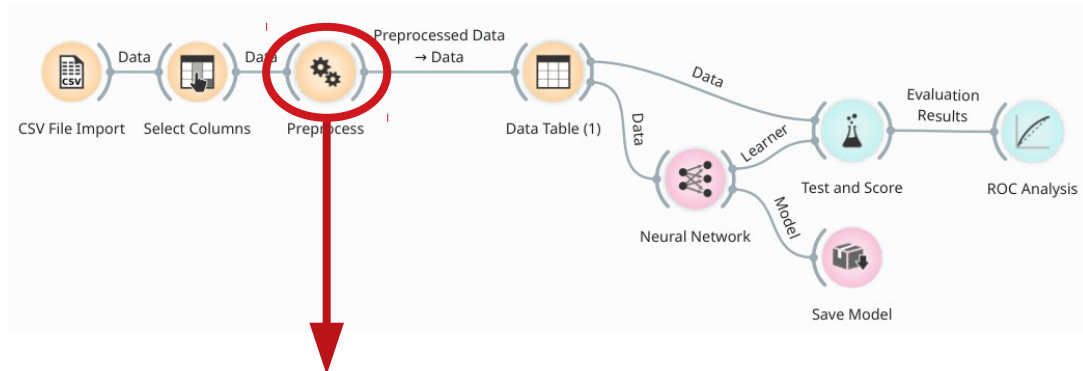


# Machine Learning – Orange3



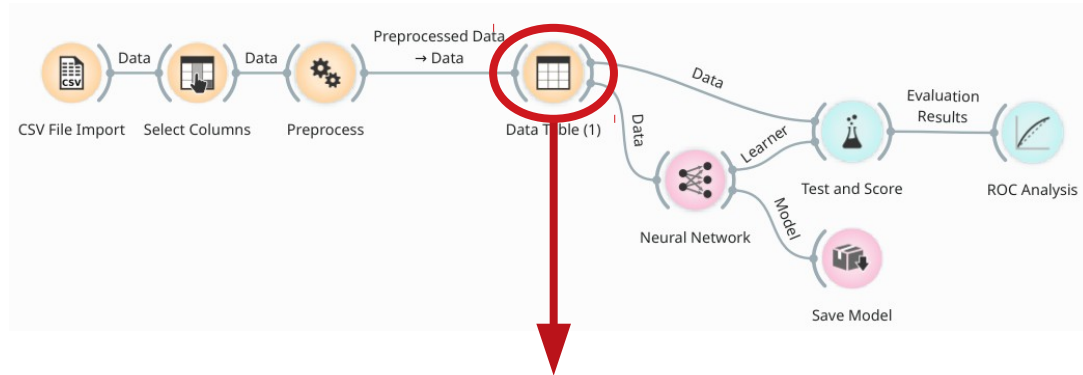
Wir wollen lernen zu steuern,  
also ist das die Target Variable.  
X = nichts zun  
A = links lenken  
B = rechts lenken

# Machine Learning – Orange3



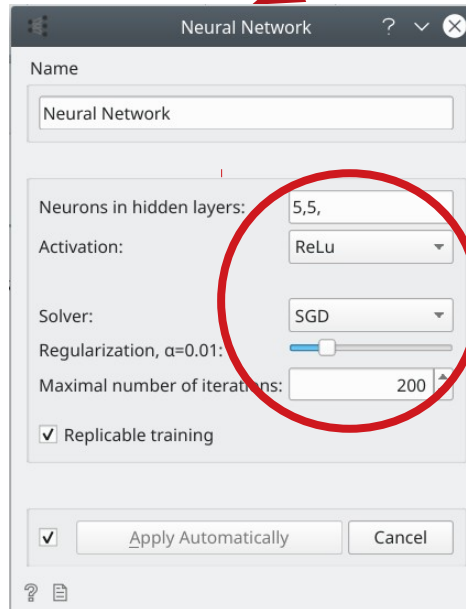
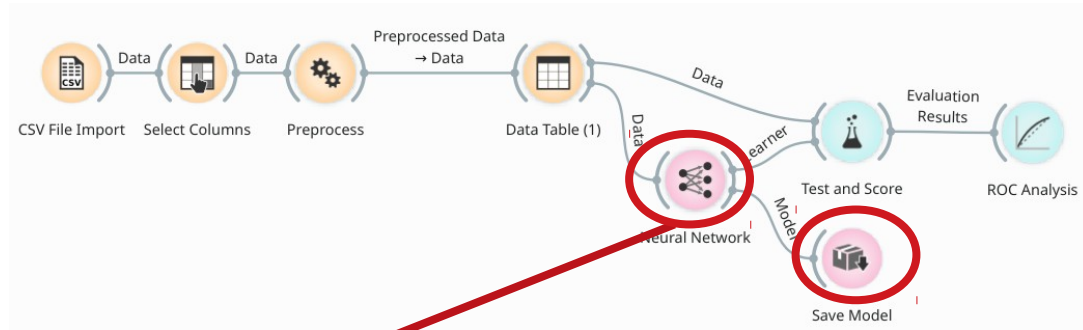
Der Input in ein neuronales Netzwerk sollte auf 0..1 skaliert sein.

# Machine Learning – Orange3



Data Table (1)								
Info								
842 instances (no missing values)								
6 features (no missing values)								
Discrete class with 3 values (no missing values)								
No meta attributes								
Variables								
<input checked="" type="checkbox"/> Show variable labels (if present)								
<input type="checkbox"/> Visualize numeric values								
<input checked="" type="checkbox"/> Color by instance classes								
Selection								
<input checked="" type="checkbox"/> Select full rows								
	Action	PlayerPos	Car1Pos	Car2Pos	Car3Pos	Car4Pos	Car5Pos	
1	x	0.5	0	0	0	0	0	
2	x	0.5	0	0	0	0	0	
3	x	0.5	0	0	0	0	0	
4	x	0.5	0.25	0	0	0.5	0	
5	x	0.5	0.5	0.25	0	0.75	0	
6	x	0.5	1	0.5	0	1	0	
7	B	0.5	1	0.75	0.25	1	0.25	
8	x	0.75	1	1	0.75	0	0.75	
9	x	0.75	1	1	1	0.25	1	
10	A	0.75	1	1	1	0.5	1	
11	x	0.5	1	1	1	1	0.25	
12	A	0.5	1	1	1	1	0.5	
13	B	0.25	1	1	1	1	0.75	
14	x	0.5	0.25	0.25	1	0	1	
15	x	0.5	0.75	0.5	0	0.25	1	
16	x	0.5	1	0.75	0.25	0.5	1	

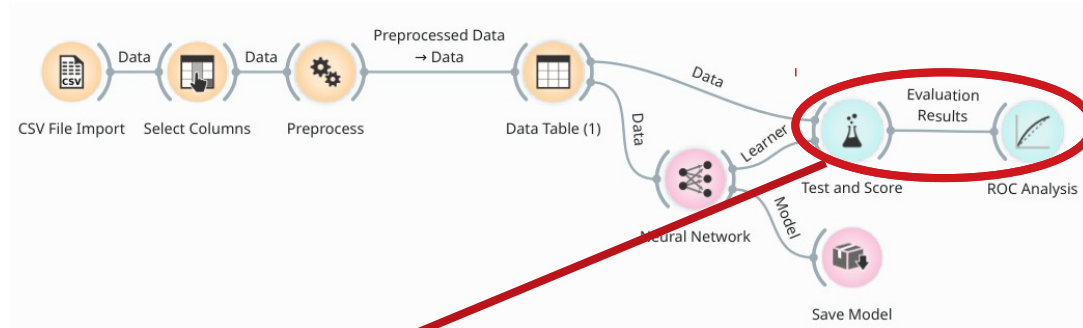
# Machine Learning – Orange3



Die Hyperparameter des neuronalen Netzwerks werden hier definiert. In der Dokumentation beispielhaft ein sehr kleines NN: Zwei Hidden Layer mit je 5 Neuronen.

Das fertig trainierte Modell kann zur Einbindung in Anwendungen gespeichert werden (Save Model Widget)

# Machine Learning – Orange3



Test and Score

Sampling

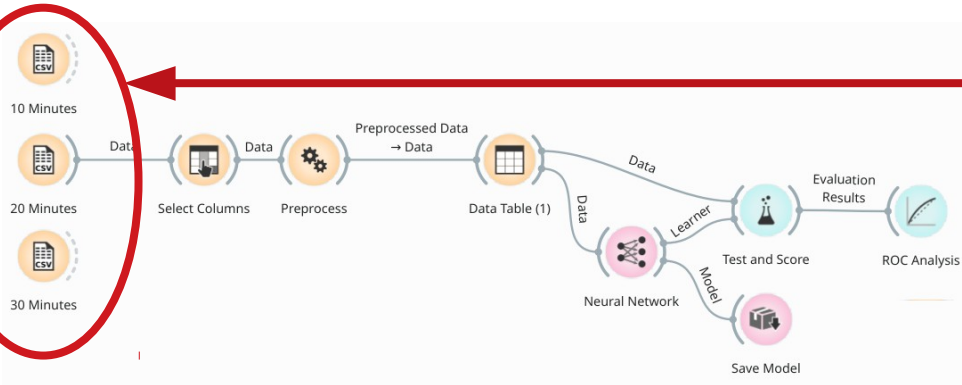
- ☐ Cross validation
- Number of folds: 10
- ☒ Stratified
- ☐ Cross validation by feature
- ☐ Selected
- ☒ Random sampling
- Repeat train/test: 10
- Training set size: 80 %

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
Neural Network	0.515	0.753	0.646	0.567	0.753

Wie man sieht ist die Leistung des Netzwerks mit nur 10min Trainingsdaten relativ schlecht. "Gut" wären Wert  $>0.8$ , besser  $>0.9$

# Machine Learning – Orange3



Je mehr Trainingsdaten gesammelt werden (hier: 10min, 20min, 30min), desto besser wird die Leistung des neuronalen Netzes in allen Metriken!

10min:

Test and Score						
Sampling		Evaluation Results				
<input type="radio"/> Cross validation						
Number of folds: 10						
Model	AUC	CA	F1	Precision	Recall	
Neural Network	0.515	0.753	0.646	0.567	0.753	

20min:

Test and Score						
Sampling		Evaluation Results				
<input type="radio"/> Cross validation						
Number of folds: 10						
Model	AUC	CA	F1	Precision	Recall	
Neural Network	0.578	0.762	0.658	0.580	0.762	

30min:

Test and Score						
Sampling		Evaluation Results				
<input type="radio"/> Cross validation						
Number of folds: 10						
Model	AUC	CA	F1	Precision	Recall	
Neural Network	0.613	0.770	0.669	0.592	0.770	