

Q1. Product Reviews 1a) Target (continuous) Future sales volume (units) over a fixed horizon (e.g., next 30 days). 1b) Linear model with rating + judgment words Let Score be the numeric rating. Let freq\_good, freq\_bad, freq\_doesnt\_work be normalized word frequencies (e.g., per 100 tokens). Model:  $\text{SalesNext30d} = b_0 + b_1 \cdot \text{Score} + b_2 \cdot \text{freq\_good} + b_3 \cdot \text{freq\_bad} + b_4 \cdot \text{freq\_doesnt\_work} + \text{error}$  1c) Reducing features - LASSO (L1) regularization to shrink unhelpful coefficients to zero. - Dimensionality reduction (e.g., PCA on text features). - Aggregate to a single sentiment score instead of multiple word counts. 1d) Mixed scales (1-5 vs 1-10) Normalize ratings to a common scale before modeling (e.g., convert 1-10 to 1-5 by dividing by 2, or standardize to z-scores). Optionally add an indicator feature for original scale (e.g., is\_scale\_10) to allow different intercepts/slopes if needed. Without normalization, the Score coefficient is not comparable across scales. Q2. Fruit dataset 2a) Categorical variables type, color, size 2b) Total variables after one-hot coding - If you one-hot encode all levels (no drop): 7 dummy vars + price = 8 total. - If you use drop-first (to avoid multicollinearity with an intercept): 4 dummy vars + price = 5 total. (Counting only the encoded features: 7 vs 4.) Q3. Small linear regression Given: House 1: Size=1400, Beds=3, Price=245 House 2: Size=1600, Beds=3, Price=312 House 3: Size=1700, Beds=4, Price=279 (Price in 1000s) Model:  $\text{Price} = \beta_0 + \beta_1 \cdot \text{Size} + \beta_2 \cdot \text{Bedrooms}$  (We'll scale Size by 100 for easier arithmetic:  $\text{Size}_{100} = \text{Size}/100$ ) 3a) Feature matrix  $A$  
$$= \begin{bmatrix} 14 & 3 \\ 16 & 3 \\ 17 & 4 \end{bmatrix}$$
 3b) Target vector  $y = [245, 312, 279]^T$  3c) Normal equation  $\beta = (A^T A)^{-1} A^T y$  gives  $\beta = [\beta_0, \beta_1, \beta_2]$  
$$= [-24.5, 33.5, -66.5]$$
 (If you do NOT scale Size:  $\beta = [-24.5, 0.335, -66.5]$ ) 3d) Predict for Size = 1500 sqft ( $\text{Size}_{100} = 15$ ), Bedrooms = 3:  $\text{Price} = -24.5 + 33 \cdot 15 + 278.5 = 278.5$  (in 1000s)  $\Rightarrow$  Predicted price  $\approx$  \$278,500 Q4. When is  $(A^T A)$  invertible? Claim. For  $A \in \mathbb{R}^{n \times p}$ ,  $(A^T A)$  is invertible  $\Leftrightarrow$  the columns of  $A$  are linearly independent. Proof. ( $\Rightarrow$ ) Suppose  $A^T A$  is invertible. If  $A$ 's columns were dependent, there would exist  $x \neq 0$  with  $Ax = 0$  (i.e., a nontrivial vector in the null space of  $A$ ). Then  $A^T A x = A^T (Ax) = A^T 0 = 0$ , so  $x$  is in the null space of  $A^T A$ . But an invertible matrix has only the zero vector in its null space. Contradiction. Hence  $A$ 's columns are linearly independent. ( $\Leftarrow$ ) Suppose  $A$ 's columns are linearly independent. Then  $\text{rank}(A) = p$  (full column rank), so for any  $x \neq 0$  we have  $Ax \neq 0$  and  $x^T A^T A x = \|Ax\|^2 > 0$ . Thus  $A^T A$  is symmetric positive definite and therefore invertible. (Equivalently, by the rank identity  $\text{rank}(A^T A) = \text{rank}(A) = p$ ; since  $A^T A$  is  $p \times p$  with rank  $p$ , it is invertible.)

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression

california = fetch_california_housing(as_frame=True)
X_df = california.data
y = california.target

print("5a) Feature names:", list(X_df.columns))

print("\n5b) Rows 10-15 for HouseAge, AveRooms, Population:")
print(X_df.loc[10:15, ['HouseAge', 'AveRooms', 'Population']])

# 5c) Simple linear regression: MEDV ~ AveOccup
X = X_df[['AveOccup']].values
n = len(y)
split = (2*n)//3
X_train, X_test = X[:split], X[split:]
y_train, y_test = y.values[:split], y.values[split:]

model = LinearRegression()
model.fit(X_train, y_train)

print("\n5d) R^2 (train):", model.score(X_train, y_train))
print("5d) R^2 (test):", model.score(X_test, y_test))
print("Intercept:", model.intercept_)
print("Slope for AveOccup:", model.coef_[0])

plt.figure()
sample = np.arange(0, n, 50)
plt.scatter(X_df[['AveOccup']].iloc[sample], y.iloc[sample], s=8)
xs = np.linspace(X.min(), X.max(), 200).reshape(-1,1)
ys = model.predict(xs)
plt.plot(xs, ys, linewidth=2)
plt.xlabel("AveOccup")
plt.ylabel("MedHouseValue ($100k)")
plt.title("Simple Linear Regression: MedHouseValue ~ AveOccup")
plt.show()
```

5a) Feature names: ['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitude']

5b) Rows 10-15 for HouseAge, AveRooms, Population:

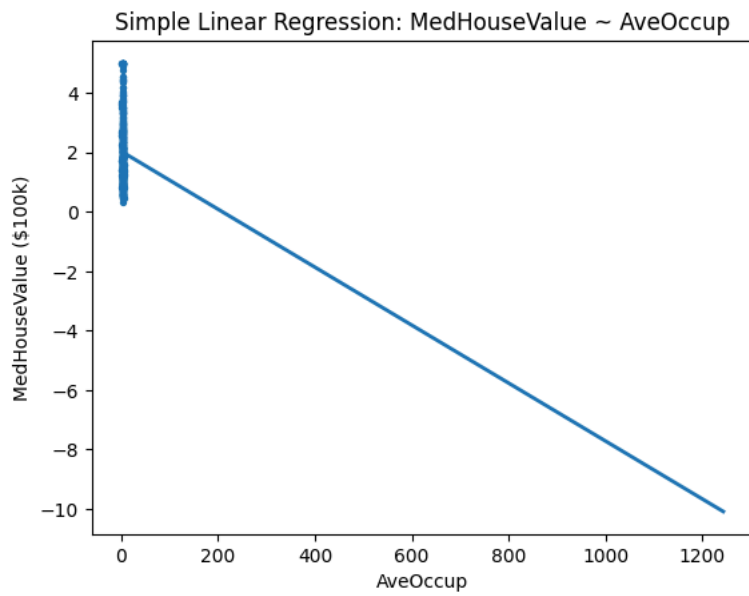
	HouseAge	AveRooms	Population
10	52.0	5.477612	910.0
11	52.0	4.772480	1504.0
12	52.0	5.322650	1098.0
13	52.0	4.000000	345.0
14	52.0	4.262903	1212.0
15	50.0	4.242424	697.0

5d) R^2 (train): 0.0023374510886858824

5d) R^2 (test): -0.03700899339889241

Intercept: 2.038064223107151

Slope for AveOccup: -0.009751795964041849



5e)  $R^2$ : fraction of target variance explained; expect low with one predictor. Big train-test gap  $\Rightarrow$  overfitting; both low  $\Rightarrow$  weak model. Q6: 6a)  $b \notin \text{col}(A)$  6b)  $b \in \text{col}(A)$  and  $\text{rank}(A)=n$  6c)  $b \in \text{col}(A)$  and  $\text{rank}(A)<n$