Lab 3: Feature Selection - Filter Method

R Abhijit Srivathsan

2448044

Loading California Housing Dataset from `sklearn`

In [13]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.feature_selection import VarianceThreshold

# Download the dataset as a pandas DataFrame.
data = fetch_california_housing(as_frame=True)
df = data.frame

# Separate features and target.
X = df.drop('MedHouseVal', axis=1)
y = df['MedHouseVal']

print("Feature names:", X.columns.tolist())
print("Data shape:", X.shape)
```

```
Feature names: ['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitude']
Data shape: (20640, 8)
```

In this lab, we load the California housing dataset from `sklearn` and apply three filter-based feature selection methods:

- **Variance Threshold**
- **Correlation Threshold**
- **Mean Absolute Deviation (MAD)**

Each section shows how adjusting the threshold affects the number of features retained.

Variance Threshold Filtering

The **Variance Threshold** method removes features with low variability. Here we demonstrate:

- The default threshold (0.0) which only removes constant features.
- A higher threshold (0.5) to remove features with variance lower than 0.5.

In [14]:
```
# (a) Using the default threshold (0.0) - removes only constant features.
vt_default = VarianceThreshold(threshold=0.0)
X_vt_default = vt_default.fit_transform(X)
print("[VarianceThreshold] With threshold=0.0, shape:", X_vt_default.shape)

# Check the variance of each feature to help decide on a threshold.
variances = X.var()
print("\nVariances of features:\n", variances)

# (b) Increasing the threshold to 0.5.
vt_higher = VarianceThreshold(threshold=0.5)
X_vt_higher = vt_higher.fit_transform(X)
print("\n[VarianceThreshold] With threshold=0.5, shape:", X_vt_higher.shape)

# Get the names of the features that pass the threshold.
selected_features_vt = X.columns[vt_higher.get_support()]
print("Features retained with VarianceThreshold (threshold=0.5):", selected_features_vt.tolist())
```

```
[VarianceThreshold] With threshold=0.0, shape: (20640, 8)

Variances of features:
 MedInc        3.609323e+00
HouseAge      1.583963e+02
AveRooms      6.121533e+00
AveBedrms     2.245915e-01
Population    1.282470e+06
AveOccup      1.078700e+02
Latitude      4.562293e+00
Longitude     4.014139e+00
dtype: float64

[VarianceThreshold] With threshold=0.5, shape: (20640, 7)
Features retained with VarianceThreshold (threshold=0.5): ['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitud
e', 'Longitude']
```

Correlation Threshold Filtering

The **Correlation Filter** examines pairwise correlations among features. If a pair of features has a correlation above a specified threshold, one of them is dropped.

- We first use a threshold of 0.8.
- Then, we use a stricter threshold of 0.6.

In [15]:
```python
def correlation_threshold_selector(X, threshold=0.8):
    """
    Identify features to drop based on correlation.

    Parameters:
      X         : DataFrame with features.
      threshold : float, correlation coefficient threshold.

    Returns:
      to_drop   : list of feature names to drop.
    """
    corr_matrix = X.corr().abs()
    # Use only the upper triangle of the correlation matrix.
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))

    # Identify columns that have any correlation above the threshold.
    to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
    return to_drop

# (a) Using a correlation threshold of 0.8.
to_drop_08 = correlation_threshold_selector(X, threshold=0.8)
print("[Correlation Filter] Features dropped (threshold=0.8):", to_drop_08)
selected_features_corr_08 = [col for col in X.columns if col not in to_drop_08]
print("Features retained (correlation threshold=0.8):", selected_features_corr_08)

# (b) Using a stricter threshold of 0.6.
to_drop_06 = correlation_threshold_selector(X, threshold=0.6)
print("\n[Correlation Filter] Features dropped (threshold=0.6):", to_drop_06)
selected_features_corr_06 = [col for col in X.columns if col not in to_drop_06]
print("Features retained (correlation threshold=0.6):", selected_features_corr_06)
```

```
[Correlation Filter] Features dropped (threshold=0.8): ['AveBedrms', 'Longitude']
Features retained (correlation threshold=0.8): ['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitude']

[Correlation Filter] Features dropped (threshold=0.6): ['AveBedrms', 'Longitude']
Features retained (correlation threshold=0.6): ['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitude']
```

Mean Absolute Deviation (MAD) Filtering

The **MAD Filter** removes features with low dispersion (spread). We:

- Calculate the MAD for each feature.
- Select features whose MAD is above a given threshold.
- Experiment with thresholds of 0.5 and 1.0.

In [16]:
```python
# Calculate the MAD for each feature manually.
mad_values = X.apply(lambda col: np.mean(np.abs(col - col.mean())))
print("Mean Absolute Deviation (MAD) for each feature:\n", mad_values)

def mad_threshold_selector(X, threshold):
    # Manually calculate MAD for each column.
    mad_values = X.apply(lambda col: np.mean(np.abs(col - col.mean())))
    selected_features = mad_values[mad_values >= threshold].index.tolist()
    return selected_features

# (a) Applying MAD threshold = 0.5.
selected_features_mad_05 = mad_threshold_selector(X, threshold=0.5)
print("\n[MAD Filter] Features retained (MAD threshold=0.5):", selected_features_mad_05)

# (b) Increasing the threshold to 1.0.
selected_features_mad_10 = mad_threshold_selector(X, threshold=1.0)
print("[MAD Filter] Features retained (MAD threshold=1.0):", selected_features_mad_10)
```

```
Mean Absolute Deviation (MAD) for each feature:
 MedInc          1.401614
HouseAge       10.551539
AveRooms        1.117619
AveBedrms       0.114093
Population    714.237277
AveOccup        0.748172
Latitude        1.975024
Longitude       1.830206
dtype: float64

[MAD Filter] Features retained (MAD threshold=0.5): ['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitude', 'L
ongitude']
[MAD Filter] Features retained (MAD threshold=1.0): ['MedInc', 'HouseAge', 'AveRooms', 'Population', 'Latitude', 'Longitude']
```

Mutual Information for Feature Selection

In this section, we use mutual information to assess the dependency between each feature and the target variable. Mutual information (MI) is a non-negative value that measures the amount of information obtained about one random variable (the target) through another random variable (the feature). A higher MI indicates a more informative feature.

For regression problems, we use `mutual_info_regression`. We then experiment with selecting features that have MI scores above a chosen threshold.

In [17]:
```python
from sklearn.feature_selection import mutual_info_regression

# Compute the mutual information scores between each feature and the target.
mi_scores = mutual_info_regression(X, y, random_state=0)
mi_series = pd.Series(mi_scores, index=X.columns)

print("Mutual Information Scores (unsorted):")
print(mi_series)

# Sort the scores in descending order for better visualization.
mi_series_sorted = mi_series.sort_values(ascending=False)
print("\nMutual Information Scores (sorted):")
print(mi_series_sorted)
```

```
Mutual Information Scores (unsorted):
MedInc        0.387646
HouseAge      0.032114
AveRooms      0.103623
AveBedrms     0.024482
Population    0.021994
AveOccup      0.072549
Latitude      0.370657
Longitude     0.401820
dtype: float64

Mutual Information Scores (sorted):
Longitude     0.401820
MedInc        0.387646
Latitude      0.370657
AveRooms      0.103623
AveOccup      0.072549
HouseAge      0.032114
AveBedrms     0.024482
Population    0.021994
dtype: float64
```

In [18]:

```python
# Define a threshold for selecting features.
# (The threshold value is somewhat arbitrary and may need tuning based on your dataset.)
mi_threshold = 0.1

# Select features that have a mutual information score greater than or equal to the threshold.
selected_features_mi = mi_series[mi_series >= mi_threshold].index.tolist()
print("\nFeatures selected with mutual information score >= {}:".format(mi_threshold))
print(selected_features_mi)
```

```
Features selected with mutual information score >= 0.1:
['MedInc', 'AveRooms', 'Latitude', 'Longitude']
```

Comprehensive Observations on Feature Selection Methods

1. Variance Threshold

- **Threshold = 0.0:**
  - All 8 features are retained since only features with zero variance would be removed.
  - **Dataset Shape:** (20640, 8)
- **Threshold = 0.5:**
  - The feature **`AveBedrms`** is removed because its variance (~0.2246) is below 0.5.
  - **Retained Features:** `['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitude', 'Longitude']`
  - **Dataset Shape:** (20640, 7)

2. Correlation Filter

- **Thresholds 0.8 and 0.6:**
  - Both thresholds resulted in the same outcome:
    - **Features Dropped:** `['AveBedrms', 'Longitude']`
    - **Retained Features:** `['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitude']`
- **Insight:**
  - **`AveBedrms`** is consistently identified as redundant or uninformative.
  - **`Longitude`** is removed due to its high correlation with other features—even though, as seen later, it carries significant predictive information according to mutual information analysis.

3. Mean Absolute Deviation (MAD) Filter

- **MAD Values for Each Feature:**
  - `MedInc` : 1.4016
  - `HouseAge` : 10.5515
  - `AveRooms` : 1.1176
  - `AveBedrms` : 0.1141

    - `Population` : 714.2373
    - `AveOccup` : 0.7482
    - `Latitude` : 1.9750
    - `Longitude` : 1.8302
- **Threshold = 0.5:**
    - Removes features with MAD below 0.5, leading to the removal of **`AveBedrms`**.
    - **Retained Features:** `['MedInc', 'HouseAge', 'AveRooms', 'Population', 'AveOccup', 'Latitude', 'Longitude']`
- **Threshold = 1.0:**
    - With a stricter threshold, features with MAD below 1.0 are removed; here, both **`AveBedrms`** and **`AveOccup`** (MAD ≈ 0.7482) are dropped.
    - **Retained Features:** `['MedInc', 'HouseAge', 'AveRooms', 'Population', 'Latitude', 'Longitude']`
- **Insight:**
    - The MAD filter highlights features with very low dispersion. **`AveBedrms`** consistently shows minimal variation, while **`AveOccup`** is borderline and only gets dropped at higher thresholds.

4. Mutual Information (MI) for Feature Selection

- **Mutual Information Scores (Unsorted):**
    - `MedInc` : 0.387646
    - `HouseAge` : 0.032114
    - `AveRooms` : 0.103623
    - `AveBedrms` : 0.024482
    - `Population` : 0.021994
    - `AveOccup` : 0.072549
    - `Latitude` : 0.370657
    - `Longitude` : 0.401820
- **Mutual Information Scores (Sorted Descending):**

    1. `Longitude` : 0.401820
    2. `MedInc` : 0.387646
    3. `Latitude` : 0.370657
    4. `AveRooms` : 0.103623
    5. `AveOccup` : 0.072549
    6. `HouseAge` : 0.032114
    7. `AveBedrms` : 0.024482
    8. `Population` : 0.021994
- **Feature Selection (Threshold = 0.1):**
    - **Selected Features:** `['MedInc', 'AveRooms', 'Latitude', 'Longitude']`
- **Insight:**
    - The MI analysis reveals that **geographical features** (`Longitude` and `Latitude`) and `MedInc` are highly informative.
    - `AveRooms` is marginal but still passes the threshold.
    - Lower MI scores for `HouseAge` , `AveBedrms` , `Population` , and `AveOccup` indicate they contribute less unique predictive information.

Overall Synthesis

- **Consistency:**
    - `AveBedrms` is consistently removed or deemed uninformative across all methods (variance, correlation, MAD, and MI), reinforcing its limited value.
- **Complementary Perspectives:**
    - **Variance and MAD filters** assess the spread and dispersion of the feature values.
    - The **Correlation filter** targets redundancy between features.
    - **Mutual Information** evaluates how much predictive power each feature contributes toward the target variable.
- **Notable Contrasts:**

  - Although the **Correlation filter** removes `Longitude` due to high inter-feature correlation, the **MI analysis** shows that `Longitude` is one of the most informative features, indicating that even highly correlated features can be valuable predictors.
- **Practical Implications:**
  - Combining these methods provides a more nuanced view of the data. It helps in balancing dimensionality reduction with the retention of informative features.
  - Careful threshold tuning is essential to ensure that the final feature set is both non-redundant and rich in predictive content.