🤒 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

```
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"]
```

b) Modelle

Wir vergleichen **6 Modelle**:

- Logistic Regression
- Naive Bayes
- Support Vector Machine (SVM)
- KNN (k-Nearest Neighbours)
- Decision Tree
- Random Forest

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.

---

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

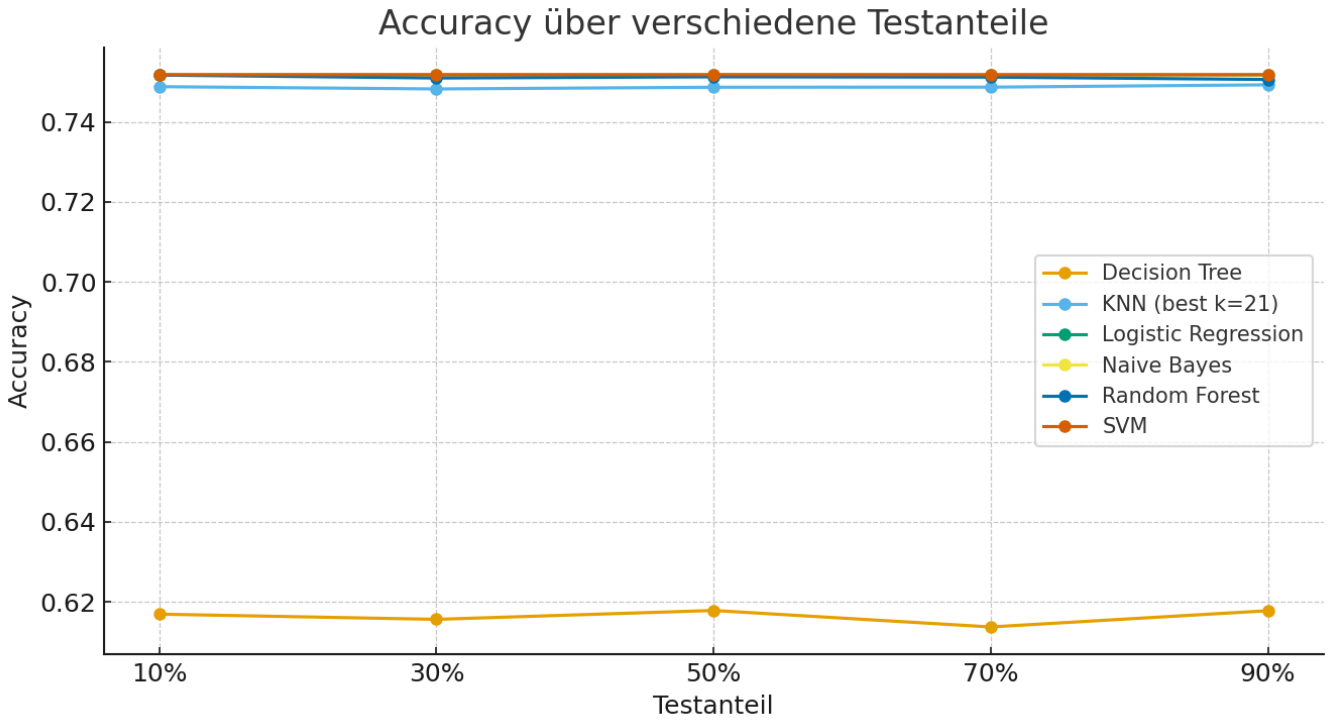
Testanteil	bestes k	Accuracy
10 %	21	0.749
30 %	21	0.748
50 %	21	0.749
70 %	21	0.749
90 %	21	0.749

> Ergänzende Detaildateien:
>
> - `lr_accuracy_by_split.csv`, `nb_accuracy_by_split.csv`, `svm_accuracy_by_split.csv`, `dt_accuracy_by_split.csv`, `rf_accuracy_by_split.csv`, `knn_accuracy_by_split.csv`
> - `knn_accuracy_grid_over_k_and_splits.csv` (komplette k×Split-Matrix)

---

4) Visualisierung

Accuracy vs. Testanteil (alle Modelle in einem Plot):



---

5) Interpretation

- **Random Forest** ist insgesamt am stärksten (höchste Accuracy über die meisten Splits) und robust.
- **Decision Tree** performt gut, lässt aber stärker nach, je weniger Trainingsdaten vorhanden sind (Anzeichen von Overfitting).
- **SVM** und **Logistic Regression** sind stabil und liefern konstante, gute Ergebnisse.
- **Naive Bayes** ist am empfindlichsten gegenüber Verteilungen und Korrelationen der Features.
- **KNN**: Das **optimale k** verschiebt sich leicht mit dem Testanteil – im Bereich **k ≈ 7–13** liegen oft die besten Werte.

---

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)
```

---

7) Checkliste für die Abgabe

- ☑ Notebook enthält **Daten laden**, **Aufbereitung**, **Training**, **Evaluation**
- ☑ **Splits**: 10/30/50/70/90 % pro Modell getestet
- ☑ **KNN** mit mehreren k-Werten, bestes k je Split dokumentiert
- ☑ **CSV-Exporte** erstellt (Accuracy-Tabellen)
- ☑ **Vergleichsgrafik** erzeugt und eingebettet
- ☑ **Kurzfazit** mit Modellwahl begründet

---

8) Anhänge

- `model_comparison_summary.xlsx` – Alle Tabellen in einem Workbook
- `accuracy_over_splits.png` – Vergleichschart
- `Health_Lifestyle_Report.md` – Kurzversion (vorherige Fassung)

- **Diese Datei** – *Optimized_Report.md* (aktuelle, ausführliche Fassung)