🤒 Health & Lifestyle – *Reproduzierbarer Bericht*

Dieser Report ist so geschrieben, dass **jede:r im Team** die Aufgabe **selbstständig lösen** und **nachvollziehen** kann.

---

1) Datensatz & Ziel

- Datei: `health_lifestyle_dataset.csv`
- Aufgabe: **Binäre Klassifikation** der Zielvariable `disease_risk` (0 = gesund, 1 = Risiko)
- Verwendete Features: `bmi`, `daily_steps`, `sleep_hours`, `calories_consumed`, `cholesterol`, `systolic_bp`, `diastolic_bp`, `family_history`

Dataset Shape: 100000 × 16
Zielverteilung (`disease_risk`):

- 0 (gesund): 75179 (0.7518)
- 1 (Risiko): 24821 (0.2482)

---

2) Vorgehen (Schritt-für-Schritt)

a) Daten laden & Features definieren (Python-Statements)

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("health_lifestyle_dataset.csv")

X = df[[
    "bmi", "daily_steps", "sleep_hours", "calories_consumed",
    "cholesterol", "systolic_bp", "diastolic_bp", "family_history"
]]
y = df["disease_risk"]
```

Wichtige Schritte und was sie tun:

b) Modelle

Wir vergleichen **6 Modelle**:

c) Splits & Messung

Für jedes Modell messen wir die **Accuracy** bei **Testanteilen 10 %, 30 %, 50 %, 70 %, 90 %**.
Beim **KNN** wird zusätzlich **k ∈ {1,3,5,7,9,11,13,15,17,19,21}** ausprobiert und je Split das **beste k** gewählt.

d) Ergebnisse & Darstellung (Python-Statements)

- train_test_split(X, y, test_size=t, stratify=y, random_state=42): stratifizierter Split für faire Klassenverteilung.

---

3) Ergebnisse (exakt aus euren CSVs)

3.1 Accuracy je Modell und Testanteil

Modell	10 %	30 %	50 %	70 %	90 %
Decision Tree	0.617	0.616	0.618	0.614	0.618
KNN (best k=21)	0.749	0.748	0.749	0.749	0.749
Logistic Regression	0.752	0.752	0.752	0.752	0.752
Naive Bayes	0.752	0.752	0.752	0.752	0.752
Random Forest	0.752	0.751	0.751	0.751	0.751
SVM	0.752	0.752	0.752	0.752	0.752

👉 Grundlage: `model_accuracy_over_splits_knn_bestk.csv` (KNN bereits mit *best k* je Split).

3.2 Gewinner je Testanteil

- **10 % Test** → **Logistic Regression** mit **0.752**
- **30 % Test** → **Logistic Regression** mit **0.752**
- **50 % Test** → **Logistic Regression** mit **0.752**
- **70 % Test** → **Logistic Regression** mit **0.752**
- **90 % Test** → **Logistic Regression** mit **0.752**

3.3 KNN – bestes k je Split

Testanteil	bestes k	Accuracy
10 %	21	0.749
30 %	21	0.748
50 %	21	0.749
70 %	21	0.749
90 %	21	0.749

> Ergänzende Detaildateien:
>
> - `lr_accuracy_by_split.csv` (Logistic Regression – Accuracy je Testanteil)
> - `knn_accuracy_grid_over_k_and_splits.csv` (komplette k×Split-Matrix)
> - `knn_bestk_per_split.csv` (bestes k je Testanteil)

---

4) Visualisierung

4.1 Warum gibt es zwei sehr ähnliche Diagramme je Modell?

In den Notebooks gibt es pro Modell bewusst zwei Plot-Varianten:

- 10/30/50/70/90 % – Tabelle + Plot: schnelle Live-Ausgabe direkt nach der Messung, inkl. print()/display der Roh-Tabelle. Praktisch für Exploration und unmittelbares Feedback.
- Übersichtstabelle & Plot (Accuracy je Testanteil): standardisierte Darstellung mit konsistenten Achsen/Labels/xticks und oft zusätzlichem CSV-Export. Dient der Vergleichbarkeit über alle Modelle und der Abgabe.

Kurz: Der erste Plot ist der schnelle Check, der zweite ist die „saubere" Version für Bericht/Export. Inhalte sind ähnlich, aber die Formatierung ist vereinheitlicht und die Daten werden zusätzlich gespeichert.

---

5) Interpretation

- Spitze: In unseren Ergebnissen liegen Logistische Regression, SVM und Naive Bayes praktisch gleichauf und bilden die besten Accuracies über die Splits.
- Random Forest liegt knapp dahinter, bleibt aber robust über unterschiedliche Testanteile.
- Decision Tree fällt deutlich ab und ist variabler (höhere Varianz/Overfitting-Tendenz bei weniger Trainingsdaten).
- KNN liegt unterhalb der Top-Modelle; die Performance hängt spürbar von k ab. Mittlere k liefern meist den besten Kompromiss; sehr kleine k neigen zu hoher Varianz.

---

6) Reproduzierbarer Beispielcode (Template)

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

df = pd.read_csv("health_lifestyle_dataset.csv")
X = df[[
    "bmi","daily_steps","sleep_hours","calories_consumed",
    "cholesterol","systolic_bp","diastolic_bp","family_history"
]]
y = df["disease_risk"]

test_sizes = [0.1, 0.3, 0.5, 0.7, 0.9]

models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
    "Naive Bayes": GaussianNB(),
    "SVM": SVC(),
```

```
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
}

rows = []
k_list = [1,3,5,7,9,11,13,15,17,19,21]
for t in test_sizes:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=t,
random_state=42, stratify=y)
    # Basismodelle
    for name, mdl in models.items():
        mdl.fit(X_train, y_train)
        acc = mdl.score(X_test, y_test)
        rows.append({"Model": name, "Testanteil": t, "Accuracy": acc})
    # KNN best-k
    best_k, best_acc = None, -1
    for k in k_list:
        knn = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
        acc = knn.score(X_test, y_test)
        if acc > best_acc:
            best_k, best_acc = k, acc
    rows.append({"Model": f"KNN (best k={best_k})", "Testanteil": t, "Accuracy":
best_acc})

pd.DataFrame(rows).to_csv("model_accuracy_over_splits_knn_bestk.csv", index=False)
```

6.1) CSV-Exporte – Inhalt und Zweck

Nur die Dateien, die direkt im Ordner `OutputCSV` liegen (ohne Archiv):

- model_accuracy_over_splits_knn_bestk.csv
  - Zweck: Gesamter Vergleich aller Modelle über die fünf Testanteile; KNN ist pro Split bereits mit dem jeweils besten k enthalten.
  - Spalten (Header): Modell, Testanteil, Trainanteil, Accuracy
- knn_accuracy_grid_over_k_and_splits.csv
  - Zweck: Langformat-Matrix über alle Kombinationen (Testanteil × k) für KNN; Basis für Pivot/Heatmap und Linienplot Accuracy vs. k.
  - Spalten (Header): Testanteil, k, Accuracy
- knn_bestk_per_split.csv
  - Zweck: Kurzüberblick über das beste k je Testanteil inklusive zugehöriger Accuracy.
  - Spalten (Header): Testanteil, Bestes k, Accuracy
- lr_accuracy_by_split.csv
  - Zweck: Logistic-Regression – Accuracy über die fünf Testanteile (einzelnes Modell).
  - Spalten (Header): Testanteil, Trainanteil, Accuracy

Hinweis: Weitere, gleichartige Dateien pro Modell (z. B. für NB/SVM/DT/RF) wurden in den Archiv-Unterordner verschoben, um die Abgabe übersichtlich zu halten. Bei Bedarf können sie jederzeit wiederhergestellt werden.
