CS418: Rising Cost of Homes and Unaffordability - Final Report

BIG DATA DUNCES This is our **Github repository Link** : https://github.com/uic-ds-fall2025/class-project-big-data-dunces

---

Team Members:

Adnan | Phil | Sufyan | David | Yanja

Introduction

For our data science project we analyzed why owning a home in the United States has become more unattainable. Is this due to lower financial literacy? Is this due to housing prices shooting up? Does this have something to do with income not rising at the same rate? We are investigating these questions.

Our project, **"Rising Cost of Homes and Unaffordability**," investigates how housing affordability in the United States has evolved over time, with a focus on generational and economic factors that make homeownership increasingly unattainable; particularly for Millennials and Gen Z.

We are analyzing multiple datasets that together provide a comprehensive view of the housing market and its socioeconomic drivers:

California Housing Dataset – includes variables such as median income, median house value, population, and geographic coordinates, allowing us to explore spatial trends in affordability.

NFCS State Data – captures demographic and financial behavior at the state level, including living arrangements, marital status, and financial well-being indicators.

Household Income Regression Dataset – contains detailed individual-level attributes such as age, education level, occupation, income, employment status, and homeownership, which we use to identify patterns between personal characteristics and homeownership likelihood.

New York Housing Dataset – provides detailed listing-level data for New York, enabling localized analysis of property prices and market variations.

Median U.S. Household Income and Median Home Price (1984–2024) – two time-series datasets that help us analyze the long-term divergence between household earnings and home prices.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/My Drive/Colab Notebooks/418BigDataDunces
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/My Drive/Colab Notebooks/418BigDataDunces
```

Data Cleaning (Needed for ML/Stats)

All **cleaned datasets and code scripts** used to clean the data are available in our Github repository : https://github.com/uic-ds-fall2025/class-project-big-data-dunces

Here are the **scripts**: https://github.com/uic-ds-fall2025/class-project-big-data-dunces/blob/main/CS418_Project_Datasets.ipynb

And here is the **updated scripts in a pdf format**: https://github.com/uic-ds-fall2025/class-project-big-data-dunces/blob/main/Final%20report%20data%20cleaning%20scripts.pdf

We had to collapse data cleaning code as the pdf was getting longer than 10 pages. We do have the data scripts in our github links pasted above.

↳ 5 cells hidden

ML/Stats

For our machine learning part the first ML technique we did was a logistic regression model to see how well we could predict whether someone is a homeowner or a renter based on the information in the household income dataset. We included things like age, education level, marital status, work experience, household size, income, and location. We also added a log version of the income to shrink the effect of the very large values and income per person, which divides income by the number of people in the household.

before training the model, we encoded the different categorical variables and we scaled the numeric ones. There were more house owners than renters in the data so we used class weighting so the model wouldn't lean toward predicting owner for the results.

After we trained and tested the model with an 80/20 split the accuracy came out to 63.2% which is better than the baseline accuracy of 59.6% from always guessing the majority class. The performance was not balanced. For the owners, the model did well with precision 067 and recall 0.77. For renters, the model struggled and returned 0.55 for precision and 0.42 for recall. This shows that it finds owners much easier than renters.

We tried adding engineered income features but they didn't improve the results much. The most likely reason for this is that the real differences between renters and owners aren't fully captured by the variables that we have. The local housing market, savings, debt, credit score, and other financial factors would likely matter a lot but they aren't included in the dataset.

Overall, this model shows that income and basic demographics only explain part of the store. The harder that it becomes to afford a home, the less income alone predicts whether someone actually owns one. This lines up with what we are seeing with the rest of the project.

To expand on our findings from the first ML analysis, we trained a Random Forest regression model to predict median house value in the California housing dataset using geographic, demographic, and economic features (e.g., latitude, longitude, housing_median_age, median_income, and ocean_proximity) that weren't fully captured previously.

Our Random Forest regression model achieved an Root Mean Squared Error of about 48.7k and an R^2 of 0.82, significantly improving over a mean-value baseline (RMSE ≈ $114k, R^2 ≈ 0). This indicates that median income and geographic factors together explain most of the variation in house prices, reinforcing the idea that location and earnings power are central to the affordability challenges faced by potential young homebuyers.

```
import numpy as np
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.dummy import DummyClassifier

# Import data
df = pd.read_csv("Regression-Dataset-For-Household-Income-Analysis.cleaned.csv")

# Engineer porentially relevant features
df["log_income"] = np.log1p(df["income"])
df["income_per_person"] = df["income"] / df["household_size"].replace(0, np.nan)

# Feature / target setup
features = ["age","education_level","marital_status","work_experience",
            "household_size","income","log_income","income_per_person","location"]
target = "homeownership_status"
X = df[features].dropna()
y = df.loc[X.index, target].map({"Own":1,"Rent":0})

# Preprocessing
categorical = ["education_level","marital_status","location"]
numeric = ["age","work_experience","household_size","income","log_income","income_per_person"]
preprocess = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), categorical),
```

```
    ("num", StandardScaler(), numeric)
])

# Model
pipe = Pipeline([
    ("prep", preprocess),
    ("clf", LogisticRegression(max_iter=1000, class_weight="balanced"))
])

# Split / train / evaluate
X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
pipe.fit(X_tr, y_tr)
pred = pipe.predict(X_te)
print("Accuracy:", accuracy_score(y_te, pred))
print(classification_report(y_te, pred, target_names=["Rent","Own"]))

dummy = DummyClassifier(strategy="most_frequent")
dummy.fit(X_tr, y_tr)
baseline_preds = dummy.predict(X_te)

baseline_acc = accuracy_score(y_te, baseline_preds)
print("Baseline accuracy:", baseline_acc)
```

```
Accuracy: 0.632
              precision    recall  f1-score   support

        Rent       0.55      0.42      0.48       796
         Own       0.67      0.77      0.72      1204

    accuracy                           0.63      2000
   macro avg       0.61      0.60      0.60      2000
weighted avg       0.62      0.63      0.62      2000

Baseline accuracy: 0.602
```

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.dummy import DummyRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load cleaned California housing data
housing = pd.read_csv("housing_clean.csv")

print(housing.head())
print(housing.columns)

# Define features and target
target = "median_house_value"
if target not in housing.columns:
    raise ValueError("median_house_value not found in housing dataframe. Check column names.")

# All columns except target are features
feature_cols = [c for c in housing.columns if c != target]

X = housing[feature_cols].copy()
y = housing[target].copy()

# Identify numeric vs categorical
categorical = []
numeric = []

for col in X.columns:
    if X[col].dtype == "object":
        categorical.append(col)
    else:
        numeric.append(col)

print("Numeric features:", numeric)
print("Categorical features:", categorical)

# Train / test / split
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Baseline model: always predict mean house value
baseline = DummyRegressor(strategy="mean")
baseline.fit(X_train, y_train)
y_pred_base = baseline.predict(X_test)

base_mse = mean_squared_error(y_test, y_pred_base)
base_rmse = np.sqrt(base_mse)
base_mae = mean_absolute_error(y_test, y_pred_base)
base_r2 = r2_score(y_test, y_pred_base)

print("\n=== Baseline (mean predictor) ===")
print("RMSE:", base_rmse)
print("MAE :", base_mae)
print("R^2 :", base_r2)

# Preprocessing + Random Forest regression model
preprocess = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), categorical),
        ("num", StandardScaler(), numeric)
    ]
)

rf_model = Pipeline(steps=[
    ("prep", preprocess),
    ("rf", RandomForestRegressor(
        n_estimators=300,
        random_state=42,
        n_jobs=-1
    ))
])

# Train RF model
rf_model.fit(X_train, y_train)

# Evaluate
y_pred = rf_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\n=== Random Forest Regressor ===")
print("RMSE:", rmse)
print("MAE :", mae)
print("R^2 :", r2)

# Get feature names after encoding
prep = rf_model.named_steps["prep"]
rf = rf_model.named_steps["rf"]

# Encoded feature names
encoded_cat = prep.named_transformers_["cat"].get_feature_names_out(categorical)
all_features = np.concatenate([encoded_cat, np.array(numeric)])

importances = rf.feature_importances_
idx_sorted = np.argsort(importances)[::-1][:15]  # top 15

print("\nTop 15 feature importances:")
for i in idx_sorted:
    print(f"{all_features[i]}: {importances[i]:.4f}")
```

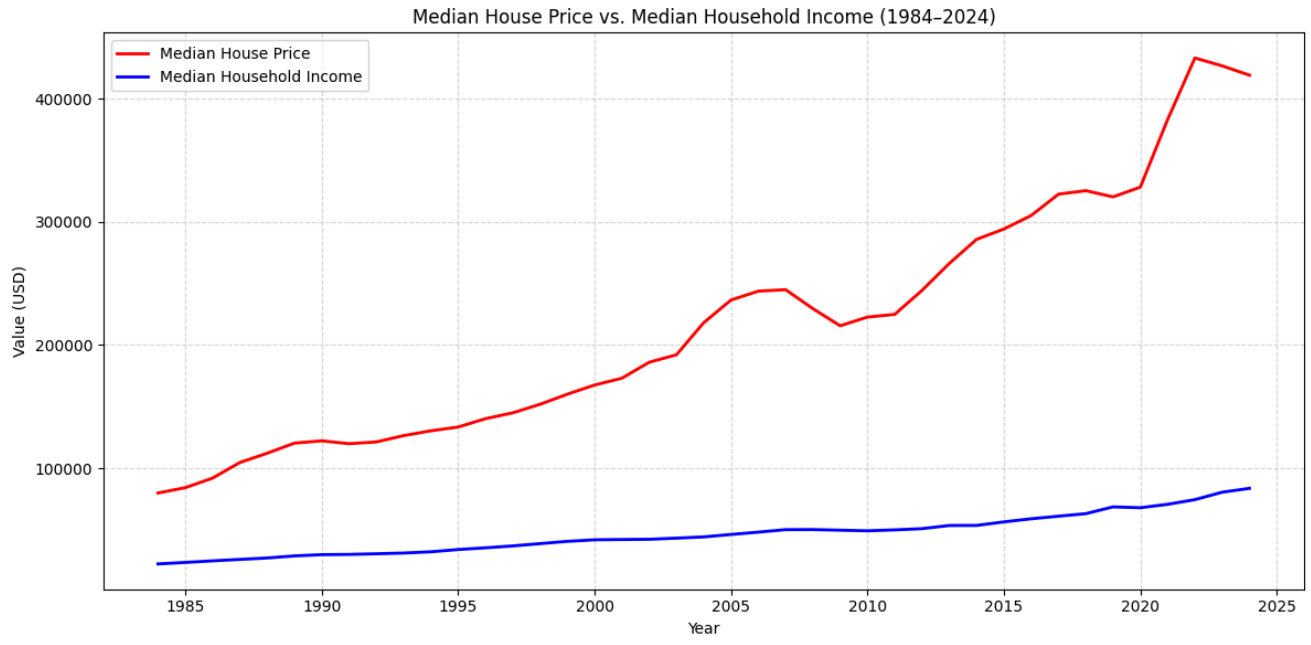
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
```

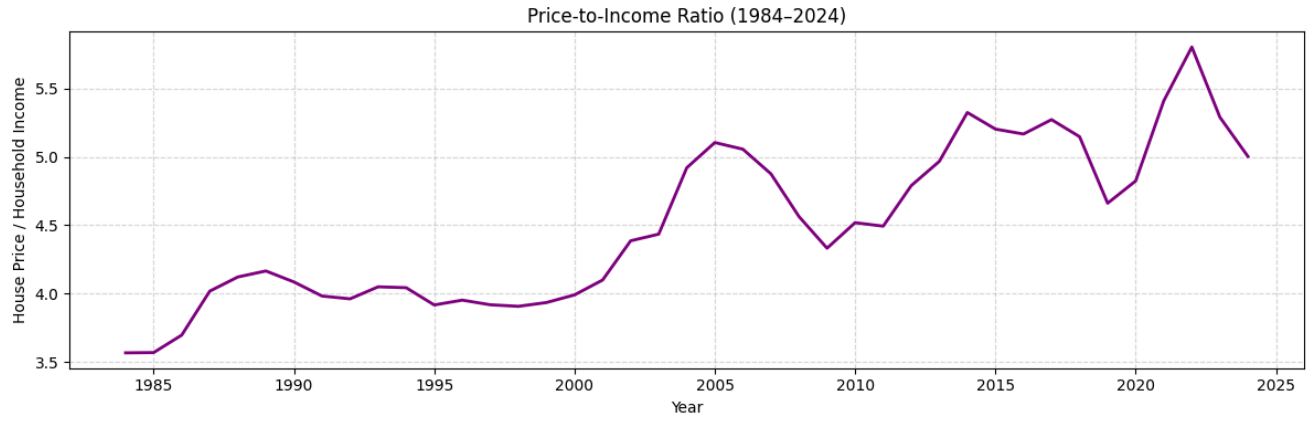
```
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
Numeric features: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', '
Categorical features: ['ocean_proximity']

=== Baseline (mean predictor) ===
RMSE: 114485.63543099792
MAE : 90606.8549000715
R^2 : -0.00021908714592466794

=== Random Forest Regressor ===
RMSE: 48719.3214874529
MAE : 31467.895266472868
R^2 : 0.8188678248080926

Top 15 feature importances:
median_income: 0.4900
ocean_proximity_INLAND: 0.1414
longitude: 0.1061
latitude: 0.1012
housing_median_age: 0.0524
population: 0.0325
total_bedrooms: 0.0238
total_rooms: 0.0237
households: 0.0180
ocean_proximity_NEAR OCEAN: 0.0063
ocean_proximity_<1H OCEAN: 0.0036
ocean_proximity_NEAR BAY: 0.0008
ocean_proximity_ISLAND: 0.0003
```

Visualization

```
# Load data
income = pd.read_csv("Median Household Income.csv")
house_price = pd.read_csv("Median House Price.csv")

# --- ONLY FIX: correct the column names ---
income = income.rename(columns={"Median HH Income": "Median Household Income"})
house_price = house_price.rename(columns={"Median House Price ": "Median House Price"})


# Merge datasets on Year
merged = income.merge(house_price, on="Year")

merged["Price_to_Income"] = (
    merged["Median House Price"] / merged["Median Household Income"]
)

# ------------------------------------------
# Plot 1 — Same-axis comparison (clean + honest)
# ------------------------------------------
plt.figure(figsize=(12, 6))

plt.plot(
    merged["Year"],
    merged["Median House Price"],
    label="Median House Price",
    color="red",
    linewidth=2
)

plt.plot(
    merged["Year"],
    merged["Median Household Income"],
    label="Median Household Income",
    color="blue",
    linewidth=2
)

plt.title("Median House Price vs. Median Household Income (1984–2024)")
```

```
plt.xlabel("Year")
plt.ylabel("Value (USD)")
plt.grid(True, linestyle="--", alpha=0.5)
plt.legend()
plt.tight_layout()
plt.show()

###############################################3

plt.figure(figsize=(12, 4))
plt.plot(
    merged["Year"],
    merged["Price_to_Income"],
    color="purple",
    linewidth=2
)

plt.title("Price-to-Income Ratio (1984–2024)")
plt.xlabel("Year")
plt.ylabel("House Price / Household Income")
plt.grid(True, linestyle="--", alpha=0.5)
plt.tight_layout()
plt.show()
```



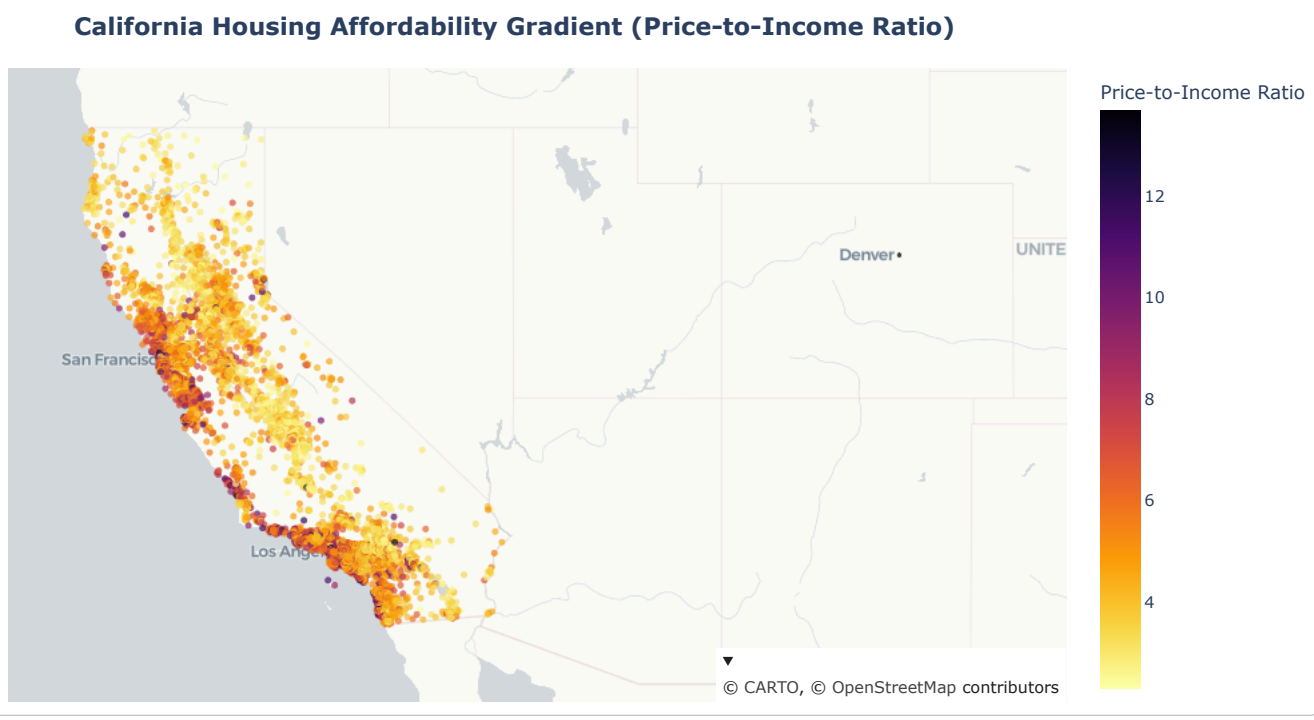


```
pip install plotly
```

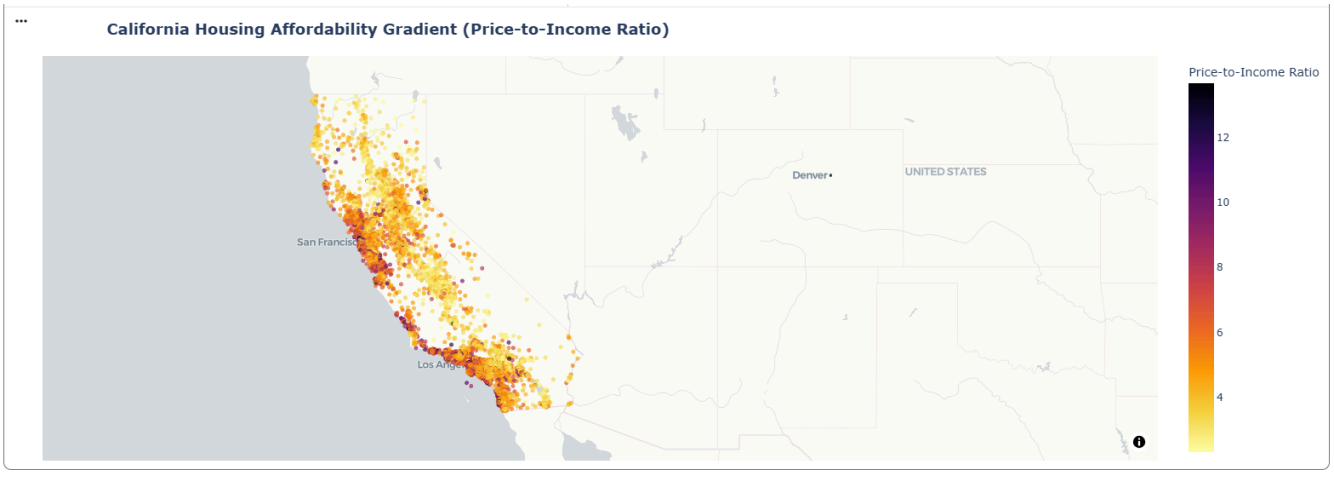
```
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly) (9.1.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from plotly) (25.0)
```

```
import plotly.express as px
import plotly.io as pio
import matplotlib.pyplot as plt
%matplotlib inline

HousingData = pd.read_csv("housing_clean.csv")
HouseholdIncomeData = pd.read_csv("Regression-Dataset-For-Household-Income-Analysis.cleaned.csv")


def assign_age_cohort(age):
    """Categorize ages into generational cohorts."""
    if pd.isna(age):
        return np.nan
    age = int(age)
    if 18 <= age <= 27:
        return "Gen Z (18–27)"
    elif 28 <= age <= 43:
        return "Millennial (28–43)"
    elif 44 <= age <= 59:
        return "Gen X (44–59)"
    elif age >= 60:
        return "Boomer+ (60+)"
    else:
        return "Under 18"

# Apply to household income dataset (which contains age info)
HouseholdIncomeData["age_cohort"] = HouseholdIncomeData["age"].apply(assign_age_cohort)



def calc_price_to_income(df, income_col, price_col):
    """
    Compute price-to-income ratio.
    Note: In HousingData, median_income is in units of 10,000 dollars.
    """
    df = df.copy()
    df["price_to_income_ratio"] = df[price_col] / (df[income_col] * 10000)
    # Clean up extreme outliers
    df = df[df["price_to_income_ratio"].between(df["price_to_income_ratio"].quantile(0.01),
                                                df["price_to_income_ratio"].quantile(0.99))]
    return df

# Apply to housing dataset
HousingData = calc_price_to_income(HousingData, "median_income", "median_house_value")


# Check price_to_income_ratio exists, if not create
if "price_to_income_ratio" not in HousingData.columns:
    HousingData["price_to_income_ratio"] = HousingData["median_house_value"] / (HousingData["median_income"] * 10000)

# Removing outliers (1st–99th percentile)
q_low, q_high = HousingData["price_to_income_ratio"].quantile([0.01, 0.99])
HousingData = HousingData.query("@q_low <= price_to_income_ratio <= @q_high")

# Plotly
fig1 = px.scatter_map(
    HousingData,
    lat="latitude",
    lon="longitude",
    color="price_to_income_ratio",
    color_continuous_scale=px.colors.sequential.Inferno[::-1],
    zoom=5,
    opacity=0.7,
    hover_data=["median_income", "median_house_value", "ocean_proximity"],
    title="California Housing Affordability Gradient (Price-to-Income Ratio)",
)

# --- Map style
fig1.update_layout(
    map_style="carto-positron",   # clean base map with light coastlines
    map_center={"lat": 36.5, "lon": -119.5},
```

```
    map_center={"lat": 36.5, "lon": -119.5},
    margin={"r":0,"t":50,"l":0,"b":0},
    title_font=dict(size=18, weight='bold'),
    coloraxis_colorbar=dict(title="Price-to-Income Ratio")
)

# Display in Notebook
fig1.show()
```



**Below is a screenshot of the visualization since you might not see the "legend" due to pdf formatting



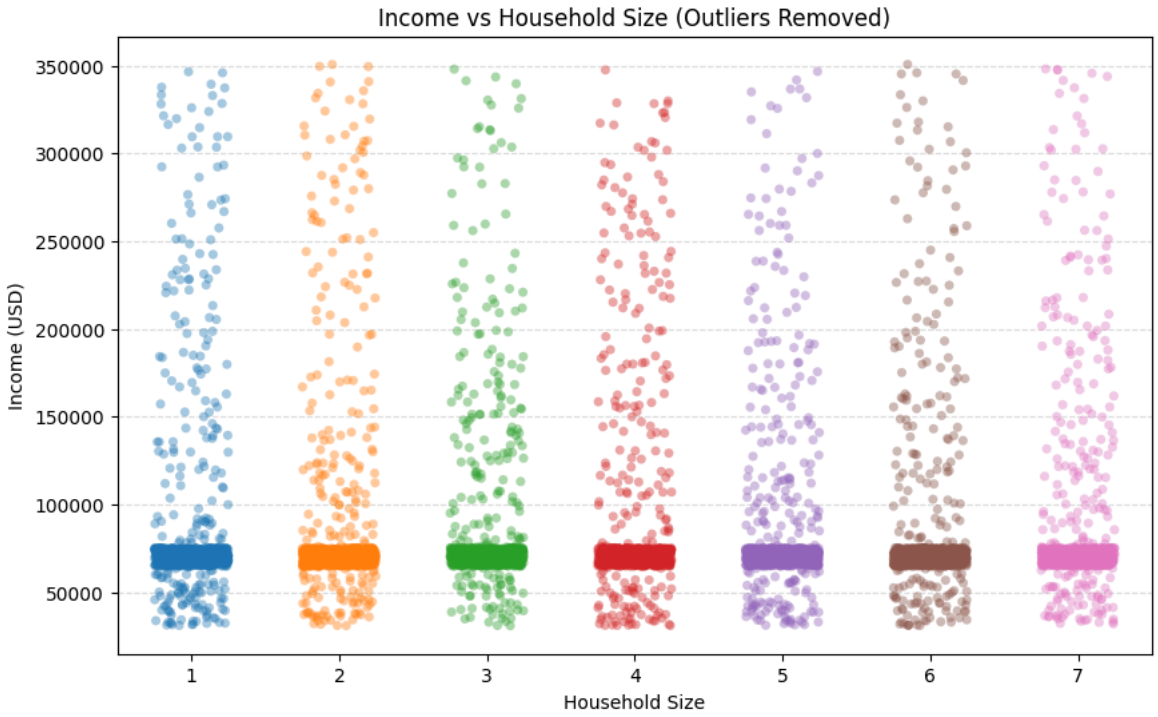
Additional Work

For the additional work section, we have decided to create an additional visualization.

The following visualization explores household incomes for various household sizes.

```
# Load dataset
regression = pd.read_csv("Regression-Dataset-For-Household-Income-Analysis.cleaned.csv")

# ----------------------------------------
# Remove Top 5% of High-Income Outliers
```

```
cutoff = regression["income"].quantile(0.75)
filtered = regression[regression["income"] <= cutoff]

# -----------------------------------------
# Scatter Plot – Income vs Household Size

plt.figure(figsize=(10,6))
sns.stripplot(
    data=filtered,
    x="household_size",
    y="income",
    jitter=0.25,
    alpha=0.4,
    hue = "household_size",
    legend = False,
    palette="tab10"
)

plt.title("Income vs Household Size (Outliers Removed)")
plt.xlabel("Household Size")
plt.ylabel("Income (USD)")
plt.grid(axis='y', linestyle='--', alpha=0.4)

# Fix y-axis from scientific notation to full numbers
plt.ticklabel_format(style='plain', axis='y')

plt.show()
```



Gen AI Guidance

The suggestions that Gen AI models provided in the last milestone were useful. We followed some of those to improve our project.

One of the suggestions was to consolidate affordability metrics, rather than examining income and house values separately. That is why we have visualizations that portray the price-to-income ratio.

Another suggestion for the visualizations was to stratify the analysis by subcategories to reveal subgroup differences, which is what we've done for the visualization in the additional work section. It helps us identify income levels for different household sizes, whether families earn more, and if larger families earn more to support themselves, among others. This helps us identify that incomes stay steady for larger households, resulting in less affordable housing.

Our first ML analysis performed slightly better than baseline so GenAI suggested that more accurately predicting affordability can be done by observing more unseen variables like geographic and economic conditions. Our second regression model performed much better than the baseline and we also had it rank the importance of each feature in its predictions to help us better understand what factors influenced home costs the most. We learned housing markets are highly predictable given income and location, suggesting that systemic geographic cost factors—not personal financial choices—are the primary drivers of housing unaffordability for young adults.

Some of the other suggestions didn't apply to our situation because ChatGPT simply didn't understand our datasets and requirements well enough and provided some improper suggestions.

Results

Over the last 40 years, median US house prices and median US household incomes have not grown at the same pace. The line chart shows a widening gap beginning in the late 1990s, accelerating during the 2000s housing boom, dipping in 2008, and then increasing sharply after 2020. The price-to-income ratio confirms this trend: affordability worsened significantly as the ratio climbed from ~3 in the 1980s to over 6 in the 2020s.

Understanding this is important as it connects to other trends: slowing wage growth, generational wealth inequality, and regional housing crises. Investigating this relationship helps quantify why homeownership, once considered a hallmark of the American middle class, has become out of reach for many families today despite overall economic growth.

Coastal regions like Los Angeles, San Francisco, and San Diego have been economic and cultural hubs with dense job markets, a variety of lifestyle amenities, and a limited housing supply. However, these advantages come with a steep affordability cost. By comparing median house value to median household income across thousands of tracts, the price-to-income ratio reveals where housing costs most exceed what residents can reasonably afford. The ratio for many people in the region reaches as high as 12, indicating that they have no choice but to buy such expensive homes on average salaries, as homes are becoming increasingly unaffordable.

The third visualization, in the additional work section, highlights how income varies across different household sizes. Median income appears similar for households of all sizes. The mean and median hover at around 70k - 80k for all household sizes. This suggests that larger households experience stronger affordability pressure, as increasing dependents do not correspond proportionally to higher income.

Compared to a baseline that always predicts the average house value, our Random Forest (2nd) model substantially improves prediction accuracy, reducing RMSE from about \$114k to \$49k and MAE from about \$91k to \$31k. An R^2 of 0.82 indicates that the model explains roughly 82% of the variation in median house values, which is strong for housing data. Feature importance analysis shows that median income is the dominant predictor of house values, followed by location-related variables such as distance from the ocean and geographic coordinates. This indicates that median income and geographic factors together explain most of the variation in house prices, reinforcing the idea that location and earnings power are central to the affordability challenges faced by potential young homebuyers.

Homeownership has become less attainable for the average American. Wage stagnation and rising property costs are widening the gap. Data-driven insight like this helps policymakers target affordability reforms.

These insights emphasize the growing affordability crisis and regional economic disparities across the U.S