LE1 PYTHON CHEAT SHEET (Lineare & Logistische Regression) – 1 Seite A4

```
--- CSV & BASIS ---
import pandas as pd
import numpy as np

df = pd.read_csv(PATH)              # CSV einlesen
df.info(); df.head()                # Überblick
df.describe(numeric_only=True)      # Statistiken
df.isna().sum()                     # Missing check
df = df.dropna()                    # (einfach) fehlende Zeilen entfernen

# Ziel/Features benennen (je nach Prüfung)
# Autos:   y = 'distance', X = 'geschwindigkeit'   (Lineare Regression)
# KK:      y = 'owns_card' (0/1), X = 'age'        (Logistische Regression)
# Titanic: y = 'Survived' (0/1), X = 'Fare' oder 'Sex' (binär)

--- PLOTS (matplotlib) ---
import matplotlib.pyplot as plt
# Scatter (kontinuierliche X)
plt.figure(); plt.scatter(df[X], df[y], alpha=0.6); plt.xlabel(X); plt.ylabel(y)

# Barplot für binäre/ kategoriale X
df.groupby(X)[y].mean().plot(kind="bar"); plt.ylabel("mean of " + y)

--- LINEARE REGRESSION (statsmodels) ---
import statsmodels.api as sm
X = sm.add_constant(df[[feature]])     # feature = 'geschwindigkeit' etc.
y = df[y]
model_lin = sm.OLS(y, X).fit()
print(model_lin.summary())
# Koeffizienten
b0, b1 = model_lin.params['const'], model_lin.params[feature]
# Vorhersage & R^2
y_pred = model_lin.predict(X)
R2 = model_lin.rsquared
# Residuen
res = y - y_pred
# Residuenplots
plt.figure(); plt.scatter(df[feature], res); plt.axhline(0, ls="--"); plt.xlabel(feature); plt.ylabel("residuals")
plt.figure(); plt.hist(res, bins=20)

--- LOGISTISCHE REGRESSION (statsmodels) ---
# Für binäres Ziel y in {0,1}
X = sm.add_constant(df[[feature]])     # feature = 'age', 'Fare', oder binäres Dummy
y = df[y].astype(int)
model_log = sm.Logit(y, X).fit(disp=False)
print(model_log.summary())
# Logit-Gleichung: log(p/(1-p)) = b0 + b1*X
b0, b1 = model_log.params['const'], model_log.params[feature]
# Wahrscheinlichkeit p(x) = 1 / (1 + exp(-(b0 + b1*x)))
p = model_log.predict(X)

--- KLASSIFIKATION, KONFUSIONSMATRIX, ACCURACY (sklearn) ---
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
threshold = 0.5
y_hat = (p >= threshold).astype(int)
cm = confusion_matrix(y, y_hat)       # [[TN, FP],[FN, TP]]
acc = accuracy_score(y, y_hat)
print(cm, "Accuracy=", acc)

# Als schön beschriftete Tabelle
tn, fp, fn, tp = cm.ravel()
print(f"TP={tp}, FP={fp}, FN={fn}, TN={tn}")

--- SPEZIFISCHES ---
# Titanic: Geschlecht in 0/1 codieren
df['Sex_bin'] = (df['Sex'].str.lower().map({'male':0,'female':1})).astype(int)
# 80-jährige Wahrscheinlichkeit (KK-Beispiel):
import numpy as np
x80 = sm.add_constant(pd.DataFrame({feature:[80]}))
p80 = float(model_log.predict(x80))
print("P(80) =", round(100*p80,2), "%")

# Logistische Kurve + Punkte plotten (gegen x)
xgrid = np.linspace(df[feature].min(), df[feature].max(), 200)
Xg = sm.add_constant(pd.DataFrame({feature:xgrid}))
pg = model_log.predict(Xg)
plt.figure(); plt.scatter(df[feature], y, alpha=0.3)
plt.plot(xgrid, pg, linestyle="--")
plt.xlabel(feature); plt.ylabel("Probability of " + str(y.name))

--- TIPPS & FALLSTRICKE ---
# • Immer add_constant() benutzen (Intercept).
# • Für binäre Variablen vorher sauber 0/1 kodieren.
# • Kein Train/Test in diesen Prüfungen (explizit untersagt).
# • Achsen & Legenden korrekt beschriften.
# • Für lineare Modelle: R^2, Residuen (≈ Normal, Erwartungswert 0, keine U-Form).
# • Für logistische Modelle: Konfusionsmatrix & Accuracy; Schwelle 0.5 dokumentieren.
# • Numerische Stabilität: bei extremen X-Werten kann p ≈ 0/1 werden – ok.
# • Formeln zum Aufschreiben:
#   Linear:   ŷ = b0 + b1 x,   R^2 = 1 - SSE/SST
#   Logit:    log(p/(1-p)) = b0 + b1 x,   p = 1/(1+exp(-(b0+b1x)))
```