**Human-AI Decision Study**

**Goal:** To study if the use of AI affects our decision making positively or negatively

**Process:**

**Initial Setup:**

*mkdir human-ai-decision-study*

*cd human-ai-decision-study*

*python3 -m venv .venv*

*.venv/bin/activate*

*Python -m pip install –-upgrade pip*

*pip install scikit-learn pandas numpy joblib SHAP streamlit*

*printf ".venv/\n\_\_pycache\_\_/\ndata/\*.csv\nmodels/\*.pkl\n.DS\_Store\n" > .gitignore* ( to ignore the local files and stuff while pushing to git)

*mkdir -p app analysis models data figures* (-p: parent doing this creates all those directories at once inside the parent directory)

**DataSet:**

Using the Caifornia housing data available in scikit library.

 **Library:** scikit-learn → a Python package for machine learning

 **Module:** sklearn.datasets → a subpart (module) inside scikit-learn that contains sample datasets

 **Function:** fetch\_california\_housing() → one of the functions in that module that loads a specific dataset

**Use the following code to check the data. The data is clean and no missing values.**

*from sklearn.datasets import fetch\_california\_housing*

*data = fetch\_california\_housing(as\_frame=True)*

*print(data.frame.head())*

We can use print(data.keys()) to print the keys which looks like:

**dict\_keys(['data', 'target', 'frame', 'target\_names', 'feature\_names', 'DESCR'])**

We can find the target column name with print(data.target\_names). Here it is:

**['MedHouseVal']**

**Conclusion from data set:**

The DataFrame includes 8 feature columns + 1 target column.

**Code**

"""

Baseline model training for 'human-ai-decision-study'.

What this script does:

1) Loads the California Housing dataset from scikit-learn.

2) Splits into train/test.

3) Trains a RandomForestRegressor (tabular-ML baseline).

4) Evaluates with Mean Absolute Error (MAE).

5) Saves the trained model and a small test sample for later steps.

Run from project root with your venv active:

python analysis/train\_model.py

"""

import joblib

import pandas as pd

from sklearn.datasets import fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

*# 1) Load the California Housing dataset*

*# as\_frame=True returns a pandas DataFrame*

data = fetch\_california\_housing(as\_frame=True)

df = data.frame.copy() *#making a copy of the dataframe to avoid modifying the original data*

*#print(data.target\_names) -> ['MedHouseVal']*

*#Sanity: target column is MedHouseVal*

*# 2) Separate features(X) and target(Y)*

X = df.drop(columns=['MedHouseVal']) *#all input features*

y = df['MedHouseVal'] *#target column, the value we want to predict*

*# 3) Split into train/test*

*# # Keep 20% aside as unseen test data to estimate generalization performance.*

*# random\_state=42 makes the split reproducible.*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.20, random\_state=42

)

*# ---------- 4) Define the model ----------*

*# Random Forest is a strong, low-maintenance baseline for tabular data.*

*# n\_estimators: number of trees. More trees -> more stable up to a point.*

*# n\_jobs=-1 uses all CPU cores to speed up training.*

model = RandomForestRegressor(

n\_estimators=400,

max\_depth=14,

min\_samples\_leaf=2,

random\_state=42,

n\_jobs=-1

)

*# ---------- 5) Train (fit) the model ----------*

model.fit(X\_train, y\_train)

*# ---------- 6) Evaluate on test data ----------*

*# MAE is average |prediction - true|. Lower is better.*

preds = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, preds)

print(f"Test MAE: {mae:.3f}") *# e.g., 0.45 ≈ $45,000 average error*

*# ---------- 7) Save artifacts ----------*

*# Save trained model to reuse later (so you don't retrain every time).*

joblib.dump(model, "models/model.pkl")

print("Saved model → models/model.pkl")

*# Save a small slice of the test set for tomorrow's demos (app + SHAP).*

sample = X\_test.copy()

sample["y\_true"] = y\_test.values

sample.head(100).to\_csv("data/test\_sample.csv", index=False)

print("Saved test sample → data/test\_sample.csv")