MACHINE LEARNING – EINE EINFÜHRUNG

BEGRIFFE UND ÜBERBLICK

TECHNIKEN UND METHODEN

HERAUSFORDERUNGEN

ANWENDUNGEN

THEMEN

1. Was ist Machine Learning (ML)?
2. Anwendungen
3. Klassische ML-Verfahren: Naive Bayes, Support Vector Machines (SVM)
4. Deep Learning und neuronale Netze
5. Überwachtes Lernen und Backpropagation
6. Gütemaße: Accuracy, Precision, Recall, F1-Score
7. Herausforderungen: Underfitting & Overfitting
8. Modellierungs- und Lernstrategien
9. Gängige ML-Frameworks für Programmierer
10. Datenportale für Machine Learning

1. WAS IST MACHINE LEARNING (ML)?

- Teilbereich der Künstlichen Intelligenz (KI), der es Computern ermöglicht, Muster in Daten zu erkennen und daraus zu lernen, ohne explizit dafür programmiert zu sein.
- **Definition:** Algorithmen, die durch Erfahrung besser werden.
- **Typen von ML:**
  - Überwachtes Lernen (Supervised Learning)
  - Unüberwachtes Lernen (Unsupervised Learning)
  - Bestärkendes Lernen (Reinforcement Learning)

2. ANWENDUNGEN

- **Zwei Hauptaufgaben von ML: Klassifikation und Regression**
- **Beispiele für Anwendungen:**
  - Bild- und Spracherkennung
  - Empfehlungen (Netflix, Amazon, bester Algorithmus)
  - Automatisiertes Fahren, ADAS
  - Natural Language Processing
  - Large Language Models (LLM), GPT, Chatbots

3. ZWEI KLASSISCHE ML-VERFAHREN

- **Naive Bayes**
  - **Grundidee:** Verwendet den Satz von Bayes, um die Wahrscheinlichkeit einer Klassenzugehörigkeit basierend auf bekannten Merkmalen zu berechnen.
  - **Anwendung:** Textklassifikation (z.B. Spam-Filter).
  - **Vorteile:** Schnell, einfach, gut geeignet für große Datensätze.
- **Support Vector Machines (SVM)**
  - **Grundidee:** Findet eine Hyperplane, die die Daten in verschiedene Klassen trennt. Ziel ist es, den maximalen Abstand zwischen den Klassen zu maximieren.
  - **Anwendung:** Bildklassifizierung, Textmining.
  - **Vorteile:** Effektiv bei hochdimensionalen Daten.

4. DEEP LEARNING UND NEURONALE NETZE

- **Neuronale Netze**
  - **Struktur:**
    - **Neuronen:** Eingabewerte, gewichtete Verbindungen und Aktivierungsfunktionen.
    - **Schichten:** Eingabeschicht, versteckte Schichten, Ausgabeschicht.
  - **Grundidee:** Jedes Neuron empfängt Eingaben, verarbeitet diese durch eine Aktivierungsfunktion und gibt sie an die nächste Schicht weiter.
- **Aktivierungsfunktionen:**
  - **Sigmoid:** Wandelt Eingabewerte in Werte zwischen 0 und 1 um.
  - **ReLU (Rectified Linear Unit):** Lässt nur positive Werte durch, um nichtlineare Beziehungen abzubilden.

5. ÜBERWACHTES LERNEN UND BACKPROPAGATION

- **Überwachtes Lernen**
  - **Definition:** Lernverfahren, bei dem das Modell auf gelabelten Daten trainiert wird.
  - **Anwendung:** Klassifikation, Regression.
  - **Ablauf:** Eingaben -> Modell -> Vorhersage -> Vergleich mit dem tatsächlichen Wert -> Fehlerberechnung.
- **Backpropagation**
  - **Definition:** Algorithmus, der verwendet wird, um die Gewichte in einem neuronalen Netz anzupassen, basierend auf dem Fehler der Vorhersage.
  - **Vorgehen:**
    - Vorwärtsdurchlauf: Berechnung der Ausgabe.
    - Rückwärtsdurchlauf: Fehlerverteilung und Anpassung der Gewichte mit Gradientenabstieg.
  - **Beispiel:** Anpassung der Gewichte bei einer Spracherkennungsaufgabe.

6. GÜTEMAßE FÜR BINÄRE KLASSIFIZIERER

- **Beispiel: Von T=100 Samples weisen P=51 ein Merkmal auf, d.h.**
  - **P = 51 haben das Merkmal (P=positiv),**
  - **N = 49 haben das Merkmal nicht (N=Negativ), T = P+N**
  - **Eine ML-Methode trifft folgende Vorhersagen:**
    - **TP = 48 Samples, FP = 4 Samples, TN = 45 Samples, FN = 3 Samples.**
  - **Accuracy:** Anteil korrekt klassifizierter Daten. Accuracy=(TP + TN) / T = 0.93
  - **Precision:** Genauigkeit der positiven Vorhersagen. Precision = TP / (TP + FP) = 0.92
  - **Recall:** Anteil der korrekt erkannten positiven Beispiele. Recall = TP / (TP + FN) = 0.94
  - **F1-Score:** Harmonisches Mittel von Precision und Recall. F1-Score = 0.93
- **Confusion Matrix**
  - Eine Matrix, die die tatsächlichen Klassen und die vorhergesagten Klassen eines Klassifikationsproblems gegenüberstellt. Sie wird verwendet, um die Vorhersagegüte besser zu verstehen.

7. HERAUSFORDERUNGEN: UNDERFITTING & OVERFITTING

- **Underfitting**
  - **Definition:** Das Modell ist zu einfach und kann die Daten nicht gut genug repräsentieren.
  - **Beispiel:** Lineare Regression bei komplexen Daten.
- **Overfitting**
  - **Definition:** Das Modell ist zu komplex und passt sich den Trainingsdaten zu stark an, was zu schlechten Ergebnissen auf neuen Daten führt.
  - **Beispiel:** Ein tiefes neuronales Netz, das zufälliges Rauschen lernt.
- **Lösungsmöglichkeiten:**
  - **Regularisierung:** Fügt eine Strafe für zu komplexe Modelle hinzu (z.B. Lasso, Ridge).
  - **Kreuzvalidierung:** Daten werden in Trainings- und Testdatensätze aufgeteilt, um die Generalisierungsfähigkeit zu überprüfen.
  - Notausstieg/Early Stopping

8. MODELLIERUNGSSTRATEGIEN UND LERNSTRATEGIEN

- **Modellierungsstrategien**
  - **Feature Engineering:** Auswahl und Transformation der wichtigsten Merkmale, die das Modell benötigt.
  - **Hyperparameter Tuning:** Feinjustierung der Parameter (z.B. Lernrate, Anzahl der Neuronen), um die Modellleistung zu verbessern.
- **Lernstrategien**
  - **Batch Learning:** Daten werden in Batches verarbeitet, um den Speicherverbrauch zu optimieren.
  - **Online Learning:** Modell lernt kontinuierlich mit neuen Daten.
  - **Transfer Learning:** Ein bereits trainiertes Modell wird für eine neue, aber ähnliche Aufgabe verwendet (z.B. vortrainierte Bildklassifizierungsmodelle).

9. GÄNGIGE ML-FRAMEWORKS FÜR PROGRAMMIERER

- **TensorFlow:** Open-Source-Bibliothek von Google für ML und DL.
  - **Vorteile:** Weit verbreitet, unterstützt verteiltes Training, große Community. https://www.tensorflow.org
- **PyTorch:** Bibliothek von Facebook, die sich für Deep Learning eignet.
  - **Vorteile:** Flexibilität, dynamische Berechnungsgrafen, einfache Handhabung. https://pytorch.org
- **Scikit-Learn:** Bibliothek für maschinelles Lernen in Python mit einfachen ML-Algorithmen.
  - **Vorteile:** Intuitive API, gut geeignet für Standard-ML-Verfahren. https://scikit-learn.org/stable/
- **Keras:** High-Level-API, die auf TensorFlow aufbaut, ideal für schnelle Prototypenerstellung.
- **OpenNN**: C++-Library/API nur für neuronale Netze.
  - Muss für die Verwendung kompiliert werden, Usability komplexer als bei Python-APIs
  - Outperforms Tensorflow (siehe Benchmarking auf der Homepage). https://www.opennn.net
- **ML.Net** (C#.Net).
  - Verbreitetes Framework im .Net Ecosystem. https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet

10. GÄNGIGE DATENPORTALE FÜR MACHINE LEARNING-PROJEKTE

- **Klassische Datasets**
  - **Iris-Dataset: Klassifizierung von Irispflanzenblüten nach deren Geometrie.**
  - **MNIST Dataset: Klassifizierung von handgeschriebenen Ziffern: https://yann.lecun.com/exdb/mnist/**
- **Datenportale:**
  - **Kaggle:** Plattform für ML-Wettbewerbe mit zahlreichen Datensätzen: https://www.kaggle.com
  - **UCI Machine Learning Repository:** Umfassende Sammlung von klassischen ML-Datensätzen: https://archive.ics.uci.edu
  - **Google Dataset Search:** Suchmaschine speziell für Datensätze: https://datasetsearch.research.google.com
  - **OpenML:** Plattform für offene Datensätze, ML-Experimente und Benchmarking: https://www.openml.org
  - **NASA Data Sets**: https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/

WIE LERNE ICH MEHR ÜBER ML?

- A. C. Müller, S. Guido: **Introduction to Machine Learning with Python**, O'Reilly Media, Inc., 2016: Anwenderorientierte Einführung in wichtige Techniken und Frameworks.
- T. Rashid: Neuronale Netze selbst programmieren, O'Reilly Media, Inc., 2017: Verständnisorientierte Einführung, Formeln und Hintergründe von Neuronalen Netzwerken, Forward- und Backprogration in Python ohne ML-Frameworks (lediglich NumPy)
- Hugging Face: https://huggingface.co
- Applied Data Science Course at Columbia Engineering (berufsbegleitend, ca. 12 Wochen)
- Roberto López Gonzáles: Neural Networks for Variational Problems in Engineering, PhD Thesis
- Raúl Rojas: Neural Networks. Springer Berlin Heidelberg

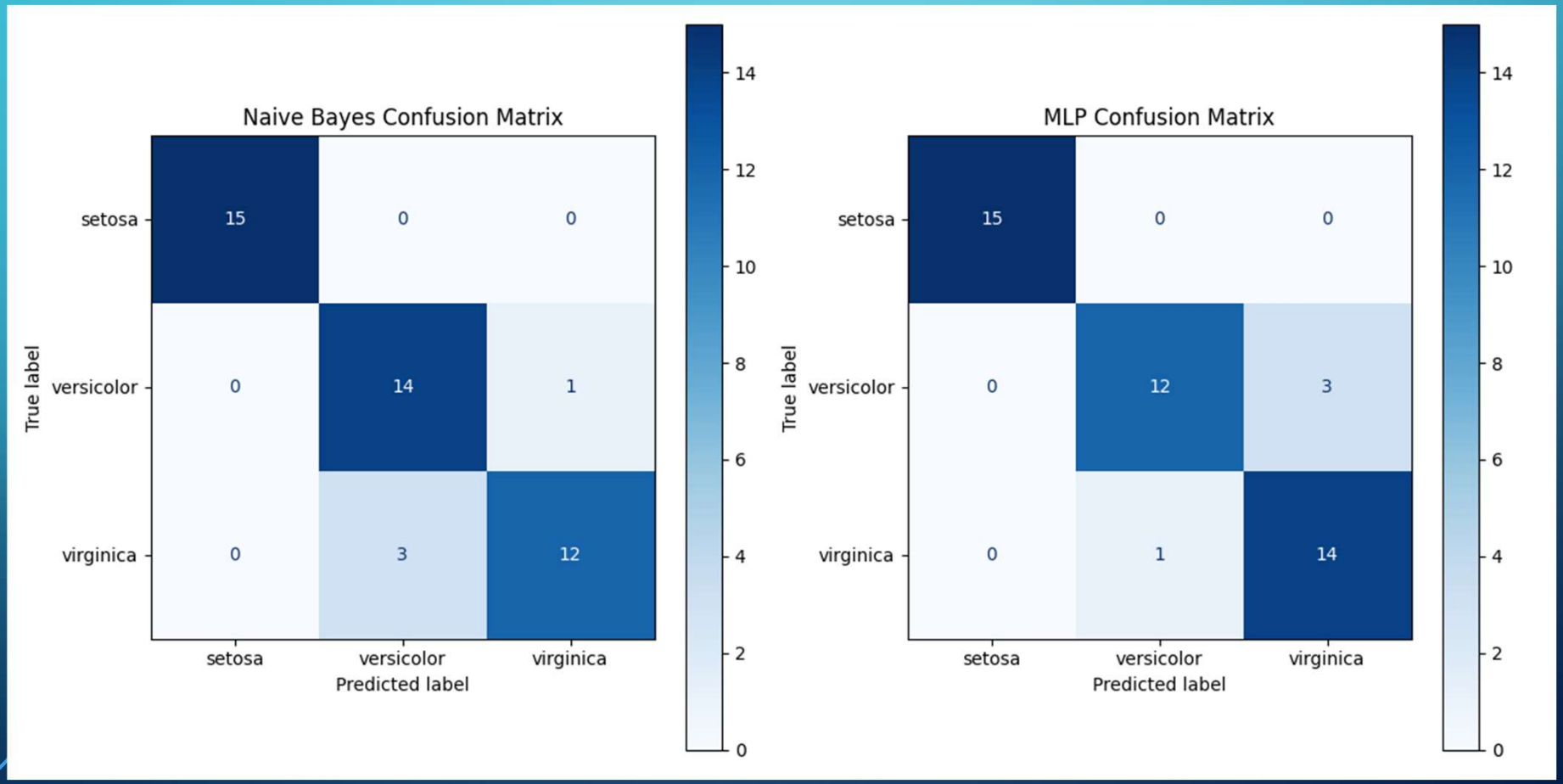
BEISPIELE

- Neuronales Netz für Klassifizierung der Iris-Dataset (R. Fisher, 1936)
  - Iris-Dataset: https://de.wikipedia.org/wiki/Schwertlilien-Datensatz
  - VS-C++-Projektmappe: https://github.com/Haasrobertgmxnet/OwnNeuralNetwork
- Neuronales Netz für Klassifizierung der MNIST-Daten:
  - https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork

WEITERES BEISPIEL 1

- Klassifizierung Iris-Daten mit generiertem Python-Code
- Prompt in ChatGPT: „*Schreibe ein einfaches Python-Programm zum Training und Test zweier ML-Klassifizierer. Verwende Naive-Bayes und Multilayer-Perceptron als Klassifizierer. Verwende Tensorflow Keras. Verwende sinnvolle Gütemaße. Wenn sinnvoll, erzeuge aussagekräftige Diagramme. Verwende eine einfache Dataset, wenn möglich Engineering/Aircraft.*“
- Den generierten Python-Code bspw. einfach in ein VS-Projekt einbetten und dort ausführen. Darauf achten, dass alle notwendigen Python-Module installiert sind

WEITERES BEISPIEL 2



WEITERES BEISPIEL 3 – ETWAS CODE

```
# 2. Multilayer-Perceptron-Modell mit TensorFlow/Keras
# One-Hot-Encoding der Labels für MLP
encoder = OneHotEncoder(sparse_output=False)
y_train_oh = encoder.fit_transform(y_train.reshape(-1, 1))
y_test_oh = encoder.transform(y_test.reshape(-1, 1))

# Standardisierung der Features für MLP
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Aufbau des MLP-Modells
mlp_model = Sequential([
    Dense(16, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(8, activation='relu'),
    Dense(len(class_names), activation='softmax')
])
mlp_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Training
mlp_model.fit(X_train_scaled, y_train_oh, epochs=50, batch_size=8, verbose=0)

# Test
mlp_loss, mlp_accuracy = mlp_model.evaluate(X_test_scaled, y_test_oh, verbose=0)
y_pred_mlp = np.argmax(mlp_model.predict(X_test_scaled), axis=1)
```