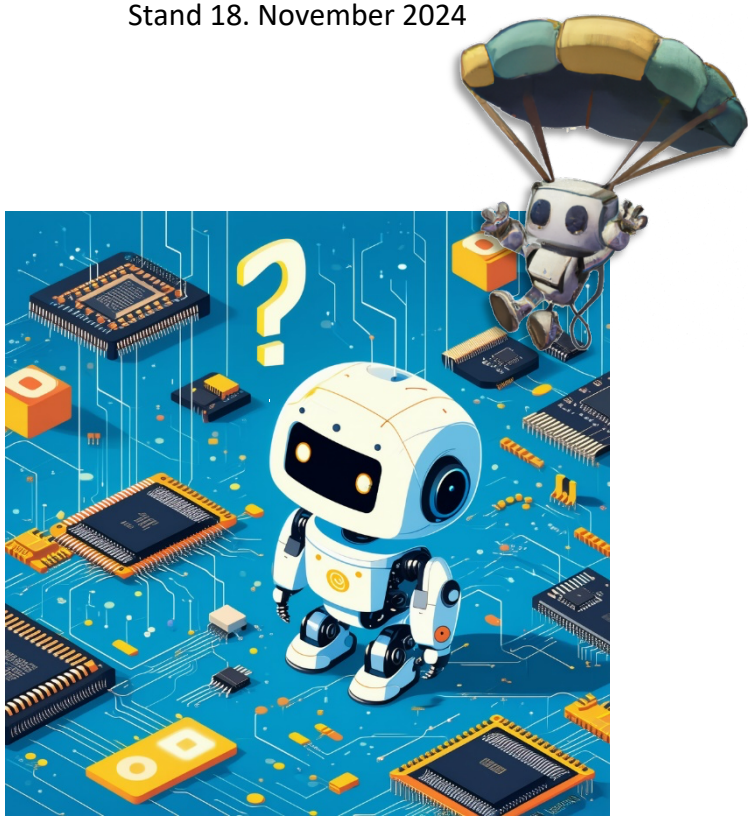
KI & Maschinelles Lernen auf dem Mikrocontroller

Teil 5: Überblick über sonstige Portierungs-Software

Der Inhalt dieses Dokuments ist verfügbar unter der Lizenz CC BY 4.0 International
Urheber: Thomas Jörg, thomas@iludis.de

Stand 18. November 2024



Welche Bibliotheken gibt es noch außer emlearn und micromlgen?

Bezeichnung	**Weblink**	**Unterstützte Algorithmen**	**Zuletzt aktualisiert**	**Ausgabeformate**
Gobber: sklearn-export	https://github.com/gobber/sklearn-export https://github.com/gobber/sklearn-export/blob/master/examples/linear_models.ipynb	Sklearn SVM Sklearn MLP Sklearn Logistic Regression Sklearn Linear Regression **Sklearn MinMaxScaler** **Sklearn Standardscaler** Sklearn Support Vector Regression	2021	JSON-Dateien mit den Koeffizienten des jeweiligen Modells.
	Installation	**Code-Beispiele**		
	$ pip install sklearn_export	https://github.com/gobber/sklearn-export/tree/master/examples		

Mit der Python-Library sklearn wurde eine Logistische Regression mit dem Iris-Datensatz trainiert. Eine zusätzliche Library namens "sklearn-export" ist dazu in der Lage, die Parameter des trainierten Modells im Json-Format auszugeben.
Das sieht dann für das Iris-Modell so aus:

{'type': 'MulticlassLogisticRegression',
'coefficients': [-0.4111635345026534, 0.5014427857156071, -0.09027925121295816, 0.9567487657508645, -0.24516272354360685,
-0.7115860422072594, -2.40210144301465, -0.21668110335362667, 2.618782546368286, -1.0169844325985156,
-0.7962492446803879, 1.8132336772789102],
'numRows': 3, 'numColumns': 4,
'intercept': [9.269282335394589, 1.9347554260562954, -11.204037761450788]}

Dieses Format soll in eine C-Programmiersprachen-Funktion umgewandelt werden, sodass zum Beispiel ein Arduino-Mikrocontroller mit diesem Modell arbeiten kann. Wandle deshalb die Parameter des Json-Codes in Arduino-C-Code um.

Bezeichnung	Weblink	Unterstützte Algorithmen	Zuletzt aktualisiert	Ausgabeformate
EmbeddedML: sklearn2c	https://github.com/EmbeddedML/sklearn2c	Bayes Decision Tree (DTClassifier) kNN C-SVC Linear Regression Polynomial Regression kMeans DBSCAN	2024	c-dateien als header, .c und .joblib-formate
	Installation	**Code-Beispiele 1**		**Code-Beispiele 2**
	pip install sklearn2c	Für jupyter-notebooks https://github.com/EmbeddedML/sklearn2c/tree/main/examples		für Implementieren auf dem MCU https://github.com/EmbeddedML/sklearn2c/blob/main/mcu_tests/classification/mcu_test_cls_dt.py

```
import os.path as osp
from sklearn.datasets import make_classification
from sklearn2c.classifiers import DTClassifier
from sklearn.model_selection import train_test_split

#samples, labels = make_classification(n_samples=100, n_features=2, n_classes=2,
random_state=42)
samples, labels = make_classification(200, 2, n_redundant=0)
train_samples, test_samples, train_labels, test_labels = train_test_split(
    samples, labels, test_size=0.2, random_state=42
)

dtc = DTClassifier()
dtc_model = osp.join("dtc_classifier.joblib")
dtc_config = osp.join("dtc_config")
dtc.train(train_samples, train_labels, save_path=dtc_model)
dtc.predict(test_samples)
dtc2 = DTClassifier.load(dtc_model)
dtc2.export(dtc_config)
```

```
import os.path as osp
from sklearn.datasets import make_blobs
from sklearn2c.clustering import Kmeans
from sklearn.model_selection import train_test_split

samples, _ = make_blobs(200, 2, centers = 3, random_state= 42)
train_samples, test_samples = train_test_split(
    samples, test_size=0.2, random_state=42
)

kmeans = Kmeans(random_state = 42)
kmeans_model = osp.join("kmeans.joblib")
kmeans_config = osp.join("kmeans_config")
kmeans.train(train_samples, save_path=kmeans_model)
kmeans_preds = kmeans.predict(test_samples)
kmeans2 = Kmeans.load(kmeans_model_dir)
kmeans2.export(kmeans_config)
```

Bezeichnung	Weblink	Unterstützte Algorithmen	Zuletzt aktualisiert	Ausgabeformate
BayesWitnesses: m2cgen	https://github.com/BayesWitnesses/m2cgen	https://github.com/BayesWitnesses/m2cgen?tab=readme-ov-file#supported-models	Apr 25, 2022	Die Bibliothek erzeugt den Code für ca. 80 (!!) verschiedene Algorithmen in ca. 16 verschiedenen Programmiersprachen (Javascript, Java, Rust, C, Python, PHP, Go, usw…)
	Installation	**Code-Beispiele**		
SEHR LEISTUNGSFÄHIG!!	pip install m2cgen	https://github.com/BayesWitnesses/m2cgen?tab=readme-ov-file#usage		

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data  # Features
y = iris.target  # Labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)
y_pred = dt_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Genauigkeit des Decision Tree Classifiers: {accuracy}")

import m2cgen as m2c
code = m2c.export_to_java(dt_classifier)
print(code)
code = m2c.export_to_c(dt_classifier)
print(code)
code = m2c.export_to_javascript(dt_classifier)
print(code)
code = m2c.export_to_python(dt_classifier)
print(code)
```

Der Code ist leicht lesbar. Hier ein Beispiel eines Iris-DecisionTrees in Python:

```
def score(input):
    if input[3] <= 0.800000011920929:
        var0 = [1.0, 0.0, 0.0]
    else:
        if input[2] <= 4.75:
            if input[3] <= 1.6500000357627869:
                var0 = [0.0, 1.0, 0.0]
            else:
                var0 = [0.0, 0.0, 1.0]
        else:
            if input[3] <= 1.75:
                var0 = [0.0, 0.5, 0.5]
            else:
                var0 = [0.0, 0.02857142857142857, 0.9714285714285714]
    return var0
```