In [ ]:
```
Machine Learning CIA-3: Created by Jibrael Jos, PhD
CIA 3:  Classifier or Regression with own dataset
Student Name: Rachna Verma
Roll No: 24122023
Submission: 12/05/25
```

🏠 Real Estate Affordability Classification

📘 : Introduction

🏠 Real Estate Affordability Classification

In this project, we aim to classify cities as **"Affordable"** or **"Not Affordable"** based on various real estate indicators.
We use features like:

- Price to Income Ratio
- Rental Yield (City & Outside)
- Mortgage as a % of Income
- Price to Rent Ratio

The goal is to build a classification model that helps determine affordability for city-level housing markets.

📥 : Import Libraries & Load Dataset

In [132…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import requests
from bs4 import BeautifulSoup

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# # Load dataset
# df = pd.read_csv("your_dataset.csv")  # replace with your file name
# df.head()
```

In [134…

```
url = "https://www.numbeo.com/property-investment/rankings.jsp"
headers = {"User-Agent": "Mozilla/5.0"}

try:
    response = requests.get(url, headers=headers, timeout=10)
    response.raise_for_status()
    soup = BeautifulSoup(response.content, "html.parser")
    table = soup.find("table", id="t2")
    table_headers = [th.get_text(strip=True) for th in table.find_all("th")]
    data = [[col.get_text(strip=True) for col in row.find_all("td")] for row in
    df = pd.DataFrame(data, columns=table_headers)
except requests.exceptions.RequestException as e:
    print(f"Error fetching data: {e}")

print("Dataset Overview:\n")
print(df.head())
```

Dataset Overview:

	Rank	City	Price To Income Ratio	\
0		Colombo, Sri Lanka	37.0	
1		Kathmandu, Nepal	36.6	
2		Beijing, China	34.7	
3		Shanghai, China	34.0	
4		Ho Chi Minh City, Vietnam	34.0	

	Gross Rental Yield City Centre	Gross Rental Yield Outside of Centre	\
0	5.0	2.7	
1	1.7	1.9	
2	1.6	1.7	
3	1.6	1.6	
4	2.7	3.4	

	Price To Rent Ratio City Centre	Price To Rent Ratio Outside Of City Centre	\
0	20.0	37.0	
1	59.6	53.1	
2	64.4	57.9	
3	62.1	63.9	
4	36.5	29.0	

	Mortgage As A Percentage Of Income	Affordability Index
0	693.9	0.1
1	512.8	0.2
2	249.4	0.4
3	246.7	0.4
4	381.7	0.3

In [136…

```
df
```

Out[136...

	Rank	City	Price To Income Ratio	Gross Rental Yield City Centre	Gross Rental Yield Outside of Centre	Price To Rent Ratio City Centre	Price To Rent Ratio Outside Of City Centre	Mortgage As A Percentage Of Income	Afford
0		Colombo, Sri Lanka	37.0	5.0	2.7	20.0	37.0	693.9	
1		Kathmandu, Nepal	36.6	1.7	1.9	59.6	53.1	512.8	
2		Beijing, China	34.7	1.6	1.7	64.4	57.9	249.4	
3		Shanghai, China	34.0	1.6	1.6	62.1	63.9	246.7	
4		Ho Chi Minh City, Vietnam	34.0	2.7	3.4	36.5	29.0	381.7	
...	...	...	...	...	...	...	...	...	
276		Omaha, NE, United States	2.3	18.2	14.1	5.5	7.1	20.4	
277		Pretoria, South Africa	2.2	11.4	11.5	8.8	8.7	28.4	
278		Indianapolis, IN, United States	2.0	17.0	14.6	5.9	6.8	17.7	
279		Saint Louis, MO, United States	1.7	17.5	16.7	5.7	6.0	15.6	
280		Cleveland, OH, United States	1.7	22.0	20.1	4.5	5.0	15.6	

281 rows × 9 columns

In [138...

```
df.columns
```

Out[138...

```
Index(['Rank', 'City', 'Price To Income Ratio',
       'Gross Rental Yield City Centre',
       'Gross Rental Yield Outside of Centre',
       'Price To Rent Ratio City Centre',
       'Price To Rent Ratio Outside Of City Centre',
       'Mortgage As A Percentage Of Income', 'Affordability Index'],
      dtype='object')
```

📊: Exploratory Data Analysis (EDA)

📊 Exploratory Data Analysis (EDA)

In this step, we explore patterns and relationships in the dataset.
We focus on identifying:

- Distribution of features
- Correlations
- Outliers and Skewness
- Insights about affordability and pricing

In [142…
```
# Basic structure
print("\nDataset Info:")
print(df.info())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 281 entries, 0 to 280
Data columns (total 9 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Rank                                    281 non-null    object
 1   City                                    281 non-null    object
 2   Price To Income Ratio                   281 non-null    object
 3   Gross Rental Yield City Centre          281 non-null    object
 4   Gross Rental Yield Outside of Centre    281 non-null    object
 5   Price To Rent Ratio City Centre         281 non-null    object
 6   Price To Rent Ratio Outside Of City Centre  281 non-null    object
 7   Mortgage As A Percentage Of Income      281 non-null    object
 8   Affordability Index                     281 non-null    object
dtypes: object(9)
memory usage: 19.9+ KB
None
```

In [144…
```
# List of columns to convert
cols_to_convert = [
    'Price To Income Ratio',
    'Gross Rental Yield City Centre',
    'Gross Rental Yield Outside of Centre',
    'Price To Rent Ratio City Centre',
    'Price To Rent Ratio Outside Of City Centre',
    'Mortgage As A Percentage Of Income',
    'Affordability Index'
]

# Clean and convert
for col in cols_to_convert:
    df[col] = df[col].replace('[%,]', '', regex=True)            # Remove commas
    df[col] = pd.to_numeric(df[col], errors='coerce')            # Convert to nu
    df[col] = df[col].round().astype('Int64')                    # Round and con
```

In [146…
```
print(df.dtypes)      #................................CHECK DATATYPE
```

```
Rank                                          object
City                                          object
Price To Income Ratio                          Int64
Gross Rental Yield City Centre                 Int64
Gross Rental Yield Outside of Centre           Int64
Price To Rent Ratio City Centre                Int64
Price To Rent Ratio Outside Of City Centre     Int64
Mortgage As A Percentage Of Income             Int64
Affordability Index                            Int64
dtype: object
```

In [148…
```
print(df.head())
```

```
  Rank                         City  Price To Income Ratio  \
0              Colombo, Sri Lanka                      37
1                Kathmandu, Nepal                      37
2                  Beijing, China                      35
3                 Shanghai, China                      34
4       Ho Chi Minh City, Vietnam                      34

   Gross Rental Yield City Centre  Gross Rental Yield Outside of Centre  \
0                               5                                     3
1                               2                                     2
2                               2                                     2
3                               2                                     2
4                               3                                     3

   Price To Rent Ratio City Centre  \
0                               20
1                               60
2                               64
3                               62
4                               36

   Price To Rent Ratio Outside Of City Centre  \
0                                          37
1                                          53
2                                          58
3                                          64
4                                          29

   Mortgage As A Percentage Of Income  Affordability Index
0                                 694                    0
1                                 513                    0
2                                 249                    0
3                                 247                    0
4                                 382                    0
```

In [150…
```
# Summary statistics
df.describe()
```

Out[150…

	Price To Income Ratio	Gross Rental Yield City Centre	Gross Rental Yield Outside of Centre	Price To Rent Ratio City Centre	Price To Rent Ratio Outside Of City Centre	Mortgage As A Percentage Of Income	Affordability Index
count	281.0	281.0	281.0	281.0	281.0	281.0	281.0
mean	11.480427	5.814947	5.935943	22.651246	21.448399	126.715302	1.405694
std	6.906038	3.39031	3.114968	12.411375	11.221392	130.527354	1.179817
min	2.0	1.0	1.0	4.0	5.0	16.0	0.0
25%	7.0	4.0	4.0	14.0	14.0	56.0	1.0
50%	10.0	5.0	5.0	21.0	20.0	86.0	1.0
75%	15.0	7.0	7.0	28.0	26.0	155.0	2.0
max	37.0	22.0	20.0	80.0	84.0	1332.0	6.0

In [152…

```
# Check missing values
df.isnull().sum()
```

Out[152…

```
Rank                                    0
City                                    0
Price To Income Ratio                   0
Gross Rental Yield City Centre          0
Gross Rental Yield Outside of Centre    0
Price To Rent Ratio City Centre         0
Price To Rent Ratio Outside Of City Centre    0
Mortgage As A Percentage Of Income      0
Affordability Index                     0
dtype: int64
```

📊 Distribution of Numerical Features

🧠 Purpose

Understanding the distribution of numerical features helps identify the shape, spread, skewness, and presence of outliers in the data. It is crucial for choosing the right machine learning models and data transformation techniques.

🔍 What We Did

We plotted histograms with Kernel Density Estimates (KDE) for all numeric columns in the dataset. This includes:

- `Price To Income Ratio`
- `Gross Rental Yield (City Centre and Outside)`
- `Price To Rent Ratio (City Centre and Outside)`
- `Mortgage As A Percentage Of Income`
- `Affordability Index`

Each feature is visualized to see:

- If the distribution is normal, skewed, or multimodal.
- If extreme values (outliers) exist.
- How concentrated or dispersed values are.

📌 Observations

- Many features are **right-skewed**, such as `Mortgage As A Percentage Of Income`, suggesting a few cities have very high values.
- The `Affordability Index` is relatively **normally distributed**, which is ideal for modeling.
- Features like `Price To Rent Ratio` show a concentration toward lower values, implying housing is relatively more affordable in many cities.

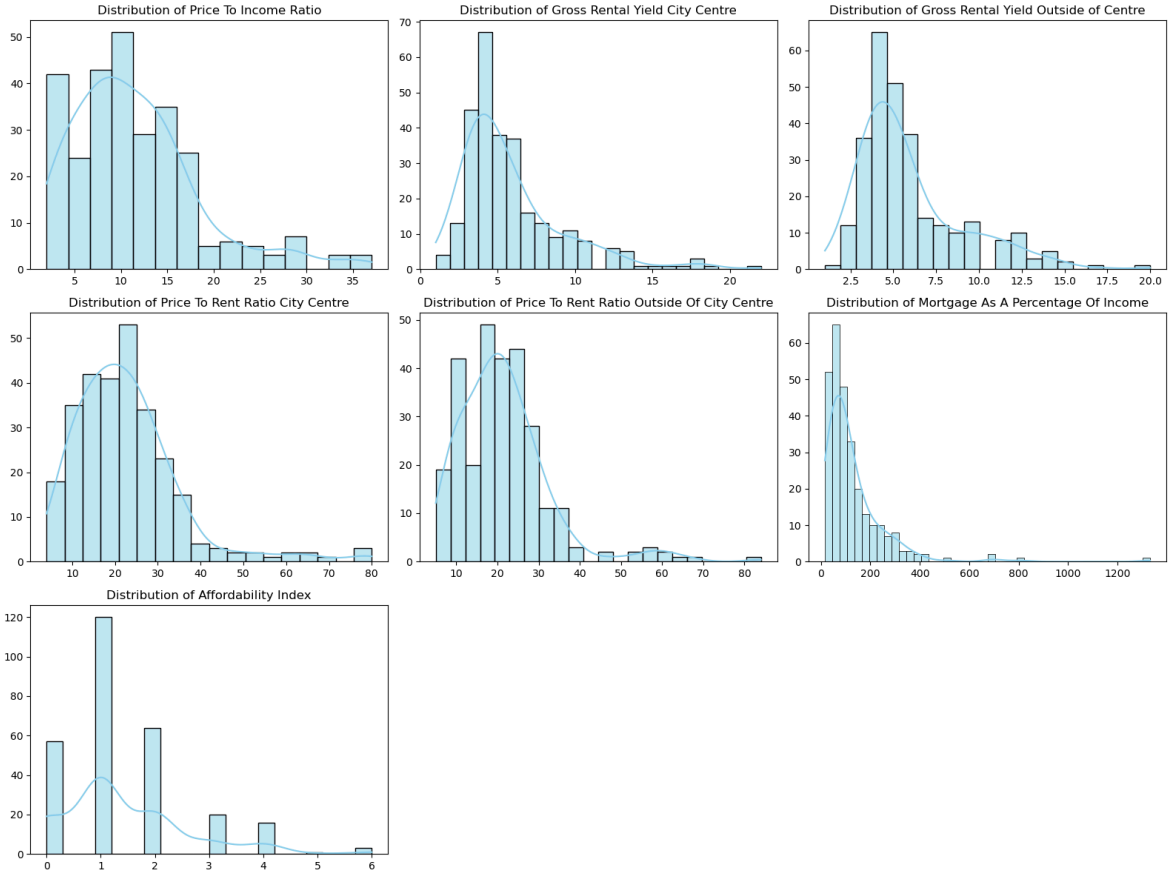
🛠️ Why It Matters

These insights help us:

- Decide on feature transformation (e.g., log-scaling for skewed data).
- Understand variability and inequality across global cities.
- Prepare the data for training classification models later.

In [155…

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

# Ensure your dataset is loaded and available as df

# Selecting only numerical columns
num_cols = df.select_dtypes(include=np.number).columns.tolist()

# Set up the figure
plt.figure(figsize=(16, 12))

# Plot histograms with KDE
for i, col in enumerate(num_cols):
    plt.subplot(3, 3, i + 1)
    sns.histplot(df[col].dropna(), kde=True, color='skyblue')
    plt.title(f'Distribution of {col}')
    plt.xlabel('')
    plt.ylabel('')

plt.tight_layout()
plt.show()
```



🔍 Correlation Heatmap

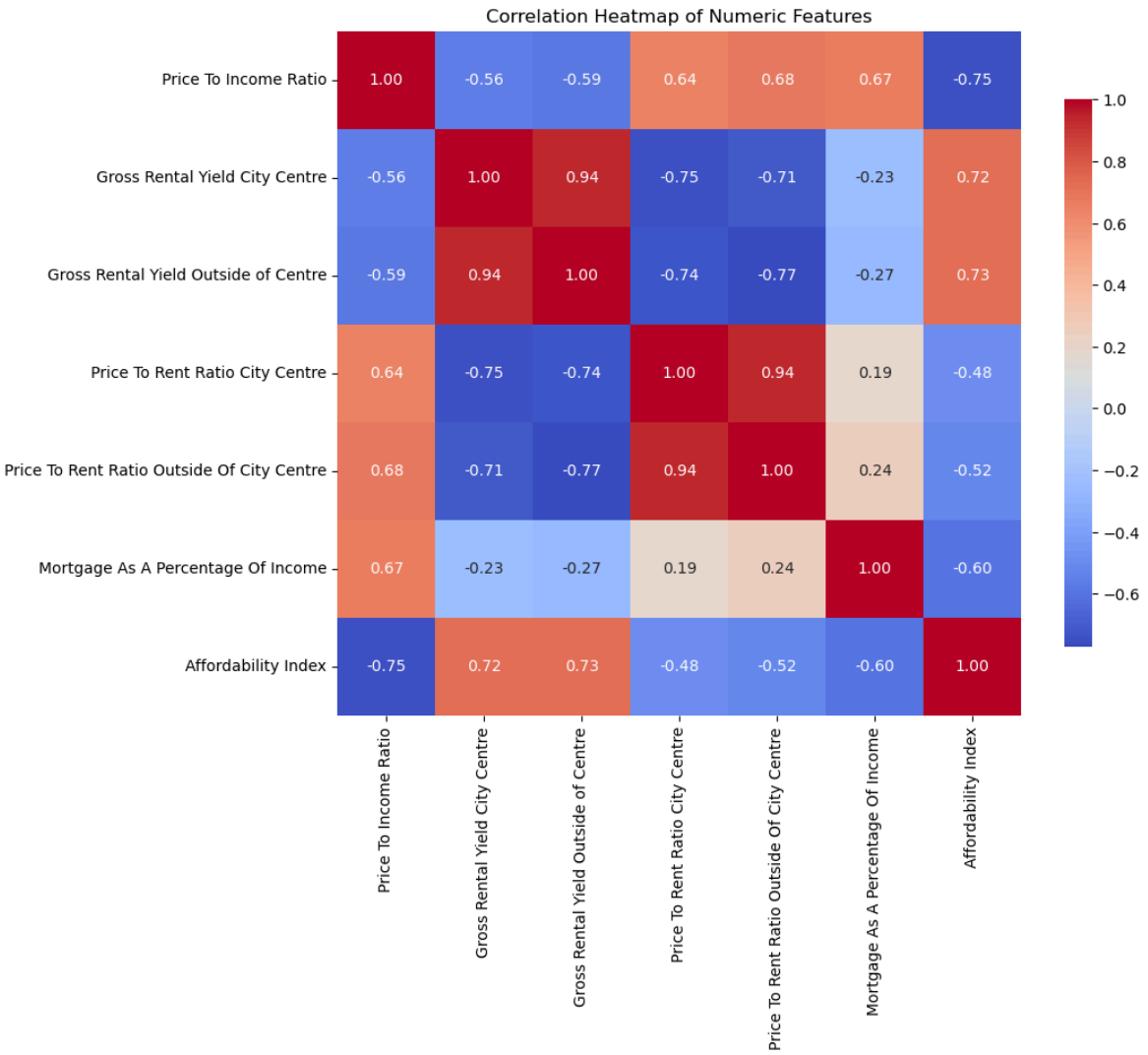
🔥 Correlation Heatmap

We use a correlation heatmap to understand how numerical features relate to one another.

- ✅ **Why**: To identify strong correlations that may influence affordability or cause multicollinearity in modeling.
- 🔍 **What it shows**: Brighter colors and higher absolute values (near 1 or -1) indicate stronger relationships.
- 📈 **How it helps**: Helps us detect redundant features or key drivers of affordability like *Mortgage as % of Income* or *Price to Income Ratio*.

In [165…

```
# Select only numeric columns for correlation
numeric_df = df.select_dtypes(include=[np.number])

# Compute correlation matrix
correlation_matrix = numeric_df.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", square=T
plt.title("Correlation Heatmap of Numeric Features")
plt.show()
```



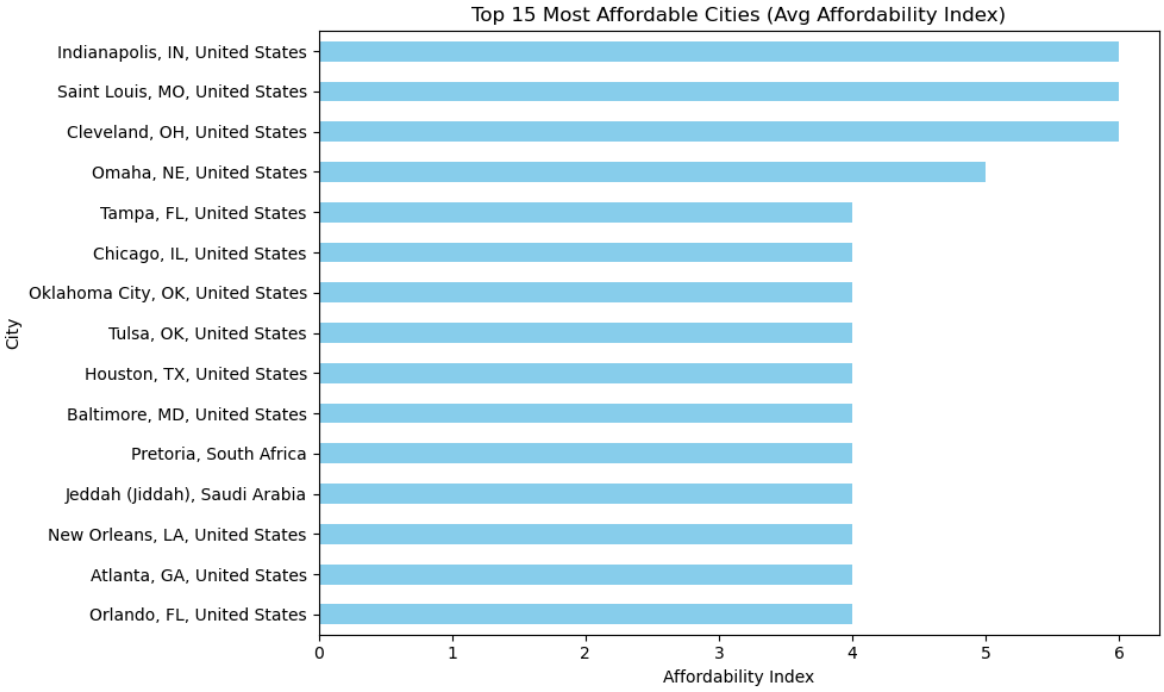
🏙️ 5. City-wise Affordability Overview here we are taking Top 15 most affordable cities by average index

This horizontal bar chart shows the top 15 cities with the highest average affordability.

- ✅ **Why**: To highlight which cities have more affordable housing markets based on the average Affordability Index.
- 🔍 **What it shows**: Cities at the top of the chart are considered more affordable for residents.
- 📈 **How it helps**: Useful for city-level comparisons, real estate investment decisions, or understanding affordability clusters.

In [169…

```
# Top 15 most affordable cities by average index
city_afford = df.groupby('City')['Affordability Index'].mean().sort_values(ascen

plt.figure(figsize=(10, 6))
city_afford.plot(kind='barh', color='skyblue')
plt.title("Top 15 Most Affordable Cities (Avg Affordability Index)")
plt.xlabel("Affordability Index")
plt.gca().invert_yaxis()  # highest on top
plt.tight_layout()
plt.show()
```



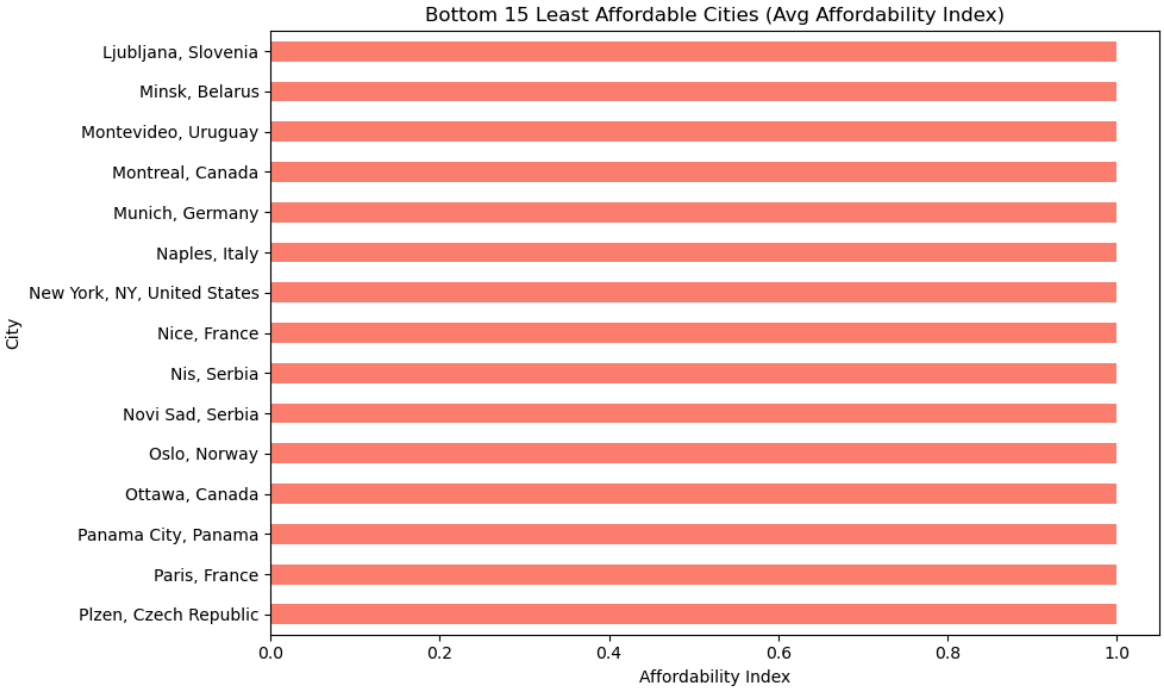
🏙️ Bottom 15 Least Affordable Cities by Average Affordability Index

This chart highlights the cities with the lowest affordability scores, indicating where housing is most expensive relative to income.

- ✅ **Why**: To identify areas where affordability is a concern.
- 🔍 **What it shows**: Cities where the Affordability Index is the lowest, often associated with high housing costs or low incomes.
- 📈 **How it helps**: Useful for understanding which regions struggle the most with affordable housing.

In [174…

```
# Calculate average Affordability Index per city
# Filter out zero or null affordability index entries
df_clean = df[df['Affordability Index'] > 0]

# Group and sort
bottom15 = df_clean.groupby('City')['Affordability Index'].mean().sort_values().

# Plot
bottom15.plot(kind='barh', color='salmon', figsize=(10, 6))
plt.title("Bottom 15 Least Affordable Cities (Avg Affordability Index)")
plt.xlabel("Affordability Index")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



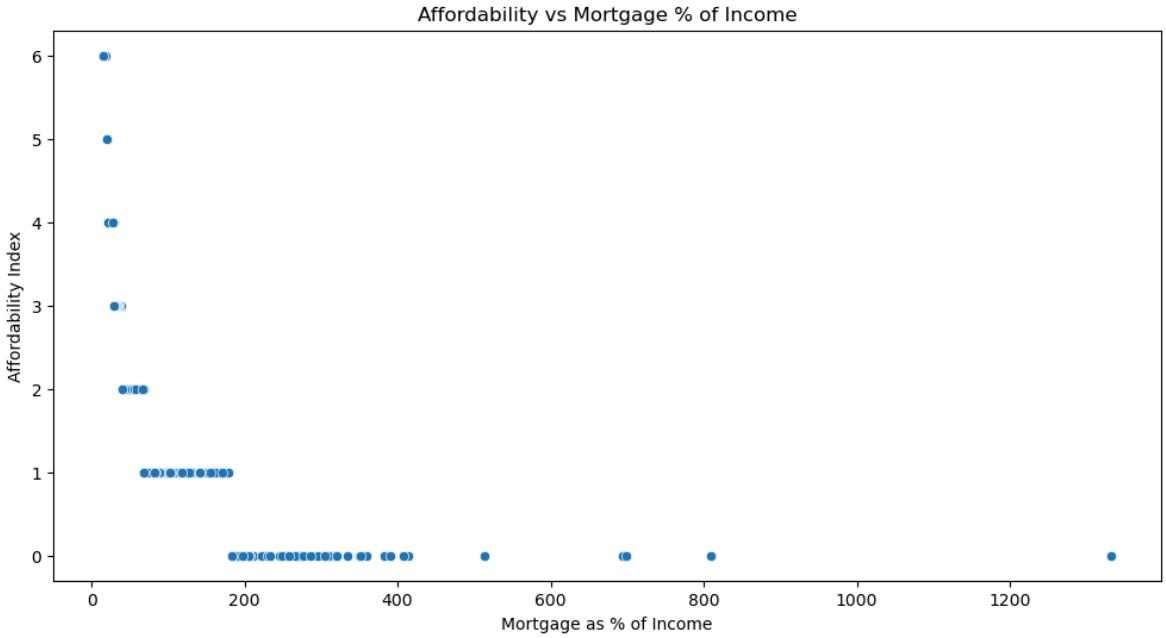
💵 Scatter Plot: Mortgage % of Income vs Affordability Index

We visualize the relationship between how much income goes to mortgage and the affordability index.

- ✅ **Why**: Mortgage burden is a direct indicator of housing stress.
- 🔍 **What it shows**: Typically, as mortgage % increases, affordability should decrease (negative correlation).
- 📈 **How it helps**: Helps determine if this variable is a good predictor for our classification model.

In [180…

```
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Mortgage As A Percentage Of Income', y='Affordability Index',
plt.title('Affordability vs Mortgage % of Income')
plt.xlabel('Mortgage as % of Income')
plt.ylabel('Affordability Index')
plt.show()
```



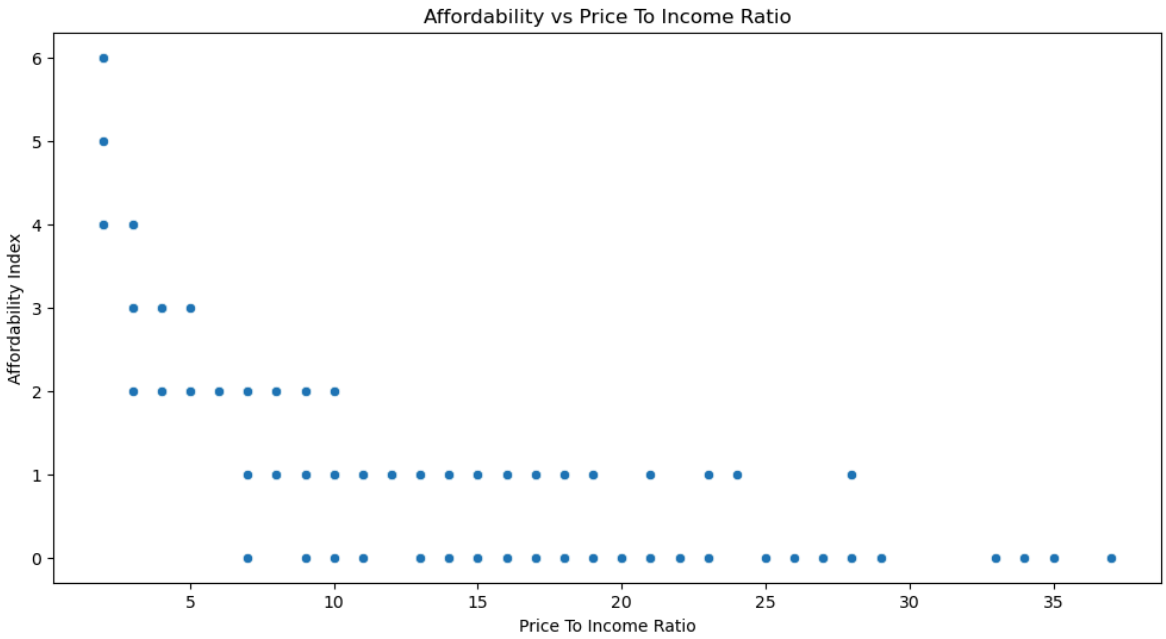
📊 Scatter Plot: Price to Income Ratio vs Affordability Index

This chart helps us see if higher home prices (relative to income) make a city less affordable.

- ✅ **Why**: This ratio is a standard affordability metric used globally.
- 🔍 **What it shows**: Often a negative relationship — higher price/income = lower affordability.
- 📈 **How it helps**: Helps evaluate predictive strength of this feature.

In [183…

```
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Price To Income Ratio', y='Affordability Index', data=df)
plt.title('Affordability vs Price To Income Ratio')
plt.xlabel('Price To Income Ratio')
plt.ylabel('Affordability Index')
plt.show()
```



🟢 7. Compare Affordable vs Not Affordable (Post Target Creation)

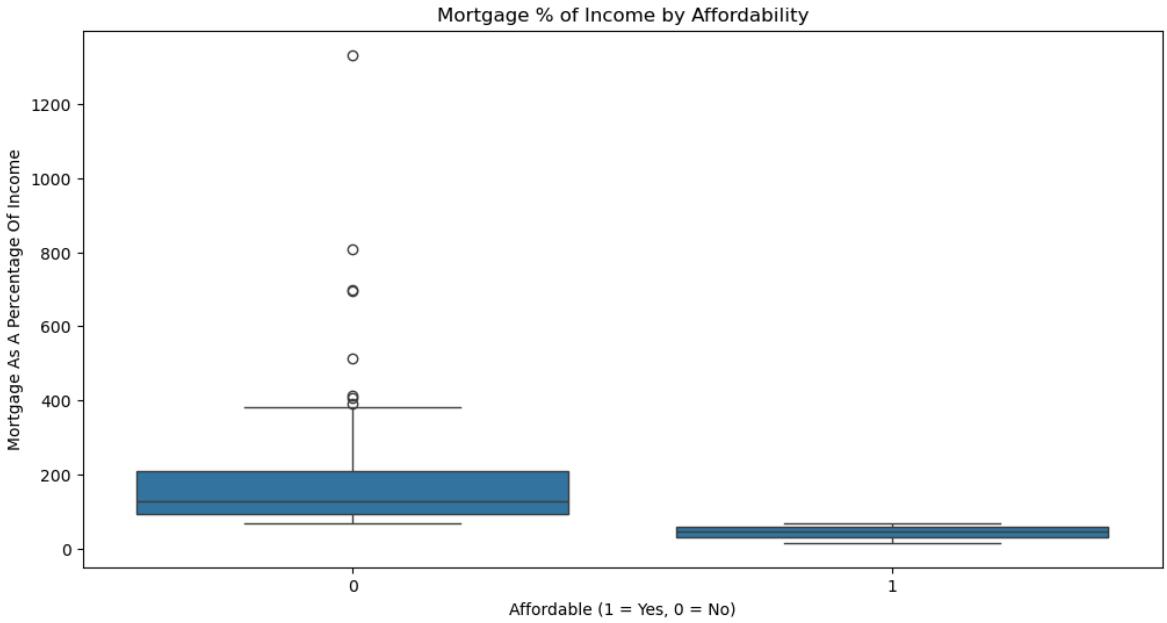
📦 Box Plot: Mortgage Burden Across Affordability Classes

A boxplot helps compare the distribution of mortgage % between affordable and non-affordable cities.

- ✅ **Why**: To visually compare distributions between two target classes.
- 🔍 **What it shows**: Non-affordable cities typically have higher mortgage % outliers and medians.
- 📈 **How it helps**: Confirms that mortgage % of income is a strong classifier in the model.

In [187…

```
# Create the target before this step (if not already)
df['Affordable'] = (df['Affordability Index'] > df['Affordability Index'].median

# Boxplot comparison
plt.figure(figsize=(12, 6))
sns.boxplot(x='Affordable', y='Mortgage As A Percentage Of Income', data=df)
plt.title('Mortgage % of Income by Affordability')
plt.xlabel('Affordable (1 = Yes, 0 = No)')
plt.show()
```



🔧 : Data Preprocessing & Feature Engineering

🔧 Preprocessing & Target Creation

- Drop unused columns like 'Rank'
- Create the binary target column: **Affordable (1) or Not Affordable (0)** using the median of `Affordability Index`

In [191…
```
# Drop Rank if not needed
df = df.drop(columns=['Rank'])
```

In [193…
```
# Create target variable
df['Affordable'] = (df['Affordability Index'] > df['Affordability Index'].median

# Drop 'City' and 'Affordability Index' if not needed for prediction
X = df.drop(columns=['City', 'Affordability Index', 'Affordable'])
y = df['Affordable']

# Check class balance
print(y.value_counts())
```

```
Affordable
0    177
1    104
Name: count, dtype: int64
```

📊: Feature Scaling

⚖️ Feature Scaling

Since the features are on different scales (ratios, percentages), we scale them using `StandardScaler`.

In [197…
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

🔀 : Train-Test Split

🔀 Split Data into Training and Testing Sets

We use 80% of the data for training and 20% for testing.

In [201…
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
```

🤖:Model Training & Evaluation

🤖 Try Multiple Classification Models

We train:

- Logistic Regression
- Random Forest
- Support Vector Machine (SVM)

In [205…
```
models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(random_state=42),
    'SVM': SVC()
}
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n📊 {name} Results:")
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

```
📊 Logistic Regression Results:
[[37  1]
 [ 1 18]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        38
           1       0.95      0.95      0.95        19

    accuracy                           0.96        57
   macro avg       0.96      0.96      0.96        57
weighted avg       0.96      0.96      0.96        57


📊 Random Forest Results:
[[38  0]
 [ 0 19]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        38
           1       1.00      1.00      1.00        19

    accuracy                           1.00        57
   macro avg       1.00      1.00      1.00        57
weighted avg       1.00      1.00      1.00        57


📊 SVM Results:
[[37  1]
 [ 1 18]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        38
           1       0.95      0.95      0.95        19

    accuracy                           0.96        57
   macro avg       0.96      0.96      0.96        57
weighted avg       0.96      0.96      0.96        57
```

📊 Feature Importance (Random Forest)

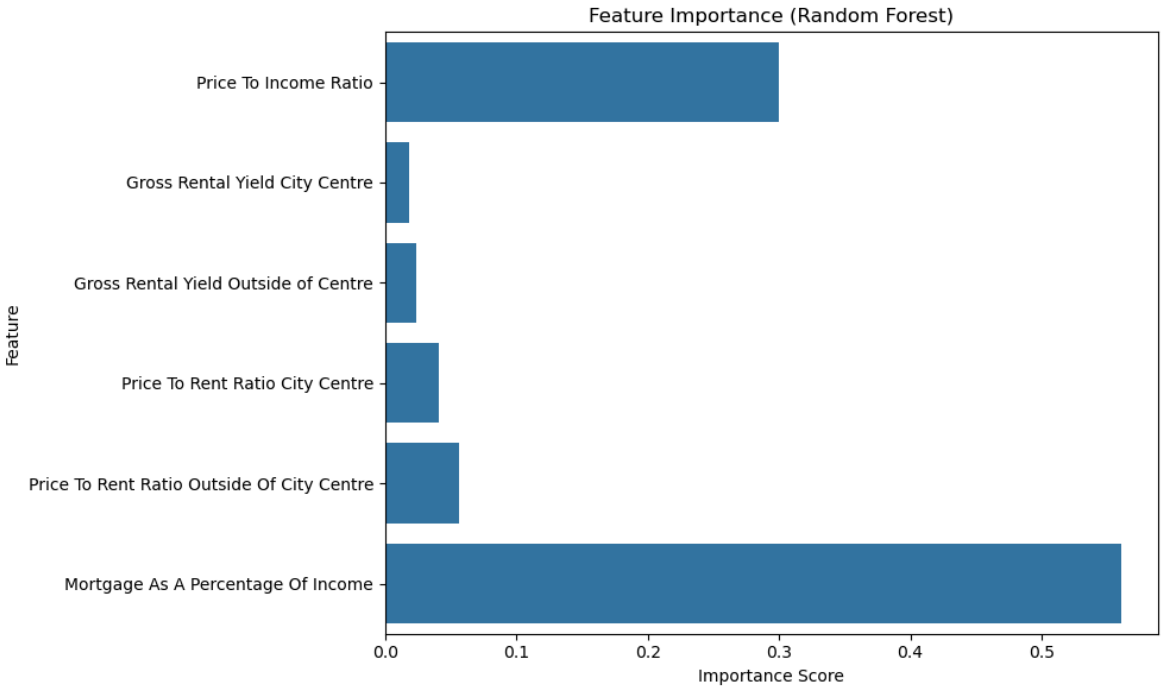
📌 Feature Importance

We visualize which features most influence the affordability classification using the Random Forest model.

In [209…

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

feature_importance = rf_model.feature_importances_
```

```
features = X.columns

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x=feature_importance, y=features)
plt.title("Feature Importance (Random Forest)")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```



🎯 Conclusion

✅ Conclusion

- We successfully classified cities as "Affordable" or "Not Affordable".
- Random Forest gave us a clear picture of feature influence:
  - Mortgage as % of Income and Price-to-Income Ratio are strong predictors.
- Such models can support investment decisions and housing policy analysis.

🚀 what we can do more:-

- Try hyperparameter tuning (GridSearchCV)
- Add more city-level data (crime rate, employment rate)
- Deploy the model using Flask or Streamlit for real-time use

In [ ]: