🧠 Was passiert im Code?

1️⃣ Import und Datenaufbereitung

```
df = pd.read_csv('CaliforniaHousingCut_extended.csv')
```

- Der Datensatz enthält reale Housing-Daten mit künstlich hinzugefügten Spalten: `Postal_Code`, `Extra_Noise`, `Random`, `Average_NotBedrooms`.
- Zielvariable (`y`) ist `Price`.

2️⃣ Modelle mit unterschiedlichen Variablen

Du trainierst **acht verschiedene lineare Regressionsmodelle**, um den Einfluss der Eingangsvariablen auf das **Erklärungsmaß R^2** zu vergleichen.

📘 Modell 1 – *Basis-Modell*

```
X_base = [‘Median Income’, ‘Age of House’, ‘Average Rooms’, ‘Average Bedrooms’, ‘Population’, ‘Average Occupancy’, ‘Latitude’, ‘Longitude’]
```

- Enthält **alle sinnvollen erklärenden Variablen**.
- Erwartung: **höchstes R^2**, da das Modell umfassend und informativ ist.

📘 Modell 2 – *Nur 3 wichtige Variablen*

```
["Median Income", "Average Rooms", "Population"]
```

- Nur stark korrelierte Kernvariablen.
- Erwartung: **etwas niedrigeres R^2**, aber noch relativ gut, da Einkommen & Räume zentrale Einflussgrößen sind.

📘 Modell 3 – *Andere 3 Variablen*

```
["Median Income", "Age of House", "Average Occupancy"]
```

- Variablen, die *weniger direkt* mit Preis zusammenhängen.
- Erwartung: **deutlich geringeres R^2** – das Modell erklärt weniger Varianz.

📘 Modell 4 – *Basis + Postal Code*

```
X_base + ['Postal_Code']
```

- Hinzufügen einer **irrelevanten** Variablen (Postleitzahl = zufällig verteilt).
- Erwartung: **R^2 steigt minimal** (wegen Overfitting), aber der Zugewinn ist **nicht sinnvoll**.

📘 Modell 5 – *Basis + Extra Noise*

`X_base + ['Extra_Noise']`

- Zufällige Normalverteilung, keine echte Korrelation mit `Price`.
- Erwartung: **kein echter Zugewinn**, evtl. minimale Schwankung im R^2.

📘 Modell 6 – *Basis + Random*

`X_base + ['Random']`

- Gleichverteilte Zufallsvariable, ebenfalls ohne Zusammenhang.
- Erwartung: ähnlich wie bei Extra_Noise, **kein echter Effekt auf R^2**.

📘 Modell 7 – *Basis + Average_NotBedrooms*

`X_base + ['Average_NotBedrooms']`

- Abgeleitete Variable, evtl. **leicht korreliert** mit „Average Rooms" oder „Average Bedrooms".
- Erwartung: **möglicher leichter Zugewinn** im R^2, da zusätzliche strukturelle Information vorhanden sein kann.

📘 Modell 8 – *Nur 3 Variablen + Postal Code*

`["Median Income", "Age of House", "Average Occupancy"] + Postal_Code`

- Ein kleines Modell mit irrelevanter Zusatzvariable.
- Erwartung: R^2 **bleibt gleich oder steigt minimal** – wieder Overfitting ohne echten Erkenntnisgewinn.

📊 Erwartete Ergebnisse (typische Tendenz)

Modell	Beschreibung	Erwartetes R^2	Erklärung
1	Basis-Modell	**≈ 0.70–0.80**	Gute Erklärungskraft
2	3 Kernvariablen	**≈ 0.60–0.75**	Weniger umfassend, aber noch solide
3	Andere 3 Variablen	**≈ 0.40–0.60**	Schwächere Korrelationen
4	+ Postal Code	**minimal höher** als 1	Nur Overfitting
5	+ Extra Noise	≈ Modell 1	Keine relevante Verbesserung
6	+ Random	≈ Modell 1	Keine relevante Verbesserung
7	+ Average_NotBedrooms	leicht ↑ (0.01–0.03)	evtl. nützliche Zusatzinfo
8	3 Var + Postal Code	minimal ↑ ggü. Modell 3	Overfitting, kein echter Zugewinn

---

💬 Interpretation für Studierende

- **Mehr Variablen ≠ besseres Modell.** R^2 kann leicht steigen, aber das bedeutet nicht, dass das Modell *bessere Vorhersagen* macht.
- **Irrelevante Variablen** (Postal Code, Random, Noise) führen zu Overfitting.
- **Adjusted R^2** wäre das bessere Maß, weil es solche Scheinverbesserungen bestraft.
- **Weniger ist oft mehr:** Ein Modell mit weniger, aber aussagekräftigen Variablen ist stabiler und besser interpretierbar.

In [1]:
```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [2]:
```
# Originaldaten laden (bereits erweitert)
df = pd.read_csv('CaliforniaHousingCut_extended.csv')
```

In [3]:
```
# Basisvariablen
y = df['Price']

X_base = df[[
    'Median Income',
    'Age of House',
    'Average Rooms',
    'Average Bedrooms',
    'Population',
    'Average Occupancy',
    'Latitude',
    'Longitude'
]]
```

In [4]:
```
# Modell 1: Basis Model
X1_train, X1_test, y_train, y_test = train_test_split(X_base, y, test_size=0.2,
model1 = LinearRegression().fit(X1_train, y_train)

y_pred1 = model1.predict(X1_test)
r2_1 = r2_score(y_test, y_pred1)
```

In [5]:
```
#  Modell 2: Modell mit nur 3 Variablen
X_2 = df[["Median Income", "Average Rooms", "Population"]]

X2_train, X2_test, y_train, y_test = train_test_split(X_2, y, test_size=0.2, ran
model2 = LinearRegression().fit(X2_train, y_train)

y_pred2 = model2.predict(X2_test)
r2_2 = r2_score(y_test, y_pred2)
```

In [6]:
```
#  Modell 3: Modell mit nur 3 Variablen, aber anderen
X_3 = df[["Median Income", "Age of House", "Average Occupancy"]]

X3_train, X3_test, y_train, y_test = train_test_split(X_3, y, test_size=0.2, ran
model3 = LinearRegression().fit(X3_train, y_train)
```

```
y_pred3 = model3.predict(X3_test)
r2_3 = r2_score(y_test, y_pred3)
```

In [7]:
```
# Modell 4: Basis Model mit Postal Code
X_4 = X_base.join(df["Postal_Code"])

X4_train, X4_test, y_train, y_test = train_test_split(X_4, y, test_size=0.2, ran
model4 = LinearRegression().fit(X4_train, y_train)
y_pred4 = model4.predict(X4_test)
r2_4 = r2_score(y_test, y_pred4)
```

In [8]:
```
# Modell 5: Basis Model mit Extra Noise
X_5 = X_base.join(df["Extra_Noise"])

X5_train, X5_test, y_train, y_test = train_test_split(X_5, y, test_size=0.2, ran
model5 = LinearRegression().fit(X5_train, y_train)
y_pred5 = model5.predict(X5_test)
r2_5 = r2_score(y_test, y_pred5)
```

In [9]:
```
# Modell 6: Basis Model mit Random
X_6 = X_base.join(df["Random"])

X6_train, X6_test, y_train, y_test = train_test_split(X_6, y, test_size=0.2, ran
model6 = LinearRegression().fit(X6_train, y_train)
y_pred6 = model6.predict(X6_test)
r2_6 = r2_score(y_test, y_pred6)
```

In [10]:
```
# Modell 7: Basis Model mit Random
X_7 = X_base.join(df["Average_NotBedrooms"])

X7_train, X7_test, y_train, y_test = train_test_split(X_7, y, test_size=0.2, ran
model7 = LinearRegression().fit(X7_train, y_train)
y_pred7 = model7.predict(X7_test)
r2_7 = r2_score(y_test, y_pred7)
```

In [11]:
```
# Modell 8: Basis Model mit Random
X_8 = df[[
    'Median Income',
    'Age of House',
    'Average Occupancy',
]]
X_8.join(df["Postal_Code"])

X8_train, X8_test, y_train, y_test = train_test_split(X_8, y, test_size=0.2, ran
model8 = LinearRegression().fit(X8_train, y_train)
y_pred8 = model8.predict(X8_test)
r2_8 = r2_score(y_test, y_pred8)
```

In [12]:
```
# Ergebnisse ausgeben
print(f"R² (Modell 1: Basis):                    {r2_1:.6f}")
print(f"R² (Modell 2: 3 Var):                    {r2_2:.6f}")
print(f"R² (Modell 3: 3 Var):                    {r2_3:.6f}")
print(f"R² (Modell 4: Basis + Postal Code):      {r2_4:.6f}")
print(f"R² (Modell 5: Basis + Extra Noise):      {r2_5:.6f}")
print(f"R² (Modell 6: Basis + Random):           {r2_6:.6f}")
print(f"R² (Modell 7: Basis + Avg. NotBedrooms): {r2_7:.6f}")
print(f"R² (Modell 8: 3 Var + Postal Code):      {r2_8:.6f}")
```

```
R² (Modell 1: Basis):                    0.545194
R² (Modell 2: 3 Var):                    0.368594
R² (Modell 3: 3 Var):                    0.374251
R² (Modell 4: Basis + Postal Code):      0.545223
R² (Modell 5: Basis + Extra Noise):      0.545213
R² (Modell 6: Basis + Random):           0.545180
R² (Modell 7: Basis + Avg. NotBedrooms): 0.545194
R² (Modell 8: 3 Var + Postal Code):      0.374251
```

In [ ]: