PREFLIGHT-CHECK Übungsaufgaben

Vom leeren Ordner zum lauffähigen ML-Projekt — Schritt für Schritt

So funktionieren diese Übungen

Die folgenden Aufgaben bauen aufeinander auf. Du startest bei Null und hast am Ende ein funktionierendes Mini-Projekt mit sauberer Ordnerstruktur, eigenem Python-Environment, installiertem seaborn und einem kleinen Skript, das alles zusammenführt.

Jede Aufgabe hat eine Erfolgskontrolle — damit weißt du selbst, ob es geklappt hat. Arbeite die Aufgaben der Reihe nach durch. Wenn du bei einer Aufgabe nicht weiterkommst, schaue im Preflight-Check-Dokument im passenden Modul nach.

Zeitaufwand: ca. 45–60 Minuten für alle Aufgaben.

Block 1: Dateisystem & Ordnerstruktur

Bezug: Modul A (Dateisystem & Pfade) und Modul B (Projektordner & Struktur)

Aufgabe 1: Den sicheren Hafen finden

1. Öffne den Datei-Explorer (Windows) bzw. Finder (Mac).
2. Navigiere zu deinem Benutzerordner (z. B. C:\Users\deinname oder /Users/deinname).
3. Prüfe: Enthält der Pfad irgendwo das Wort „OneDrive“ oder „iCloud“? Notiere den vollständigen Pfad unten.
4. Falls du in einem Cloud-Ordner gelandet bist: Navigiere zum echten lokalen Benutzerordner.
5. Erstelle dort (falls noch nicht vorhanden) einen Ordner namens projekte.

✅ **Erfolgskontrolle:** Du hast einen Ordner projekte an einem lokalen Ort, der KEIN Cloud-Sync-Ordner ist.

Mein Pfad zum projekte-Ordner: **Cloud-Sync erkannt? (Ja/Nein + wo):**

Aufgabe 2: Projektstruktur anlegen

1. Erstelle innerhalb von projekte/ einen neuen Ordner: seaborn-uebung.
2. Erstelle innerhalb von seaborn-uebung/ vier Unterordner: data, notebooks, scripts, output.
3. Prüfe die Struktur: Kannst du im Explorer/Finder alle fünf Ordner sehen?

✅ **Erfolgskontrolle:** Dein Datei-Explorer zeigt: projekte/seaborn-uebung/ mit den Unterordnern data, notebooks, scripts und output.

Aufgabe 3: Pfade verstehen

1. Lade die Datei iris.csv herunter (dein Dozent stellt sie bereit, oder suche „iris dataset csv“).
2. Finde die heruntergeladene Datei in deinem Downloads-Ordner.
3. Verschiebe (nicht kopieren!) die Datei von Downloads nach seaborn-uebung/data/.
4. Notiere den vollständigen (absoluten) Pfad zur Datei.
5. Notiere den relativen Pfad von seaborn-uebung/ zur Datei.

✅ **Erfolgskontrolle:** iris.csv liegt in seaborn-uebung/data/ und ist nicht mehr in Downloads.

Absoluter Pfad zu iris.csv: **Relativer Pfad (von seaborn-uebung/ aus):**

Block 2: VS Code & Python-Environment

Bezug: Modul C (Datei-Explorer & VS Code) und Modul D (Python-Environments)

Aufgabe 4: Projektordner in VS Code öffnen

1. Navigiere im Datei-Explorer / Finder zu deinem Ordner seaborn-uebung.
2. Rechtsklick auf den Ordner → „Mit Code öffnen“ (Windows) bzw. „Open with Code“ (Mac).
3. Prüfe in VS Code: Ist in der Seitenleiste links dein Projektordner mit allen Unterordnern sichtbar?
4. Prüfe die Titelleiste oben: Steht dort „seaborn-uebung“?

✅ **Erfolgskontrolle:** VS Code zeigt in der Seitenleiste: data/, notebooks/, scripts/, output/ — und die Titelleiste zeigt den Ordnernamen.

> ⚠️ **Achtung:** Falls du nur eine einzelne Datei geöffnet hast (kein Ordner in der Seitenleiste, nur ein Dateiname in der Titelleiste), schließe VS Code und öffne den Ordner erneut über den Datei-Explorer.

Aufgabe 5: Python-Environment erstellen

1. Stelle sicher, dass die Extension „Python Environments“ in VS Code installiert ist.
2. Öffne die Command Palette: Ctrl+Shift+P (Windows) bzw. Cmd+Shift+P (Mac).
3. Tippe „Create Environment“ und wähle den Eintrag aus.
4. Wähle „Venv“ als Typ und dann deine Python-Version (z. B. Python 3.11).
5. Warte, bis VS Code das Environment erstellt hat.
6. Prüfe die Statusleiste unten links: Dort sollte jetzt .venv oder der Environment-Name stehen.

✅ **Erfolgskontrolle:** In der VS Code-Statusleiste (unten links) steht der Name des Environments. In der Seitenleiste siehst du einen neuen Ordner .venv.

Aufgabe 6: seaborn installieren und testen

1. Öffne in VS Code ein Terminal: Menü → Terminal → Neues Terminal.
2. Prüfe: Steht am Anfang der Zeile (.venv)? Falls ja, ist dein Environment automatisch aktiv.
3. Installiere seaborn: pip install seaborn
4. Installiere zusätzlich: pip install pandas jupyter
5. Teste die Installation: Tippe python und dann import seaborn; print(seaborn.__version__)
6. Beende Python wieder mit exit().

✅ **Erfolgskontrolle:** Der Befehl import seaborn gibt keine Fehlermeldung, und du siehst eine Versionsnummer (z. B. 0.13.2).

```
# Im VS Code Terminal (Environment muss aktiv sein!):
pip install seaborn pandas jupyter
# Schnelltest:
python
>>> import seaborn
>>> print(seaborn.__version__)
>>> exit()
```

Block 3: Speicherplatz prüfen

Bezug: Modul E (Speicherplatz & Systemhygiene)

Aufgabe 7: Speicherplatz-Audit

1. Prüfe, wie viel freier Speicherplatz auf deiner Festplatte ist. (Windows: Explorer → Dieser PC → Rechtsklick auf C: → Eigenschaften. Mac: Apple-Menü → Über diesen Mac → Festplatten.)

2. Notiere den freien Speicherplatz.

3. Prüfe die Größe deines .venv-Ordners: Rechtsklick auf den Ordner → Eigenschaften (Windows) bzw. Informationen (Mac).

4. Frage an dich selbst: Wenn dieses Projekt fertig ist und du den .venv-Ordner löschst — wie viel Platz gewinnst du zurück?

✅ **Erfolgskontrolle:** Du kennst deinen freien Speicherplatz und die Größe deines Environments.

Freier Speicherplatz: **Größe des .venv-Ordners:**

Block 4: Dateien orchestrieren

Bezug: Modul F (Dateien orchestrieren) — Hier kommt alles zusammen.

Aufgabe 8: Analyse-Skript schreiben

1. Erstelle in VS Code eine neue Datei: Rechtsklick auf den Ordner scripts/ → Neue Datei → analyse.py.

2. Schreibe folgendes Skript (oder tippe es ab — nicht kopieren, das Üben zählt!):

3. Das Skript soll: (a) pandas importieren, (b) die Datei data/iris.csv laden, (c) die ersten 5 Zeilen ausgeben, (d) die Spalte „species“ zählen (.value_counts()).

4. Speichere die Datei.

5. Führe das Skript aus: Im VS Code-Terminal: python scripts/analyse.py.

✅ **Erfolgskontrolle:** Das Skript gibt die ersten 5 Zeilen und die Artenverteilung aus — ohne Fehlermeldung.

```
# scripts/analyse.py
import pandas as pd
# Daten laden (relativer Pfad!)
df = pd.read_csv('data/iris.csv')
# Erste 5 Zeilen
print('=== Erste 5 Zeilen ===')
print(df.head())
print()
# Artenverteilung
print('=== Artenverteilung ===')
print(df['species'].value_counts())
```

⚠️ **Typischer Fehler:** FileNotFoundError: 'data/iris.csv' — Das bedeutet, dass das Terminal nicht im Projektordner steht. Lösung: Stelle sicher, dass du den Projektordner in VS Code geöffnet hast (nicht eine einzelne Datei) und das Terminal dort startet.

Aufgabe 9: Visualisierung mit seaborn

1. Erstelle eine neue Datei: scripts/visualisierung.py.

2. Schreibe ein Skript, das: (a) seaborn und pandas importiert, (b) iris.csv lädt, (c) einen seaborn-Plot erstellt (z. B. sns.pairplot), (d) den Plot als Bilddatei in output/ speichert.

3. Führe das Skript aus: python scripts/visualisierung.py.

4. Prüfe: Liegt die Bilddatei jetzt in output/?

✅ **Erfolgskontrolle:** Im Ordner output/ liegt eine Datei iris_pairplot.png, die du im Explorer/Finder öffnen und anschauen kannst.

```
# scripts/visualisierung.py
import pandas as pd
import seaborn as sns
import matplotlib
matplotlib.use('Agg')  # Kein Fenster, nur Datei
import matplotlib.pyplot as plt

# Daten laden
df = pd.read_csv('data/iris.csv')

# Plot erstellen
plot = sns.pairplot(df, hue='species')

# In output/ speichern
plot.savefig('output/iris_pairplot.png', dpi=150)
print('Plot gespeichert: output/iris_pairplot.png')
```

Aufgabe 10: Den roten Faden zeichnen

1. Zeichne auf Papier (oder hier unten) den Datenfluss deines Mini-Projekts:

2. Welche Datei ist die Eingabe (Datenquelle)?

3. Welche Skripte verarbeiten die Daten?

4. Welche Dateien werden erzeugt (Ausgaben)?

5. Zeichne Pfeile zwischen den Komponenten und schreibe die Ordner dazu.

✅ **Erfolgskontrolle:** Du hast eine Übersicht, die zeigt: data/iris.csv → scripts/analyse.py → Terminal-Ausgabe, und data/iris.csv → scripts/visualisierung.py → output/iris_pairplot.png.

Mein Datenfluss-Diagramm:

Bonus: Transferaufgabe

Diese Aufgabe ist für alle, die schneller fertig sind. Sie verknüpft die Übung mit dem, was im Kurs als Nächstes kommt: sklearn-Pipelines und Gradio-Deployment.

Aufgabe 11: Mini-ML-Pipeline (für Schnelle)

1. Erstelle ein neues Skript: scripts/train.py.

2. Schreibe ein Skript, das: (a) iris.csv lädt, (b) ein einfaches sklearn-Modell trainiert (z. B. DecisionTreeClassifier), (c) das trainierte Modell als models/iris_model.pkl speichert (mit joblib).

3. Erstelle den Ordner models/ falls noch nicht vorhanden.

4. Führe das Skript aus und prüfe, ob models/iris_model.pkl existiert.

5. Bonusfrage: Welches Skript würde als Nächstes kommen, um das Modell zu nutzen? (Tipp: Gradio!)

✅ **Erfolgskontrolle:** In models/ liegt iris_model.pkl. Du kannst erklären, welches Skript das Modell als Nächstes laden würde.

```
# scripts/train.py
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import joblib

# Daten laden
df = pd.read_csv('data/iris.csv')
X = df.drop('species', axis=1)
y = df['species']

# Modell trainieren
model = DecisionTreeClassifier()
model.fit(X, y)
print(f'Accuracy: {model.score(X, y):.2%}')

# Modell speichern
joblib.dump(model, 'models/iris_model.pkl')
print('Modell gespeichert: models/iris_model.pkl')
```

Blick nach vorn

Der nächste Schritt im Kurs wäre ein scripts/app.py, das:

1. Das Modell aus models/iris_model.pkl lädt
2. Eine Gradio-Oberfläche erstellt
3. Nutzereingaben entgegennimmt und Vorhersagen liefert

Damit schließt sich der Kreis: data/ → scripts/train.py → models/ → scripts/app.py → Gradio-Interface. Genau dieser Datenfluss ist das, was wir im Kurs aufbauen.

Abschluss-Checkliste

Wenn du alle Aufgaben durchgearbeitet hast, solltest du jeden dieser Punkte mit „Ja“ beantworten können:

✓	Das kann ich jetzt
☐	Ich habe einen lokalen projekte-Ordner, der nicht in einem Cloud-Sync liegt.
☐	Mein Projekt seaborn-uebung hat eine saubere Ordnerstruktur (data/, notebooks/, scripts/, output/).
☐	Ich öffne den Projektordner in VS Code (nicht einzelne Dateien).
☐	Mein Python-Environment (.venv) ist im Projekt erstellt und in VS Code aktiv.
☐	seaborn, pandas und matplotlib sind im Environment installiert.
☐	Ich kann ein Skript aus scripts/ ausführen, das Daten aus data/ liest.
☐	Ich kann eine Ausgabedatei in output/ erzeugen.
☐	Ich kann den Datenfluss meines Projekts erklären.
☐	Ich kenne meinen freien Speicherplatz und die Größe meines Environments.

Alle Häkchen gesetzt? Dann bist du bereit für den nächsten Kursabschnitt. Deine Arbeitsumgebung steht, dein Workflow sitzt, und du weißt, wo jede Datei hingeht. Das ist kein Vorwissen — das ist eine Kompetenz, die du dir gerade erarbeitet hast.