Python-Befehle zum Kurs
KI für Alle – Einführung in die Künstliche Intelligenz

KI für Alle

Stand 26. März 2025 (Version 2)

Inhaltsverzeichnis

1 Grundlagenbefehle **1**

2 Listen **2**

3 Numpy **3**

4 SciPy Stats **5**

5 Plotly Express **6**

6 Sklearn **7**

7 Keras **10**

8 Pandas **12**

9 Sonstige **13**

1 Grundlagenbefehle

- `print(`$a_1, a_2, \ldots, a_n$`)`
  - Gibt a_1 bis a_n auf der Konsole aus.
  - Beispiel:
    ```
    i=4
    print("Anzahl Durchläufe: ", i)
    ```
  - Ausgabe: `Anzahl Durchläufe: 4`

- `import` *modulname*
  - Importiert das Modul *modulname*
  - Optional kann mit `as` *neuerName* ein neuer Name für das Modul vergeben werden.
  - Beispiel:
    ```
    import numpy as np
    ```
  - Import das Modul `numpy` unter dem Namen `np`
- `type`(*variablenname*)
  - Gibt den Datentyp der Variable *variablenname* zurück.
  - Beispiel:
    ```
    type(zahl)
    ```
  - Gibt den Datentyp der Variable `zahl` zurück.
- `int`(*variablenname*)
  - Wandelt die Variable *variablenname* in einen Integer um.
  - Existiert auch für andere Datentypen, z.B. `str`(*variablenname*) für Strings und `float`(*variablenname*) für Fließkommazahlen.
  - Beispiel:
    ```
    int(zahl)
    ```
  - Wandelt die Variable `zahl` in eine Integerzahl um.

2 Listen

- *listenname*`=[`$a_1, a_2, \ldots, a_n$`]`
  - Manuelle Erstellung einer Liste *listenname* mit den Elementen a_1 bis a_n
  - Beispiel:
    ```
    erscheinungsjahr = [1911, 1945, 1865]
    ```
  - Erstellt eine Liste `erscheinungsjahr` mit den Einträgen 1911, 1945, 1865.
- *listenname*`.append(`*listenelement*`)`
  - Anhängen eines Elementes *listenelement* an das Ende einer Liste *listenname*
  - Beispiel:
    ```
    krimis.append("Mord im Orient-Express")
    ```

  - Hängt „Mord im Orientexpress“ als Listenelement an die Liste `krimis` an.

- *listenname*`.insert(`*position*, *listenelement*`)`
  - Einfügen eines Elementes *listenelement* an Position *position* einer Liste *listenname*
  - Beispiel:
    ```
    krimis.insert(1, "Alibi")
    ```
  - Fügt „Alibi“ an Position 1 der Liste `krimis` ein.

- `len(`*listenname*`)`
  - Gibt die Länge einer Liste *listenname* zurück.
  - Beispiel:
    ```
    len(kinderbuecher)
    ```
  - Gibt die Länge der Liste `kinderbuecher` zurück.

- `del` *listenname*`[`$anfang$, *ende*`]`
  - Löscht aus Liste *listenname* die Listenelemente von Position $anfang$ (inklusiv) bis Position *ende* (exklusiv).
  - Beispiel:
    ```
    del krimis[0:2]
    ```
  - Löscht die Elemente an Position 0 bis 1 aus der Liste `krimis`.

- *listenname*`.clear()`
  - Löscht alle Elemente aus Liste *listenname*.
  - Beispiel:
    ```
    krimis.clear()
    ```
  - Löscht alle Elemente aus der Liste `krimis`.

3 Numpy

```
import numpy as np
```

- `np.array(`*arrayinhalt*`)`
  - Erstelle manuell ein multidimensionales Numpy Array.

  - Beispiel:
    ```
    np.array([[45, 1.82, 78.5], [35, 1.73, 56.3],
              [67, 1.87, 82], [23, 1.95, 90.4]])
    ```
  - Erstellt das Array $\begin{pmatrix} 45 & 1.82 & 78.5 \\ 35 & 1.73 & 56.3 \\ 67 & 1.87 & 82 \\ 23 & 1.95 & 90.4 \end{pmatrix}$.

- *arrayname*`.shape`
  - Erhalte die Dimensionen des Numpy-Arrays *arrayname*, also die Anzahl der Zeilen und Spalten.
  - Beispiel:
    `patienten_daten.shape`
  - Gibt die Dimension des Arrays `patienten_daten` zurück.

- *arrayname*`.dtype`
  - Erhalte den Datentyp des Numpy-Arrays *arrayname*.
  - Beispiel:
    `patienten_daten.dtype`
  - Gibt den Datentyp des Arrays `patienten_daten` zurück.

- `np.where(`*bedingung*, *wertJa*, *wertNein*)
  - Falls die optionalen Parameter *wertJa* und *wertNein* spezifiziert sind, wird ein numpy-Array ausgegeben. In diesem sind alle Werte auf *wertJa* gesetzt, die die Bedingung *bedingung* erfüllen, und auf *wertNein*, die diese Bedingung nicht erfüllen. Wenn nur die Bedingung *bedingung* angegeben wird, gibt die Funktion die Indizes der Werte zurück, die *bedingung* erfüllen.
  - Beispiel 1:
    `np.where(X['who'] == 'child', 1, 0)`
  - Gibt eine Kopie der Spalte `who` im Array `X` zurück. In dieser Kopie ist ein Eintrag auf 1 gesetzt, wenn in `X['who']` an dieser Stelle der String `child` steht. Sonst ist der entsprechende Eintrag auf 0 gesetzt.
  - Beispiel 2:
    `np.where(patienten_daten[:,0] > 35)`
  - Gibt die Indizes der Patienten in dem Array `patienten_daten` zurück, die über 35 Jahre alt sind.

Für eine ausführliche Dokumentation siehe: `https://numpy.org/doc/1.23/`

4 SciPy Stats

`from scipy import stats`

- `stats.tmin(`*arrayname*`, axis = 0)`
  - Erhalte das spaltenweise Minimum des Arrays *arrayname*.
  - Optional: Angabe einer unteren Schranke mit `lowerlimit = (`*untereSchranken*)
  - Beispiel:
    `stats.tmin(patienten_daten, axis = 0, lowerlimits = (23, 1.73, 60))`
  - Erhalte das spaltenweise Minimum des Arrays `patienten_daten`, wobei nur Werte ab jeweils 23, 1.73 bzw. 60 berücksichtigt werden.
- `stats.tmax(`*arrayname*`, axis = 0)`
  - Erhalte das spaltenweise Maximum des Arrays *arrayname*.
  - Optional: Angabe einer oberen Schranke mit `upperlimit = (`*obereSchranken*)
  - Beispiel:
    `stats.tmax(patienten_daten, axis = 0)`
  - Erhalte das spaltenweise Maximum des Arrays `patienten_daten`.
- `stats.mode(`*arrayname*`, axis = 0)`
  - Erhalte den spaltenweisen Modus des Arrays *arrayname*.
  - Beispiel:
    `stats.mode(patienten_daten, axis = 0)`
  - Erhalte den spaltenweisen Modus des Arrays `patienten_daten`.
- `stats.tmean(`*arrayname*`, axis = 0)`
  - Erhalte den spaltenweisen Durchschnitt des Arrays *arrayname*. Optional: Angabe einer unteren Schranke mit `lowerlimit = (`*untereSchranken*) und einer oberen Schranke mit `upperlimit = (`*obereSchranken*).
  - Beispiel:
    `stats.tmean(patienten_daten, axis = 0)`
  - Erhalte den spaltenweise Durchschnitt des Arrays `patienten_daten`.
- `np.median(`*arrayname*`, axis = 0)`
  - Erhalte den spaltenweisen Median des Arrays *arrayname*.
  - Beispiel:
    `np.median(patienten_daten, axis = 0)`

- Erhalte den spaltenweisen Median des Arrays `patienten_daten`.

Für eine ausführliche Dokumentation siehe: `https://docs.scipy.org/doc/scipy/reference/stats.html`

5 Plotly Express

```
import plotly.express as px
```

- `px.scatter(`*datframename*, *weitereEinstellungen*)
  - Erzeugt einen Scatterplot mit dem Dataframe *datframename* und den Einstellungen *weitereEinstellungen*.
  - Als weitere Einstellungen können zum Beispiel `x =` *attributname*, `y =` *attributname*, `size =` *attributname*, `color =` *attributname* und `symbol =` *attributname* verwendet werden.
  - Existiert auch in der dreidimensionalen Variante `scatter_3d`.
  - Beispiel:
    `px.scatter(data, x = 'lifeExp', y = 'gdpPercap'))`
  - Erzeugt einen Scatterplot mit den Daten `data`, bei dem auf der x-Achse das Attribut `lifeExp` und auf der y-Achse das Attribut `gdpPercap` aufgetragen ist.
- `px.line(`*datframename*, *weitereEinstellungen*)
  - Erzeugt ein Liniendiagramm mit dem Dataframe *datframename* und den Einstellungen *weitereEinstellungen*.
  - Als weitere Einstellungen können zum Beispiel `x =` *attributname*, `y =` *attributname* und `color =` *attributname* verwendet werden.
  - Beispiel:
    `px.line(dataGermany, x = 'year', y = 'lifeExp')`
  - Erzeugt ein Liniendiagramm mit den Daten `dataGermany`, bei dem auf der x-Achse das Attribut `year` und auf der y-Achse das Attribut `lifeExp` aufgetragen ist.
- `px.bar(`*datframename*, *weitereEinstellungen*)
  - Erzeugt ein Säulendiagramm mit dem Dataframe *datframename* und den Einstellungen *weitereEinstellungen*.
  - Als weitere Einstellungen können zum Beispiel `x =` *attributname*, `y =` *attributname* und `color =` *attributname* verwendet werden.

  - Beispiel:
    `px.bar(dataDeFrPo, x = 'country', y = 'gdpPercap', color = 'year')`
  - Erzeugt ein Säulendiagramm mit den Daten `dataDeFrPo`, bei dem auf der x-Achse das Attribut `year` und auf der y-Achse das Attribut `lifeExp` aufgetragen ist.

- `px.pie(`*dataframename*, *weitereEinstellungen*)
  - Erzeugt ein Kuchendiagramm mit dem Dataframe *dataframename* und den Einstellungen *weitereEinstellungen*.
  - Als weitere Einstellungen können zum Beispiel `values =` *attributname* und `names =` *attributname* verwendet werden.
  - Beispiel:
    `px.pie(data2007, values = "pop", names = "continent")`
  - Erzeugt ein Kuchendiagramm mit den Daten `data2007`, bei denen die Werte des Kuchendiagramms vom Attribut `pop` und die Gruppierungen vom Attribut `continent` stammen.

- `import plotly.graph_objects as go`
  `go.Figure(`*diagramm*`1.data +` *diagramm*`2.data)`
  - Kombiniert zwei Diagramme *diagramm*1 und *diagramm*2.
  - Beispiel:
    `fig = go.Figure(fig1.data + fig2.data)`
  - Kombiniert die Diagramme `fig1` und `fig2` zu einem Diagramm `fig`.

Für eine ausführliche Dokumentation siehe: `https://plotly.com/python/plotly-express/`

6 Sklearn

```
from sklearn import model_selection, linear_model, neighbors, tree, ensemble,
         cluster, preprocessing
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
```

- `model_selection.train_test_split(`*daten*, *zielmerkmal*, *optionaleParameter*)
  - Teilt die Daten *daten* mit Zielmerkmal *zielmerkmal* in eine Trainings- und eine Testmenge auf.
  - Optional kann mit den Parametern `testsize =` *anteil* die anteilige Größe der Testmenge (*anteil* zwischen 0 und 1) festgelegt werden sowie mit `random_state =` *wert* eine feste (reproduzierbare) Zerlegung gewählt werden.

  - Beispiel:
    `X_train, X_test, y_train, y_test =`
    `model_selection.train_test_split(X, y,test_size = 0.1, random_state = 45)`
  - Zerlegt die Daten `X` mit Zielmerkmal `y` in eine Trainingsmenge `X_train` und `y_train` sowie eine Testmenge `X_test` und `y_test`, wobei die Testmenge 10% der Daten ausmacht.

- *modellname*`.fit(`*trainingsdaten*, *trainingszielmerkmal*)
  - Trainiert das Modell *modellname* mit den Trainingsdaten *trainingsdaten* und dem Zielmerkmal *trainingszielmerkmal*.
  - Beispiel:
    `model.fit(X_train,Y_train)`
  - Trainiert das Modell `model` auf den Trainingsdaten `X_train` und `y_train`.

- *modellname*`.predict(`*testdaten*)
  - Sagt für das Modell *modellname* zu den Testdaten *testdaten* entsprechende Zielmerkmale vorher.
  - Beispiel:
    `model.predict(X_test)`
  - Sagt für das Modell `model` für die Testdaten `X_test` die Zielmerkmale vorher.

- `accuracy_score(`*testzielmerkmal*, *vorhergesagtesZielmerkmal*)
  - Berechnet die Accuracy zwischen dem Testzielmerkmal *testzielmerkmal* und dem jeweiligen vorhergesagten Zielmerkmal *vorhergesagtesZielmerkmal*.
  - Beispiel:
    `accuracy_score(y_test, y_pred)`
  - Berechnet die Accuracy der Vorhersagen `y_pred` verglichen mit den Testzielmerkmalen `y_test`.

- `preprocessing.minmax_scale(`*daten*)
  - Normalisiert die Daten *daten* auf den Bereich zwischen 0 und 1.
  - Beispiel:
    `preprocessing.minmax_scale(X)`
  - Normalisiert die Daten `X`.

- `TfidfVectorizer(`*parameter*)

  - Erzeugt einen TfidfVectorizer mit Parametern *parameter*, der Dokumente vorverarbeiten kann.
  - Beispiel:
    ```
    vectorizer = TfidfVectorizer(vocabulary=None, analyzer='word',
              token_pattern=r"(?u)\b[a-zA-Z]{2,}\b",  lowercase=True,
             stop_words='english' )
    ```
  - Erzeugt einen TfidfVectorizer `vectorizer`, der ein eigenes Wörterbuch erstellt, Wort-Tokens mit mindestens zwei Buchstaben und ohne Sonderzeichen erstellt, Texte in Kleinbuchstaben umwandelt und Stoppwörter entfernt.

- *vectorizername*.`fit_transform`(*corpusname*)
  - Transformiert ein Korpus *corpusname* entsprechend den Eigenschaften des Vectorizers *vectorizername*.
  - Beispiel:
    `X = vectorizer.fit_transform(documents)`
  - Transformiert das Korpus `documents` gemäß dem Vectorizer `vectorizer` und speichert es in `X`.

- *vectorizername*.`get_feature_names_out`()
  - Gibt das Wörterbuch des Vectorizers *vectorizername* zurück.
  - Beispiel:
    `terms = vectorizer.get_feature_names_out()`
  - Speichert das Wörterbuch des Vectorizers `vectorizer` in `terms`.

- *modellname* = *sklearnModel*()
  - Erzeugt ein Modell *modellname* vom Typ *sklearnModel*().
  - Beispiele für Modelltypen:
    * `linear_model.LinearRegression`(*modellparameter*) für lineare Regression
    * `linear_model.LogisticRegression`(*modellparameter*) für logistische Regression
    * `neighbors.KNeighborsClassifier`(*modellparameter*) für ein k-nächste-Nachbarn-Modell
    * `tree.DecisionTreeClassifier`(*modellparameter*) für einen Entscheidungsbaum
    * `ensemble.RandomForestClassifier`(*modellparameter*) für einen Random Forest

        * `cluster.KMeans`(*modellparameter*) für k-means-Clustering
    - Die jeweiligen Untermodule müssen vorher importiert werden.

- `tree.plot_tree`(*entscheidungsbaummodell*)
    - Plottet den Entscheidungsbaum *entscheidungsbaummodell* mit Entscheidungsregeln.
    - Beispiel:
      `tree.plot_tree(dt_classifier)`
    - Plottet den Entscheidungsbaum `dt_classifier`.

Generelles Schema:

1. Spezifizierung des Modells *modellname* mit Modelltyp *sklearnModel*, siehe oben
   $modellname = sklearnModel()$
2. Training des Modells *modellname*
   modellname`.fit`(*trainingsdaten*, *trainingszielmerkmal*)
3. Test des Modells *modellname*
   modellname`.predict`(*testdaten*)
4. Evaluation des Modells *modellname*
   $score = scorefunktion(testzielmerkmal, vorhergesagtesZielmerkmal)$ `% z. B. accuracy_score`

Für eine ausführliche Dokumentation siehe: `https://scikit-learn.org/stable/`

7 Keras

```
import keras
from keras import layers, models
```

- `keras.Sequential()`
    - Erzeugt ein leeres sequenzielles neuronales Netz.
    - Beispiel:
      `model = keras.Sequential()`
    - Erzeugt ein sequenzielles neuronales Netz `model`.

- *modellname*`.add`(*schicht*)
    - Fügt eine Schicht *schicht* zum neuronalen Netz *modellname* hinzu.
    - Mögliche Schichten sind z.B. `Input`, `Flatten`, `Dense`, `Conv2D` und `MaxPooling2D`.

  - Beispiel:
    ```
    model.add(layers.Conv2D(32, (3, 3), activation='relu',
                    input_shape=(32, 32, 3)))
    ```
  - Fügt eine `Conv2D`-Schicht zum neuronalen Netz `model` hinzu.

- *modellname*`.summary()`
  - Erzeugt eine Zusammenfassung des Modells *modellname*.
  - Beispiel:
    ```
    model.summary()
    ```
  - Erzeugt eine Zusammenfassung des Modells `model`.

- *modellname*`.compile(`*parameter*`)`
  - Kompiliert das Modell *modellname* mit den Parametern *parameter*. Mögliche Parameter sind der zu wählende Optimierer, die Verlustfunktion und die Metrik, nach der das Modell evaluiert werden soll.
  - Beispiel:
    ```
    model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    ```
  - Kompiliert das Modell `model` mit dem Optimierer `adam`, der Verlustfunktion `sparse_categorical_crossentropy` und der Metrikenliste, die nur aus `accuracy` besteht.

- *modellname*`.fit(`*parameter*`)`
  - Trainiert das neuronale Netz *modellname* mit den Parametern *parameter*. Mögliche Parameter sind die Daten, auf denen das Modell trainiert werden soll, die Validierungsdaten sowie die Anzahl der Epochen.
  - Beispiel:
    ```
    model.fit(x=x_train, y=y_train, epochs=8, validation_data=(x_val, y_val))
    ```
  - Trainiert das neuronale Netz `model` mit den Trainingsdaten `x_train` und `y_train` und den Validierungsdaten `x_val` und `y_val` für acht Epochen.

- *modellname*`.evaluate(`*testdaten*`)`
  - Gibt eine Evaluation des Trainingserfolgs vom neuronalen Netz *modellname* in Bezug auf die Testdaten *testdaten* aus.
  - Beispiel:
    ```
    model.evaluate(x_test, y_test)
    ```
  - Gibt eine Evaluation des Trainingserfolgs vom neuronalen Netz `model` in Bezug auf die Testdaten `x_test` und `y_test` aus.

- *modellname*`.predict(`*testdaten*)
  - Sagt für das Modell *modellname* zu den Testdaten *testdaten* entsprechende Zielmerkmale vorher.
  - Beispiel:
    `model.predict(X_test)`
  - Sagt für das Modell `model` für die Testdaten `X_test` die Zielmerkmale vorher.

Für eine ausführliche Dokumentation siehe: `https://www.tensorflow.org`

8 Pandas

```
import pandas as pd
```

- *dataframename*`.head()`
  - Gibt die ersten fünf Einträge des DataFrame *dataframename* aus (aus dem Modul `pandas`, wird unter anderem für `plotly.express` verwendet)
  - Optional kann in den Klammern eine von 5 abweichende Zahl eingegeben werden.
  - Beispiel:
    `data.head(10)`
  - Gibt die ersten zehn Einträge aus dem DataFrame `data` aus.
- *dataframename*`.query(`*bedingung*)
  - Gibt alle Einträge des DataFrame *dataframename* aus, auf die die Bedingung *bedingung* zutrifft.
  - Beispiel:
    `data.query("year >= 1990")`
  - Gibt alle Einträge aus dem DataFrame `data` aus, bei denen das Attribut `year` mindestens einen Wert von 1990 hat.
- *dataframename*`.info()`
  - Gibt eine kurze Zusammenfassung des DataFrame *dataframename* aus.
  - Beispiel:
    `titanic.info()`
  - Gibt die Anzahl und Namen der Spalten aus dem DataFrame `titanic` aus, sowie die dazugehörigen Datentypen und Anzahl der nicht-leeren Einträge.

- *datframename*[*merkmalsname*].`unique()`
  - Gibt eindeutige Werte des Merkmals *merkmalsname* des DataFrame *datframename* zurück.
  - Beispiel:
    `titanic['who'].unique()`
  - Gibt die eindeutigen Werte des Merkmals `who` des DataFrame `titanic` zurück, in diesem Fall `['man' 'woman' 'child']`
- *datframename*.`rename`(*parameter*)
  - Benennt Teile des DataFrame *datframename* wie in *parameter* angegeben um.
  - Beispiel:
    `X.rename(columns={'who': 'is_child'}))`
  - Benennt die Spalte `who` des DataFrame `X` in `is_child` um.
- `pd.get_dummies`(*datframename*, *parameter*)
  - Wandelt wie in *parameter* spezifiziert das DataFrame *datframename* per One-Hot-Methode um.
  - Beispiel:
    `pd.get_dummies(y, columns=['survived'], dtype=float)`
  - Wandelt die Spalte `survived` des DataFrame `y` nach der One-Hot-Methode um, mit neuem Datentyp `float`.
- `pd.DataFrame`(*datframename*).`plot`(*parameter*)
  - Plottet das DataFrame *datframename* mit Parametern *parameter*.
  - Beispiel:
    `fig = pd.DataFrame(history.history).plot(figsize=(8, 5))`
  - Plottet die Historie.

Für eine ausführliche Dokumentation siehe: `https://pandas.pydata.org/`

9 Sonstige

- *diagrammname*.`show()`
  - Zeigt ein Diagramm *diagrammname* an.

  - Beispiel:
    `fig.show()`
  - Zeigt das Diagramm `fig` an.

- *diagrammname*`.write_image(`$pfad$`)`
  - Speichert ein Diagramm *diagrammname* statisch unter dem Pfad $pfad$ statisch ab.
  - Beispiel:
    `fig.write_image("plot.png")`
  - Speichert das Diagramm `fig` statisch im selben Ordner unter dem Dateinamen `plot.png` ab.

- *diagrammname*`.write_html(`$pfad$`)`
  - Speichert ein Diagramm *diagrammname* interaktiv unter dem Pfad $pfad$ ab.
  - Beispiel:
    `fig.write_html("plot.html")`
  - Speichert das Diagramm `fig` interaktiv im selben Ordner unter dem Dateinamen `plot.html` ab.