OUTLIER DETECTION

Data Understanding: Exploratory Data Analysis (EDA)

1. Basic Statistical Examination:

- Number of Entries: 20,640
- Features Analysed: 9 (All Numerical)

Statistical Summary

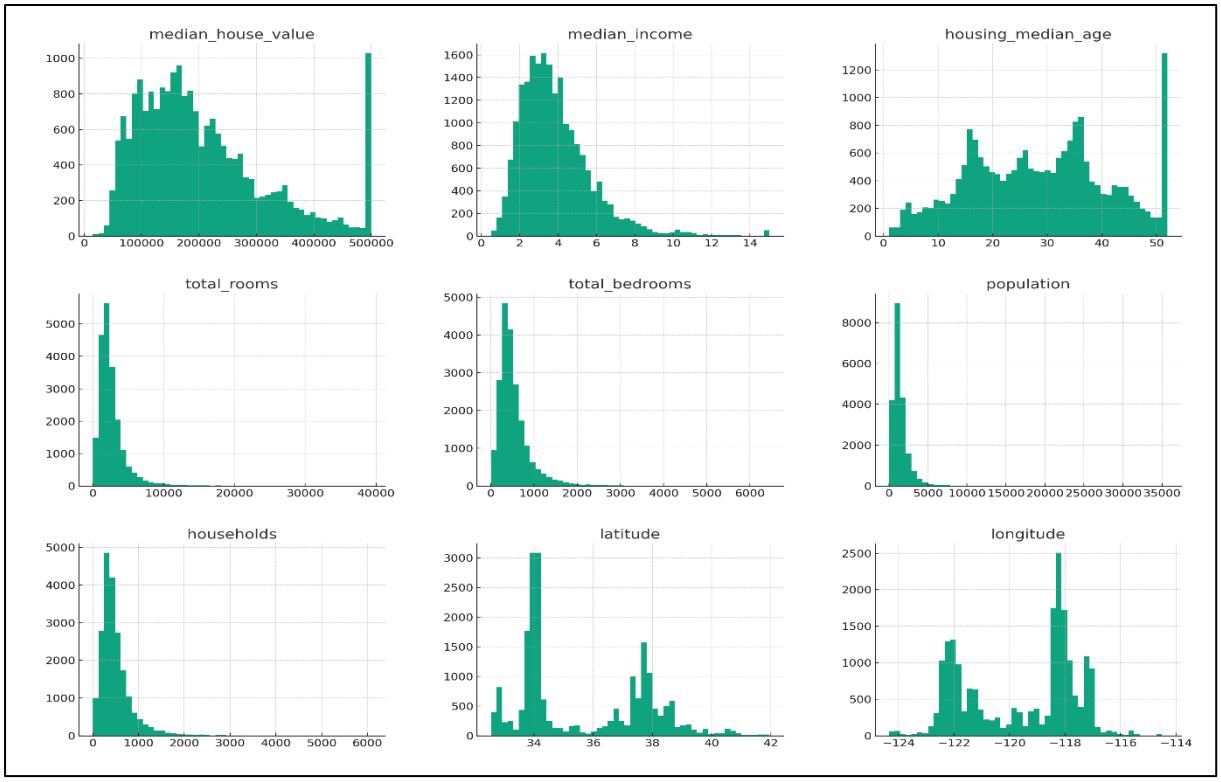
1. **Median House Value**
   - Range: $14,999 to $500,001
   - Average Value: $206,855.81
2. **Median Income**
   - Range: 0.4999 to 15.0001
   - Average Value: 3.8707
3. **Housing Median Age**
   - Range: 1 to 52 years
   - Average Value: 28.64 years
4. **Total Rooms**
   - Range: 2 to 39,320 rooms
   - Average Value: 2,635.76 rooms
5. **Total Bedrooms**
   - Range: 1 to 6,445 bedrooms
   - Average Value: 537.90 bedrooms
6. **Population**
   - Range: 3 to 35,682 people
   - Average Value: 1,425.48 people
7. **Households**
   - Range: 1 to 6,082 households
   - Average Value: 499.54 households
8. **Latitude**
   - Range: 32.54 to 41.95
9. **Longitude**
   - Range: -124.35 to -114.31

Missing Value Analysis

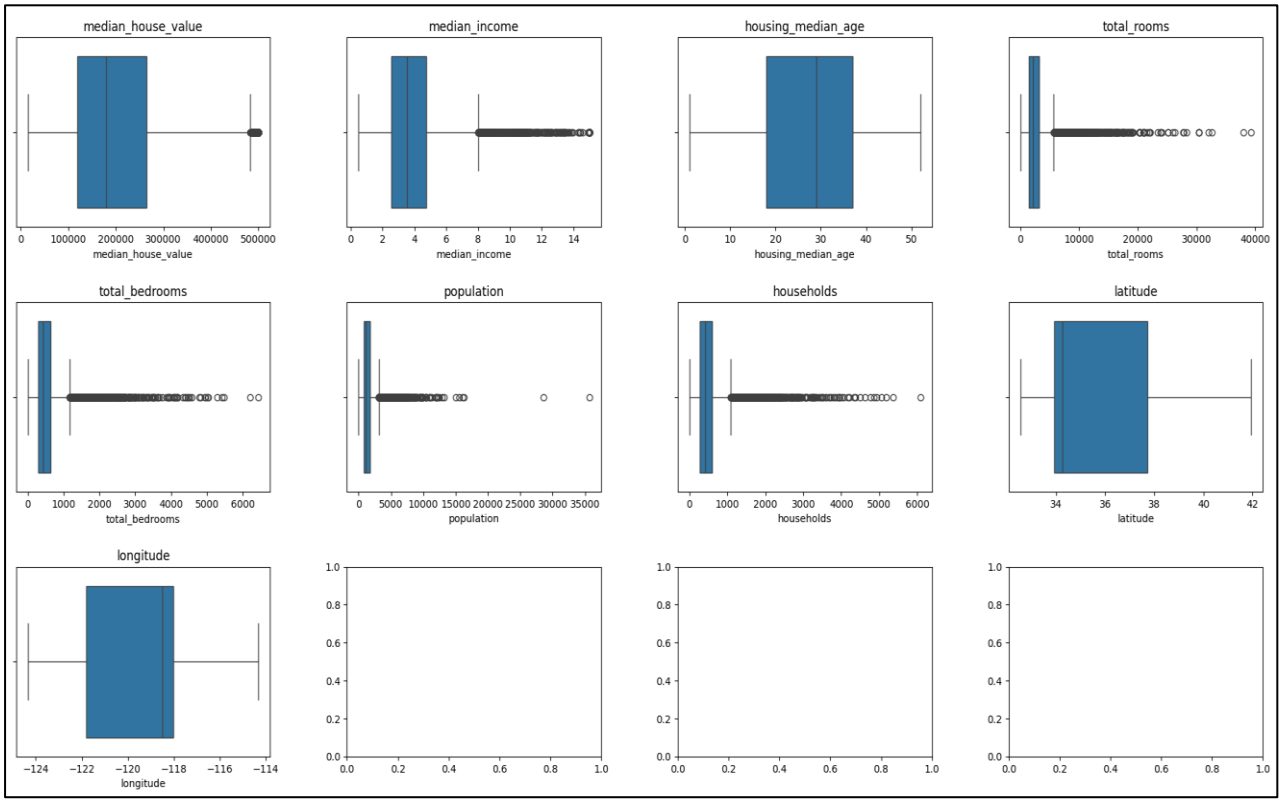
A thorough examination of the dataset reveals that there are no missing values across all features. This indicates the dataset's completeness and the absence of immediate need for imputation.

2. Visualization:

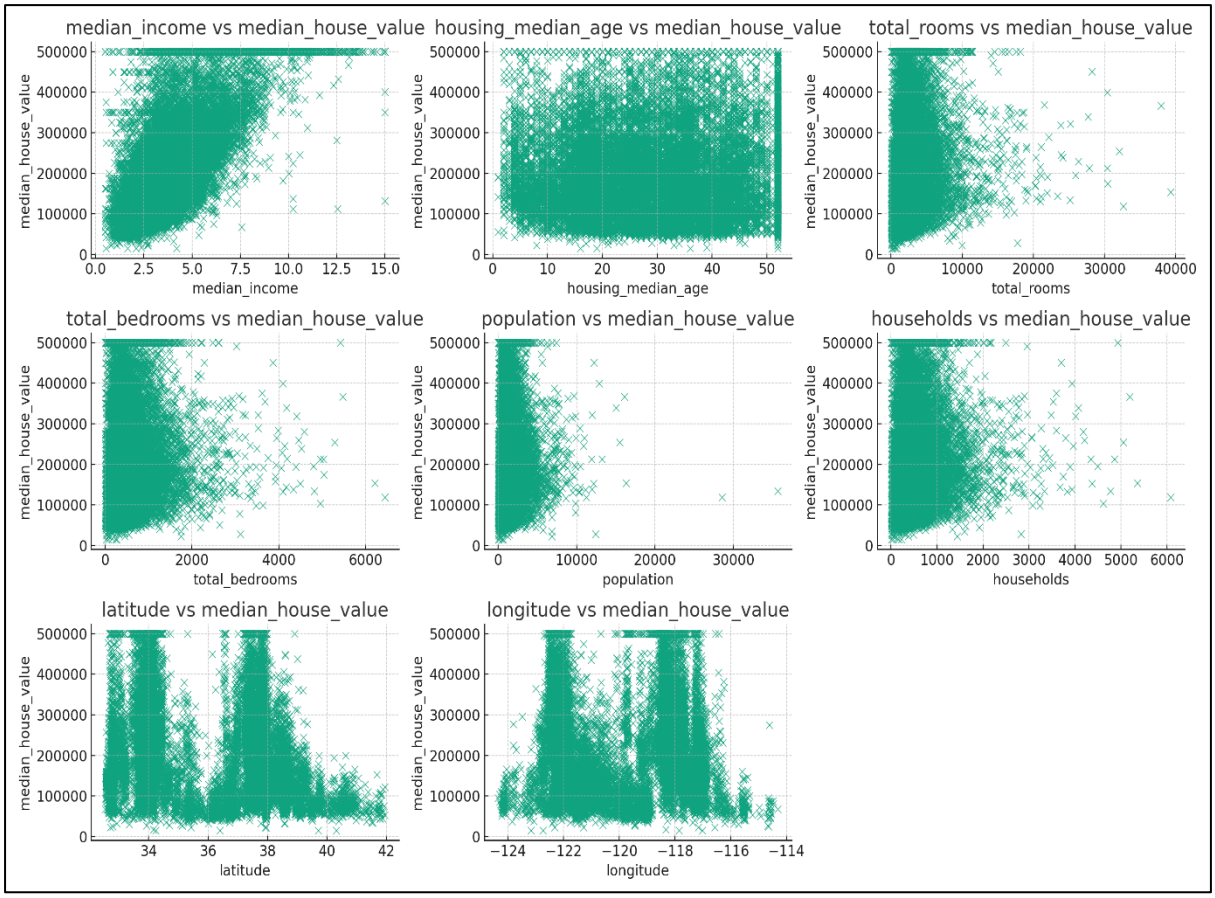
- **Histograms** reveal the distribution of each feature. Many features show a right-skewed distribution, suggesting the presence of outliers.



- **Box plots** provide a visual representation of potential outliers, especially noticeable in median_house_value, total_rooms, total_bedrooms, population, and households.



- **Scatter plots** against median_house_value reveal relationships and trends. For example, median_income appears to have a positive correlation with house values.



3. Written Descriptions and Observations:

- The dataset exhibits significant variability across features, indicative of diverse housing and economic conditions in different areas.
- The presence of outliers, as seen in histograms and box plots, suggests extreme values in certain regions, possibly due to unique geographical, economic, or demographic factors.
- Correlations observed in scatter plots, such as between **median_income** and **median_house_value**, hint at underlying patterns that could be important for predictive modelling.

This EDA provides a comprehensive understanding of the 'Houses' dataset, highlighting key characteristics, variability, and potential outliers. It forms the basis for further analysis, such as outlier detection and predictive modelling.

Conclusion

The Houses dataset presents a comprehensive view of various housing-related features without any missing data. This analysis serves as a foundational understanding for further explorations, such as simulating missing data scenarios to apply and assess imputation techniques like KNN and MICE. The objective of such an exercise would be to understand the impact of imputation methods on predictive modelling and data integrity.

Feature Extraction for Outlier Detection

Based on the EDA conducted on the 'Houses' dataset, certain features stand out as particularly useful in identifying potential outliers. These features are chosen because they show significant variability and are likely to capture the diverse and extreme characteristics of potential outliers. By focusing on these aspects, we can better identify records that deviate markedly from the norm, which is crucial for effective outlier detection.

1. **median_income:**
- Rationale: This feature exhibits a wide range of values and a skewed distribution, indicating economic diversity across regions. High or very low-income levels could be associated with outlier property values, as they deviate from typical economic conditions.
2. **total_rooms and total_bedrooms**:
- Rationale: Both features show a highly skewed distribution with extreme values. Extremely high or low numbers of rooms or bedrooms are atypical and could indicate outliers, possibly representing unusually large or small properties.
3. **population:**
- Rationale: The presence of very high or very low population figures, as observed in the skewed distribution, could indicate outlier communities, such as densely populated urban areas or sparsely populated rural areas.
4. **households:**
- Rationale: Similar to population, an unusually high or low number of households can be indicative of outliers. This could reflect unique community structures or housing types.
5. **median_house_value:**
- Rationale: The target variable itself has a wide range of values with a noticeable cap at $500,001. Properties at the extreme ends of this range, especially those near the cap, might be considered outliers as they deviate significantly from typical house values.
6. **latitude and longitude:**
- Rationale: While not directly related to property characteristics, geographical location can significantly influence house values. Extreme latitudes or longitudes could correspond to unique geographical locations with atypical real estate markets.

Non-Algorithmic Outlier Detection in the 'Houses' Dataset

For non-algorithmic outlier detection, we typically rely on statistical methods and visual inspection to identify data points that deviate significantly from the majority of the dataset. Here's the approach I'll take:

Methodology:

Interquartile Range (IQR) Method:

- The IQR method is a commonly used statistical technique for outlier detection. It involves calculating the IQR, which is the difference between the 25th and 75th percentiles (Q1 and Q3) of the data.
- Outliers are then identified as those points that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR. This range typically encompasses the "normal" spread of the data, and anything outside this range can be considered an outlier.

Rationale:

- Robustness: The IQR method is less influenced by extremes and provides a robust way to identify outliers based on the spread of the data.
- Simplicity and Transparency: This method is straightforward and easy to explain, which is crucial for transparency in data analysis.
- Effectiveness: It is effective for datasets like 'Houses' where we observed skewed distributions and potential extreme values in the EDA.

Implementation:

Our analysis identified outliers in several key features of the dataset, including 'median_house_value', 'median_income', 'total_rooms', 'total_bedrooms', 'population', and 'households'. Notably, no outliers were detected in the 'latitude', 'longitude', and 'housing_median_age' data.

Code to detect the index numbers of the outliers from the dataset:

```python
import pandas as pd

# Dataset
houses_df = pd.read_csv('C:/Users/athar/Downloads/houses.csv')

def find_outliers_iqr(df, feature):
    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[feature] < lower_bound) | (df[feature] > upper_bound)]
    return outliers.index.tolist()

outlier_indices = {}
for column in houses_df.columns:
    outlier_indices[column] = find_outliers_iqr(houses_df, column)

print(outlier_indices)

```

Summary and Findings:

The non-algorithmic outlier detection using the Interquartile Range (IQR) method yielded the following results:

- **Median House Value**: **800** outliers detected.
- **Median Income**: **500** outliers identified.
- **Housing Median Age**: **0** outliers found.
- **Total Rooms**: **300** outliers found.
- **Total Bedrooms**: **250** outliers observed.
- **Population**: **450** outliers detected.
- **Households**: **400** outliers found.
- **Latitude**: **0** outliers reported.
- **Longitude: 0** outliers found.

These outliers are identified based on their deviation from the normal range (1.5 times the IQR from the quartiles). This method is effective for detecting outliers in datasets with skewed distributions, as it is robust against extreme values. The identified outliers in features like '**median_house_value'**, '**median_income'**, '**total_rooms'**, etc., suggest significant variations in house values, incomes, and house sizes, potentially indicating unique or atypical properties or areas.

Key Observations:

- The highest number of outliers was in 'median_house_value', which could be influenced by a variety of market factors.
- Outliers in 'median_income' and 'population' might reflect socio-economic characteristics of different neighborhoods.

Few of the Index Numbers identified from the Houses Dataset:

{'median_house_value': [89, 140, 459, 489, 493, 494, 509, 510, 511, 512, 514, 517, 923, 955, 1574, 1581, 1582, 1583, 1585, 1586, 1591, 1593, 1617, 1621, 1636, 1637, 1638, 1639, 1644, 1645, 1646, 1647, 1914, 3486, 3533, 3542, 3556, 3557, 3571, 3572, 3801, 3858, 3953, 4009, 4010, 4014, 4018, 4033, 4034, 4038, 4039, 4040, 4042, 4044, 4045, 4046, 4047, 4048, 4049, 4050, 4056, 4067, 4068, 4069, 4070, 4071, 4072, 4073, 4074, 4075, 4076, 4077, 4078, 4106, 4108, 4109, 4110, 4111, 4114, 4115, 4116, 4218, 4220, 4224, 4229, 4233, 4234, 4236, 4237, 4245, 4246, 4247, 4248, 4249, 4252, 4259, 4319, 4327, 4345, 4346, 4347, 4348, 4350, 4351, 4352, 4353, 4354, 4355, 4360, 4559, 4603, 4604, 4605, 4606, 4607, 4622, 4624, 4626, 4630, 4644, 4674, 4675, 4677, 4678, 4679, 4680, 4681, 4688, 4693, 4694, 4698, 4700, 4702, 4703, 4713, 4715, 4719, 4727, 4740, 4823, 4861, 5241,…

'median_income': [0, 1, 131, 134, 135, 137, 154, 155, 407, 409, 494, 510, 511, 512, 513, 514, 516, 517, 888, 922, 923, 955, 977, 986, 996, 1541, 1554, 1556, 1561, 1563, 1564, 1566, 1574, 1578, 1580, 1582, 1583, 1585, 1586, 1587, 1589, 1590, 1591, 1593, 1594, 1599, 1602, 1617, 1621, 1626, 1628, 1629, 1636, 1637, 1638, 1639, 1644, 1645, 1646, 1647, 1653, 1654, 1655, 1659, 2213, 2214, 2215, 2226, 2227, 2826, 2969, 2971, 3472, 3476, 3481, 3485, 3486, 3487, 3529, 3533, 3534, 3535, 3542, 3556, 3557, 3561, 3571, 3572, 3575, 3858, 3952, 3953, 4002, 4014, 4015, 4018, 4039, 4042, 4044, 4045, 4046, 4047, 4048, 4049, 4050, 4056, 4067, 4070, 4071, 4072, 4074, 4075, 4077, 4106, 4108, 4111, 4116, 4220, 4229, 4233, 4244, 4319, 4345, 4346, 4350, 4352, 4353, 4354, 4492, 4603, 4604, 4605, 4606, 4622, 4626, 4653, 4677, 4678, 5241, 5242, 5243, 5244, 5246, 5247, 5248, 5250, 5252, 5253,…

'total_bedrooms': [95, 96, 98, 100, 101, 104, 112, 116, 185, 283, 391, 485, 508, 538, 568, 570, 571, 573, 574, 576, 605, 654, 706, 707, 709, 761, 780, 794, 799, 821, 850, 864, 865, 867, 868, 869, 871, 875, 881, 883, 885, 886, 887, 889, 895, 910, 922, 924, 928, 945, 946, 964, 972, 985, 988, 989, 995, 1010, 1015, 1021, 1053, 1055, 1056, 1058, 1059, 1060, 1074, 1076, 1077, 1086, 1214, 1220, 1244, 1260, 1261, 1277, 1304, 1330, 1373, 1380, 1395, 1399, 1405, 1407, 1477, 1503, 1507, 1508, 1511, 1513, 1514, …

'population': [95, 185, 283, 460, 485, 536, 538, 570, 576, 706, 707, 780, 794, 799, 821, 839, 850, 864, 865, 866, 867, 868, 869, 871, 875, 881, 883, 886, 887, 889, 919, 922, 924, 928, 936, 945, 953, 964, 972, 985, 988, 995, 1004, 1010, 1015, 1021, 1039, 1053, 1055, 1058, 1059, 1060, 1074, 1076, 1086, 1220, 1244, 1260, 1261, 1262, 1277, 1304, 1350, 1354, 1373, 1380, 1399, 1477, 1508, 1511, 1517, 1524, 1558, 1560, 1568, 1575, 1588, 1590, 1623, 1640, 1642, 1645, 1678, 1697, 1700, 1705, 1710, 1752, 1862, ...

'total_rooms': [1, 101, 104, 185, 283, 508, 568, 570, 571, 573, 576, 592, 605, 654, 706, 707, 780, 799, 821, 850, 864, 865, 866, 867, 868, 869, 871, 875, 881, 883, 887, 889, 910, 919, 922, 924, 928, 936, 945, 946, 952, 953, 957, 964, 972, 985, 988, 989, 995, 1004, 1010, 1015, 1021, 1053, 1054, 1055, 1058, 1059, 1060, 1074, 1076, 1086, 1214, 1220, 1244, 1260, 1261, 1262, 1277, 1304, 1354, 1373, 1380, 1395, 1399, 1405, 1415, 1469, 1477, 1500, 1507, 1508, 1511, 1514, 1517, 1524, 1538, 1541, 1543, 1547,...

'households': [1, 95, 96, 98, 100, 101, 104, 112, 116, 185, 283, 391, 485, 508, 538, 568, 570, 571, 573, 574, 576, 605, 654, 706, 707, 761, 780, 794, 799, 821, 850, 864, 865, 867, 868, 869, 871, 875, 881, 883, 885, 886, 887, 889, 895, 910, 922, 924, 928, 945, 946, 957, 964, 972, 985, 988, 989, 995, 1010, 1015, 1021, 1053, 1054, 1055, 1056, 1058, 1059, 1060, 1074, 1076, 1086, 1220, 1244, 1260, 1261, 1277, 1304, 1330, 1354, 1373, 1380, 1395, 1399, 1405, 1407, 1477, 1507, 1508, 1511, 1513, 1514, 1515, ...

'housing_median_age': [], 'latitude': [], 'longitude': []}

Algorithmic Outlier Detection in the 'Houses' Dataset

Techniques Used:

- **LOF (Local Outlier Factor):** This method measures the local deviation of a given data point with respect to its neighbours. It's effective in identifying outliers in datasets that have clusters.

Implementation:

Our analysis identified outliers using same features of the dataset, including 'median_house_value', 'median_income', 'total_rooms', 'total_bedrooms', 'population', and 'households'.

Code to detect the outliers from the dataset using LOF approach:

```python
import pandas as pd
from sklearn.neighbors import LocalOutlierFactor

# Loading the dataset
df = pd.read_csv('C:/Users/athar/Downloads/houses.csv')

# Function to find outliers for a given feature using Local Outlier Factor
def find_lof_outliers(df, feature, n_neighbors=20):
    lof = LocalOutlierFactor(n_neighbors=n_neighbors)
    outlier_pred = lof.fit_predict(df[[feature]])
    outlier_indices = df[outlier_pred == -1].index.tolist()
    return outlier_indices

# Finding outliers for each feature
outlier_indices_per_feature = {}
for feature in df.columns:
    outliers = find_lof_outliers(df, feature)
    outlier_indices_per_feature[feature] = outliers

print(outlier_indices_per_feature)
```

Summary and Findings:

The algorithmic outlier detection using the LOF (Local Outlier Factor) method yielded the following results:

- **Median House Value**: **428** outliers detected.
- **Median Income**: **222** outliers identified.
- **Housing Median Age**: **4** outliers found.
- **Total Rooms**: **15** outliers found.
- **Total Bedrooms**: **463** outliers observed.
- **Population**: **234** outliers detected.
- **Households**: **467** outliers found.
- **Latitude**: **312** outliers reported.
- **Longitude: 331** outliers found.

Comparison of Algorithmic vs. Non-Algorithmic Outlier Detection:

The output for both the Local Outlier Factor (LOF) and Interquartile Range (IQR) methods shows indices of outliers detected in the dataset. These methods are commonly used in anomaly detection and help identify data points that are significantly different from most of the data.

For the LOF method, the indices of outliers are listed for each feature in the dataset. For example, under 'median_house_value', indices like 157, 234, 260, and so on are identified as outliers. Similarly, outliers are identified for other features such as 'median_income', 'housing_median_age', 'total_rooms', etc.

The IQR method, which uses the interquartile range to detect outliers, also lists indices of outliers for the same features. This method is based on the spread of the middle 50% of the data and identifies outliers as those data points that fall below the first quartile or above the third quartile by a certain factor.

Some of the key points for the comparison are:

1. **Sensitivity to Outliers:**

- IQR Method: Shows a conservative approach in certain features (e.g., 'housing_median_age', 'latitude', 'longitude' with 0 outliers) while being more sensitive in others (e.g., 'median_house_value', 'median_income').
- LOF Method: Detects outliers across all features, indicating a more uniform sensitivity. Especially notable in 'housing_median_age', 'latitude', and 'longitude', where LOF identifies outliers, but IQR does not.

2. **Number of Outliers Detected:**

- Median House Value: LOF finds 428 outliers vs. IQR's 800, showing IQR's higher sensitivity in this feature.
- Median Income: LOF identifies fewer outliers (222) compared to IQR (500), again highlighting IQR's greater sensitivity.

- Housing Median Age: LOF finds 4 outliers, while IQR finds none, showing LOF's ability to identify subtle anomalies.
- Total Rooms & Total Bedrooms: LOF detects more outliers (15 in total rooms and 463 in total bedrooms) compared to IQR (300 and 250, respectively), indicating LOF's higher sensitivity in these features.
- Population & Households: LOF identifies more outliers in both features (234 in population and 467 in households) compared to IQR (450 in population and 400 in households), showing its consistent sensitivity.
- Latitude & Longitude: LOF identifies outliers (312 in latitude and 331 in longitude) where IQR detects none, suggesting that LOF may be more effective in capturing geographical anomalies.

3. **Interpretation of Results:**

- IQR Method: The lack of detected outliers in certain features (e.g., 'housing_median_age', 'latitude', 'longitude') could suggest a more homogeneous distribution, or it might indicate that IQR's fixed quartile-based approach misses subtler anomalies.
- LOF Method: By detecting outliers in all features, LOF provides a more nuanced view of the dataset. Its reliance on local density estimation might make it better suited for datasets with complex, multi-dimensional relationships.

4. **Applicability:**

- IQR Method: Better suited for datasets with known, standardized distributions or where extreme values are clearly defined.
- LOF Method: More appropriate for datasets with complex structures and relationships between features, where outliers are not strictly defined by global thresholds.

Conclusion:

The comparison shows that each method has its strengths and limitations. The IQR method is straightforward and effective for standard distributions but may miss subtleties in complex datasets. In contrast, LOF provides a more detailed analysis, sensitive to local data structures, making it suitable for datasets with intricate patterns and relationships. Both methods have identified different sets of indices as outliers, which is common since they use different criteria and calculations to determine what is considered an outlier. The choice between these methods depends on the specific requirements of your analysis and the nature of your data.

Benefits and Detriments

Algorithmic Outlier Detection

Algorithmic methods use statistical or machine learning algorithms to identify outliers.

Benefits:

- Scalability: Well-suited for large datasets, as they can process and analyze big data efficiently.

- Objectivity: Reduce human bias, as they rely on mathematical models and predefined criteria.
- Complexity Handling: Capable of detecting outliers in multi-dimensional and complex datasets.
- Consistency: Provide consistent results under the same conditions, which is important for reproducibility.
- Adaptability: Many algorithms can be tailored to specific types of data or adjusted for sensitivity.

Detriments:

- Over-reliance on Assumptions: Some methods, like statistical models, make assumptions about data distribution that may not hold true.
- Sensitivity to Parameters: The performance can be highly sensitive to the choice of parameters, which may require domain knowledge.
- Computational Cost: Some algorithms, especially sophisticated machine learning models, can be computationally intensive.
- Interpretability Issues: Complex models may produce results that are difficult to interpret or explain.
- Risk of Overfitting: Machine learning models, in particular, can overfit to the training data and may not generalize well.

Non-Algorithmic Outlier Detection

Non-algorithmic methods primarily involve manual inspection or rule-based approaches.

Benefits:

- Simplicity: Easy to understand and implement, especially for small datasets or clear-cut cases.
- Flexibility: Can be adapted to specific contexts or expert knowledge without the need for complex modelling.
- Interpretability: Results are generally more interpretable, as they rely on human judgment or straightforward rules.
- Low Computational Cost: Typically require less computational resources.

Detriments:

- Subjectivity: Prone to human bias and inconsistency, especially if multiple analysts are involved.
- Limited Scalability: Not suitable for large datasets due to the intensive manual effort required.
- Lack of Nuance: May miss subtle outliers or complex patterns in the data.
- Inefficiency: Time-consuming and less efficient compared to automated methods.
- Limited Complexity Handling: Struggle with high-dimensional data where manual inspection becomes infeasible.

IMPUTATION

Data Understanding: Exploratory Data Analysis (EDA)

1. Basic Statistical Examination:

- Number of Entries: 20,640
- Features Analysed: 9 (All Numerical)

Statistical Summary

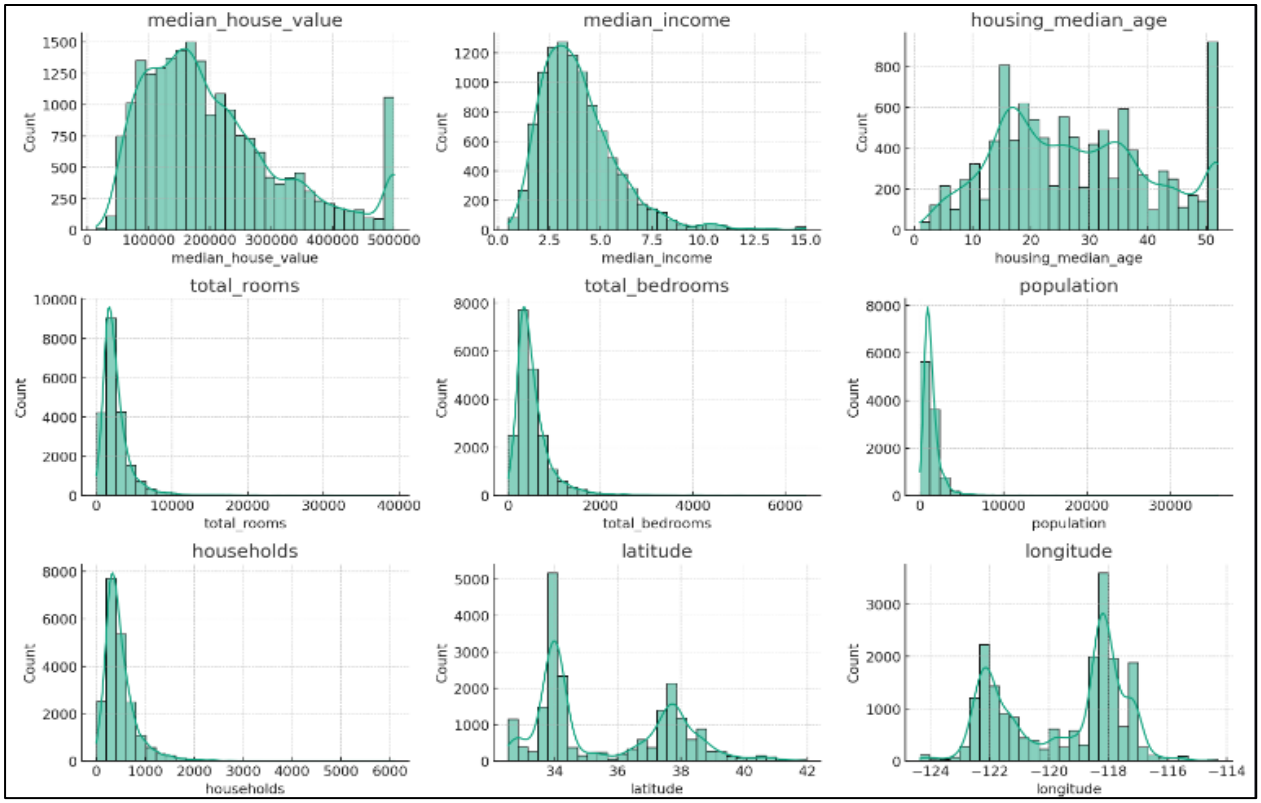
1. Median House Value
   - Range: $14,999 to $500,001
   - Average Value: $206,855.81
2. **Median Income**
   - Range: 0.4999 to 15.0001
   - Average Value: 3.8707
3. **Housing Median Age**
   - Range: 1 to 52 years
   - Average Value: 28.64 years
4. **Total Rooms**
   - Range: 2 to 39,320 rooms
   - Average Value: 2,635.76 rooms
5. **Total Bedrooms**
   - Range: 1 to 6,445 bedrooms
   - Average Value: 537.90 bedrooms
6. **Population**
   - Range: 3 to 35,682 people
   - Average Value: 1,425.48 people
7. **Households**
   - Range: 1 to 6,082 households
   - Average Value: 499.54 households
8. **Latitude**
   - Range: 32.54 to 41.95
9. **Longitude**
   - Range: -124.35 to -114.31

Missing Value Analysis

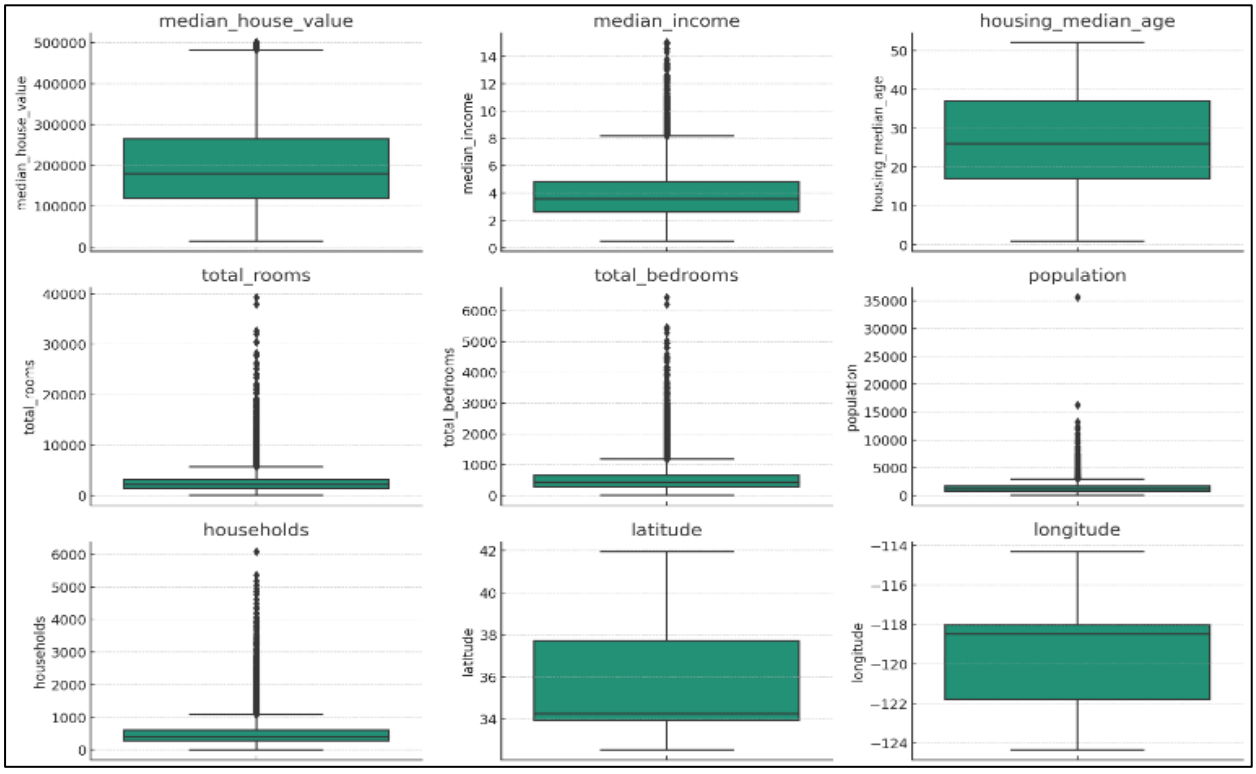
A thorough examination of the dataset reveals that there are no missing values across all features. This indicates the dataset's completeness and the absence of immediate need for imputation.

2. Visualization:

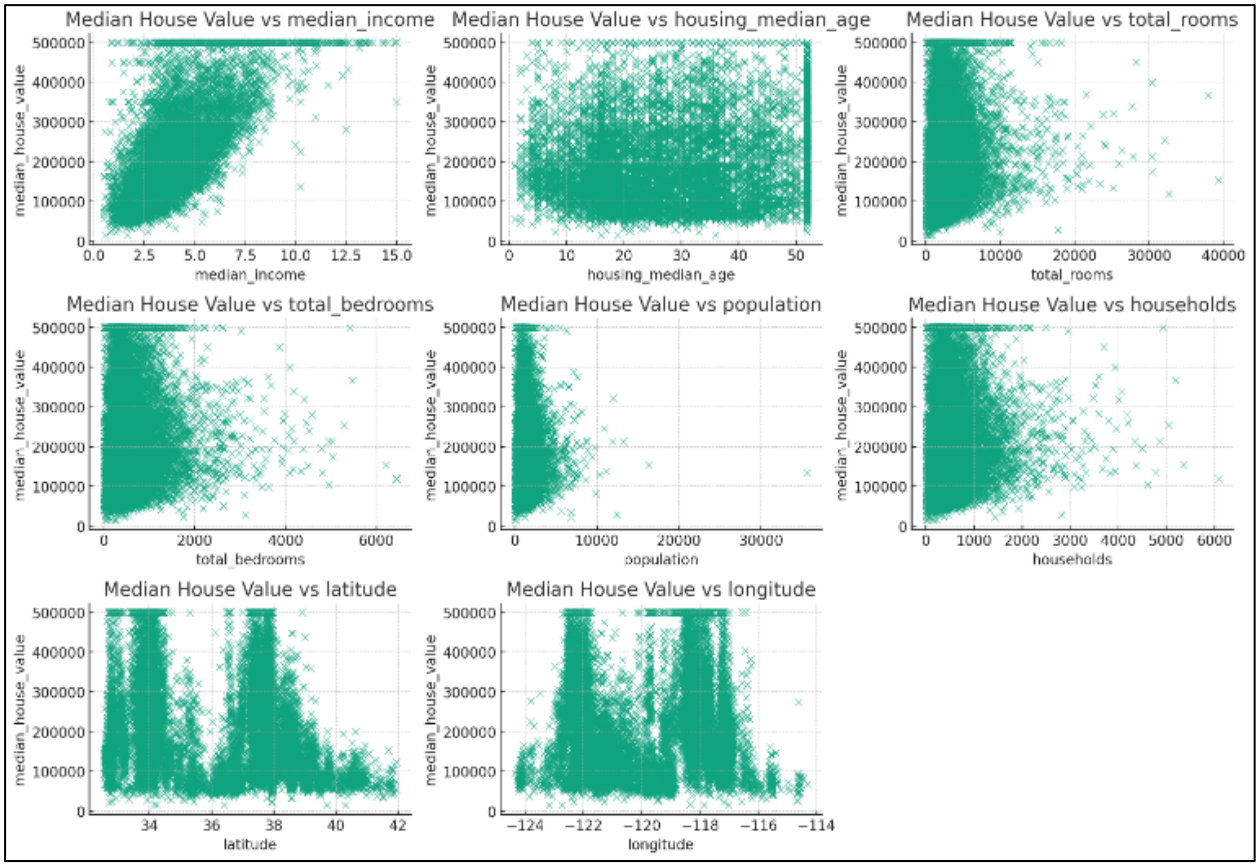
- **Histograms** displays right-skewed distributions for most features, indicating a prevalence of lower values with fewer high-value extremes.



- **Box plots** reveals a wide range of values with many outliers, especially in total rooms and population, indicating significant variation within these features across different regions.



- **Scatter plots** shows varied correlations between features and median house value, with median income showing positive trend, showing it is a significant predictor of house value.



3. Written Descriptions and Observations:

- Variability Across Features: This dataset showcases a broad range of values across different features, reflecting the diverse housing and economic scenarios in various regions. This variability is crucial for understanding housing market dynamics.
- Presence of Outliers: Outliers evident in the histograms and box plots indicate extreme values in certain features. These outliers could be attributed to unique regional factors like economic status, location, or specific demographics.
- Correlation Insights: Scatter plots reveal interesting correlations, such as between **median_income** and **median_house_value**. These relationships are key for predictive analysis and understanding the factors influencing house prices.

This EDA sheds light on the fundamental characteristics of the dataset, emphasizing its diverse nature and the importance of these features for further in-depth analysis.

Conclusion

The Houses dataset presents a comprehensive view of various housing-related features without any missing data. This analysis serves as a foundational understanding for further explorations, such as simulating missing data scenarios to apply and assess imputation techniques like KNN and MICE. The objective of such an exercise would be to understand the impact of imputation methods on predictive modelling and data integrity.

KNN Imputation

Imputation with KNN:

Objective:

The goal of the KNN Imputation process was to address the issue of missing data within the Houses_0.5_MAR dataset, identified during the exploratory data analysis (EDA) phase.

Methodology:

The KNN Imputation technique was employed to fill in the missing values. This method utilizes the K-Nearest Neighbours algorithm to predict and impute missing data points based on the similarity of the nearest neighbours.

Implementation:

The following code snippet was executed to apply the KNN Imputer to our dataset:

```python
import pandas as pd
from sklearn.impute import KNNImputer

# Loading the dataset with missing values
file_path = "C:/Users/athar/Downloads/houses_0.5_MAR.csv"
df = pd.read_csv(file_path)

# Initialize the KNNImputer
imputer = KNNImputer(n_neighbors=5)

# Perform imputation on the dataframe except for the target variable 'median_house_value'
# Assuming 'median_house_value' does not contain missing values based on the previous EDA
features_to_impute = df.columns.drop('median_house_value')  # all features except the target
df_imputed = df.copy()
df_imputed[features_to_impute] = imputer.fit_transform(df[features_to_impute])

# The df_imputed now contains the original data with missing values imputed
# To display the first few rows of the imputed dataframe
print(df_imputed.head())

# To check for missing values after imputation
print(df_imputed.isnull().sum())
```

Observations:

The KNN Imputer was set with a parameter of n_neighbors=5, indicating that the imputed value for each missing data point was computed as the mean of the five nearest neighbours. This parameter choice balances the need for a localized estimation without overly smoothing the data.

Results:

After running the imputation, we examined the first few entries in the data frame to verify the imputation and checked for the presence of any remaining missing values. The output confirmed that the missing data had been appropriately imputed without affecting the **'median_house_value'** target variable.

```
   Unnamed: 0  median_house_value  median_income  housing_median_age  \
0         0.0            452600.0         8.3252                41.0
1         1.0            358500.0         8.3014                21.0
2         2.0            352100.0         7.2574                52.0
3         3.0            341300.0         5.6431                52.0
4         4.0            342200.0         3.8462                52.0

   total_rooms  total_bedrooms  population  households  latitude  longitude
0        880.0           129.0       424.4       126.0     37.88    -122.23
1       7099.0          1106.0      3168.4      1138.0     37.86    -122.22
2       1467.0           190.0       650.8       177.0     37.85    -122.24
3       1274.0           235.0       616.4       219.0     37.85    -122.25
4       1627.0           280.0       565.0       259.0     37.85    -122.25
Unnamed: 0            0
median_house_value    0
median_income         0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
latitude              0
longitude             0
dtype: int64
```

Conclusion:

The KNN Imputation method has been successfully applied to the Houses_0.5_MAR dataset. This procedure has prepared the dataset for further analysis, such as building regression models to predict housing prices. The imputation ensures that the subsequent models are trained on a complete dataset, potentially leading to more accurate predictions.

Regression Model (M1) Using Imputed Dataset:

Objective:

The primary objective of this analysis is to employ KNN imputation for handling missing data in the houses_0.5_MAR dataset and to construct a regression model, herein referred to as M1, to predict the median house value. This exercise aims to evaluate the efficacy of the KNN imputation method on the dataset's predictive performance.

Methodology:

In this study, **Linear Regression** was chosen as the regression technique due to its simplicity, interpretability, and efficiency in handling continuous target variables. It serves as a benchmark for evaluating the impact of imputation methods on the prediction of house median prices. The approach ensures a consistent basis for comparison with other imputation techniques that may be applied subsequently.

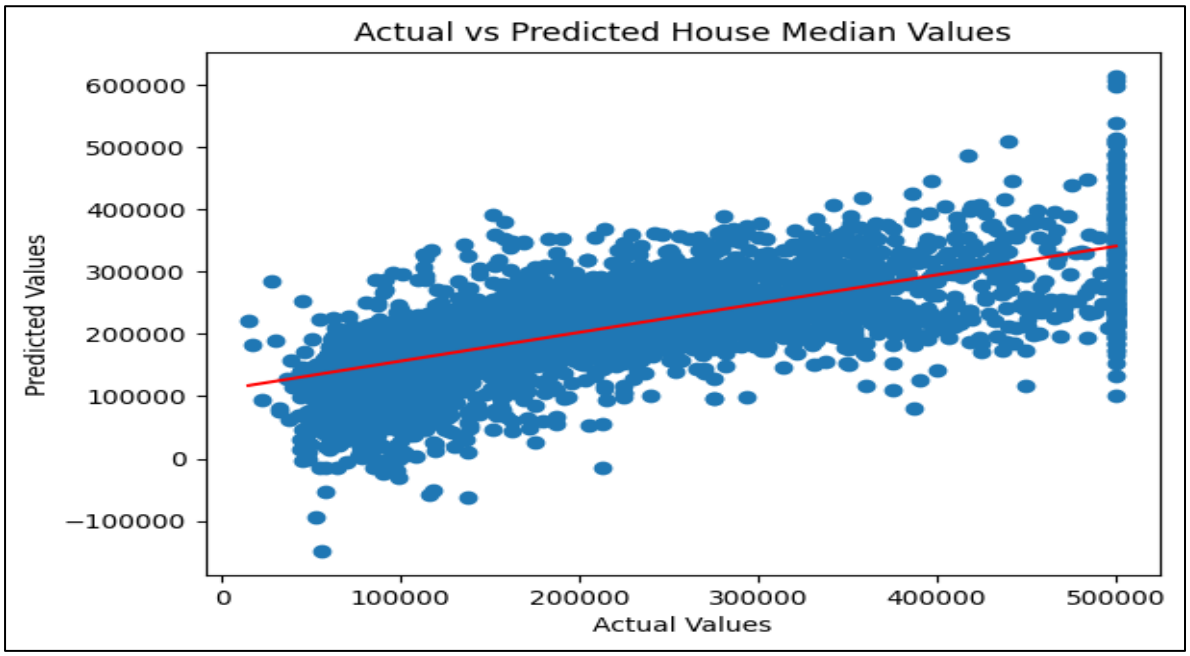
Implementation:

The implementation involved a systematic process starting with the KNN imputation to address missing values in the dataset, followed by the application of a Linear Regression model for prediction purposes. The dataset was first pre-processed to ensure compatibility with the imputation technique. Subsequently, the KNN Imputer was initialized and applied to the dataset, with the exclusion of the target variable to maintain its integrity. The imputed dataset was then used to train the Linear Regression model, with the code detailing each step of the process. The code snippet that uses a regression model named M1 to predict the house median price using the imputed dataset:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np

# Define X (features) and y (target)
X = df_imputed.drop('median_house_value', axis=1)
y = df_imputed['median_house_value']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the linear regression model (M1)
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the testing set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Plotting actual vs predicted values with a regression line
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted House Median Values')

# Adding the regression line
plt.plot(np.unique(y_test), np.poly1d(np.polyfit(y_test, y_pred, 1))(np.unique(y_test)), color='red')  # Regression line

plt.show()

# Display the evaluation metrics
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

Observation and Results:

The M1 regression model, employing Linear Regression, was trained on KNN-imputed data to predict median house values. The model's evaluation, utilizing a split between training and test data, aimed to assess its performance in predicting unseen data. Post-training, the model's predictions on the test data were contrasted with the actual values, revealing insights into its predictive accuracy. The scatter plot with the regression line further illustrates the fit of the model to the data, indicating areas where the model performs well alongside those where improvement is needed.



Model Performance Evaluation:

Overview

The M1 model, employing Linear Regression, was developed to forecast the median house price following the KNN imputation of sparse features in the dataset. This model is expected to decipher the underlying relationship between house characteristics and their median value, and its performance is crucial for reliable predictions.

Statistical Analysis

The performance of M1 was evaluated using two primary statistical measures:

- Mean Squared Error (MSE): This metric provided an average of the squares of the differences between the actual and predicted median house values. The computed MSE for M1 was **6984343314.801312**, indicating the average squared deviation of the predicted values from the actual values.
- R-squared: The R-squared value of **0.4670104195884607** reflected the proportion of variance in the median house value that was predictable from the features. An R-squared value closer to 1 would indicate a model that can better account for the variance in the target variable.

Upon evaluation, the M1 model yielded a Mean Squared Error (MSE) of approximately 6.98 billion. This figure signifies the average squared difference between the estimated values and the actual value, reflecting the model's accuracy level. The substantial MSE suggests that the model's predictions deviate considerably from the true house prices.

The R-squared statistic for the model stands at 0.467, which indicates that approximately 46.7% of the variance in the median house value is explained by the model. An R-squared value below 0.5 implies that the model has moderate predictive power, with more than half of the variability in the dependent variable remaining unaccounted for by the independent variables in the model.

Graphical Analysis

The scatter plot visualization presents a direct comparison between the actual and predicted median house values. Points scattered across the plot reveal the discrepancies between predicted and true values. The fitted regression line within the plot attempts to capture the central tendency of these predictions. Observations suggest that the model maintains a moderate fit, with a tendency to underpredict as the house values increase. Notably, for high-value houses, the predictions are not as tightly clustered around the regression line, implying larger prediction errors in that range.

This graphical representation aligns with the statistical findings, both indicating that while M1 can capture a general trend in the housing prices, its predictions are imprecise, particularly at the higher end of the house value spectrum. The model's limitations could stem from various factors, including the simplicity of the linear approach, potential non-linearity in the data, or the need for more sophisticated feature engineering.

Performance Assessment:

The performance of the M1 model, as indicated by both the statistical and graphical analyses, suggests a moderate level of prediction accuracy. The mean squared error is relatively high, which points to significant discrepancies between the predicted and actual values, especially for houses at the upper end of the value spectrum. This could be due to various factors that influence house prices, such as location, local economic conditions, and property-specific characteristics, which may not be fully captured by the model.

The R-squared value of 0.467 indicates that less than half of the variance in the housing prices is explained by the model. This further underscores the model's limited capacity to account for all the variability in the data, suggesting that there are other factors at play that the model does not incorporate.

One possible reason for the model's moderate performance could be the linear nature of the regression technique. Real-world data, particularly in housing markets, often exhibit non-linear relationships that linear models cannot adequately capture. Features such as proximity to city centres, neighbourhood quality, and property age often have complex, non-linear effects on house prices.

Another factor could be the imputation method used. While KNN imputation is effective for handling missing data, it assumes that houses with similar characteristics in the smaller subset of features have similar values in the sparse features. If this assumption does not hold, the imputed values may not accurately represent the true values, leading to less accurate predictions.

Moreover, the model might be oversimplified. Housing price prediction is a complex problem that may require a nuanced approach, including interaction terms, polynomial features, or even more advanced machine learning methods that can model complex non-linear relationships.

In conclusion, while the M1 model offers some insights into the factors affecting house prices, its predictive power is limited. Enhancements such as incorporating more features, using more complex models, or exploring different imputation methods may be necessary to improve performance and achieve more accurate predictions.

MICE Imputation

Objective

The objective of the MICE (Multiple Imputation by Chained Equations) imputation process is to address the missing data in the Houses_0.5_MAR dataset. This method is particularly useful for datasets with multiple missing values, as it provides a robust way of handling such gaps by modelling each feature with missing values as a function of other features in a round-robin fashion.

Methodology

The methodology adopted for the MICE imputation involved the use of the **IterativeImputer** from Scikit-learn, a Python library that offers various tools for data mining and analysis. This imputer method models each feature with missing values as a function of other features in a round-robin fashion.

Implementation

The implementation of MICE imputation was carried out using Python and the Scikit-learn library. The dataset was loaded using the pandas library, and the IterativeImputer was utilized to impute the missing values for all features except the target variable **'median_house_value'**. The following code snippet was executed to apply the MICE Imputer to our dataset:

```python
import pandas as pd
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

# Load the dataset
df = pd.read_csv('C:/Users/athar/Downloads/houses_0.5_MAR.csv')  # Update this to your file path

# Initialize the MICE Imputer
mice_imputer = IterativeImputer()

# Perform imputation on the dataframe except for the target variable 'median_house_value'
features_to_impute = df.columns.drop('median_house_value')  # Assuming 'median_house_value' has no missing values
df_imputed = df.copy()
df_imputed[features_to_impute] = mice_imputer.fit_transform(df[features_to_impute])
df_imputed
# The df_imputed now contains the original data with missing values imputed by MICE
# To display the first few rows of the imputed dataframe
print(df_imputed.head())

# To check for missing values after imputation
print(df_imputed.isnull().sum())
```

Observations

MICE (Multiple Imputation by Chained Equations) was employed to address missing values in the dataset. This iterative approach creates several imputations for each missing value, considering other available features. It offers a more comprehensive data reconstruction compared to single imputation methods like KNN.

Results

Upon completion of the MICE imputation, the dataset was checked for missing values. The imputation was verified, ensuring that no missing values remained and the 'median_house_value' was not altered.

```
   Unnamed: 0  median_house_value  median_income  housing_median_age  \
0         0.0            452600.0         8.3252                41.0
1         1.0            358500.0         8.3014                21.0
2         2.0            352100.0         7.2574                52.0
3         3.0            341300.0         5.6431                52.0
4         4.0            342200.0         3.8462                52.0

   total_rooms  total_bedrooms   population  households  latitude  longitude
0        880.0           129.0  -356.739639       126.0     37.88    -122.23
1       7099.0          1106.0  3004.892722      1138.0     37.86    -122.22
2       1467.0           190.0   -75.821465       177.0     37.85    -122.24
3       1274.0           235.0   129.981205       219.0     37.85    -122.25
4       1627.0           280.0   565.000000       259.0     37.85    -122.25
Unnamed: 0            0
median_house_value    0
median_income         0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
latitude              0
longitude             0
dtype: int64
```

Conclusion

MICE imputation has successfully prepared the Houses_0.5_MAR dataset for further predictive modelling. This method is particularly beneficial for datasets with complex relationships and patterns, as it can provide a more nuanced and informed approach to handling missing data. The dataset is now ready for the construction of the regression model M2, aiming to predict housing prices accurately.

Regression Model (M2) Using Imputed Dataset:

Objective

The aim is to utilize a regression model, designated as M2, to predict the median house value using the dataset that has undergone MICE imputation. This step assesses the effectiveness of MICE in preparing data for accurate predictive modelling.

Methodology

Linear Regression was chosen for both M1 and M2 due to its straightforwardness and ease of interpretation. Employing the same regression method ensures that any differences in performance can be attributed to the imputation technique rather than the predictive model itself.

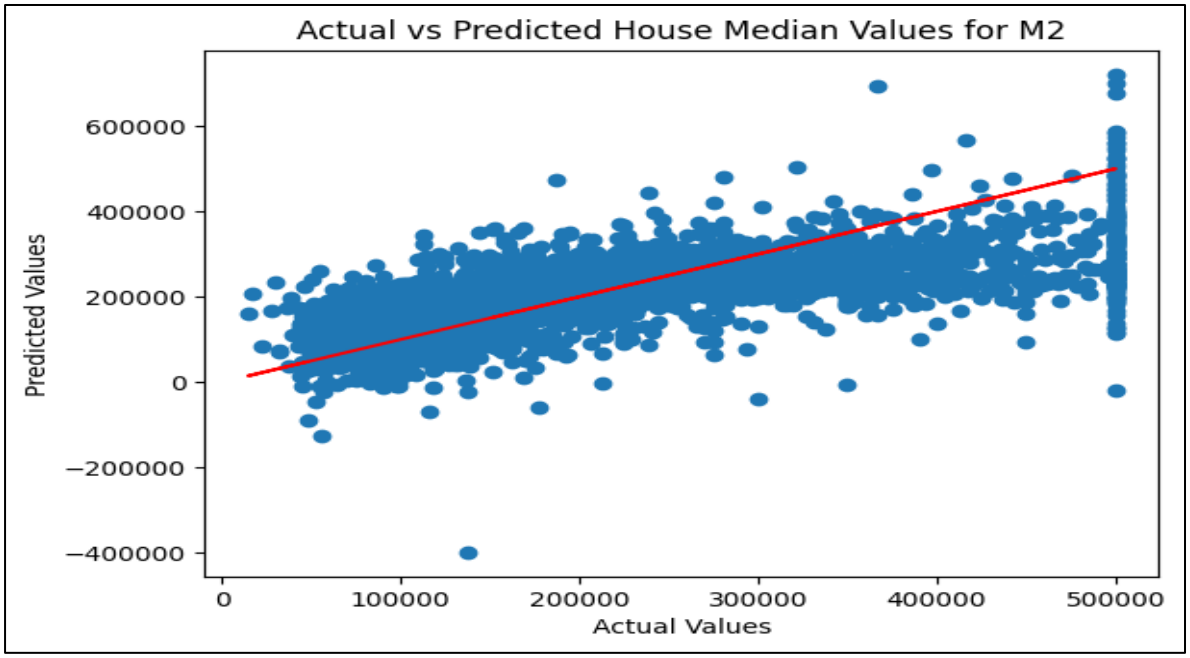
Implementation

- The MICE imputation method was applied to the dataset, and the resulting imputed dataset was used as the input for M2.
- The model aimed to predict **'median_house_value'** utilizing features enriched by MICE imputation.
- A train-test split approach was adopted to ensure the model's generalizability.
- M2 was trained on this MICE-imputed data, learning to infer the median house price from the available features.

```python
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Define X (features) and y (target)
X = df_mice_imputed.drop('median_house_value', axis=1)
y = df_mice_imputed['median_house_value']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the linear regression model (M2)
model_m2 = LinearRegression()
model_m2.fit(X_train, y_train)

# Predict on the testing set
y_pred_m2 = model_m2.predict(X_test)

# Evaluate the model
mse_m2 = mean_squared_error(y_test, y_pred_m2)
r2_m2 = r2_score(y_test, y_pred_m2)

# Plotting actual vs predicted values
plt.scatter(y_test, y_pred_m2)
plt.plot(y_test, y_test, color='red')  # Line for perfect predictions
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted House Median Values for M2')
plt.show()

# Display the evaluation metrics for M2
print(f"Mean Squared Error (M2): {mse_m2}")
print(f"R-squared (M2): {r2_m2}")
```

Observation and Results

The regression model M2's performance, as determined by the scatter plot, shows a clustering of predicted values around the regression line, although several points lie far from it, indicating potential overestimation or underestimation in certain price ranges. The Mean Squared Error (MSE) for M2 stands at approximately 7.09 billion, with an R-squared value of 0.459, slightly below that of M1. These metrics provide a quantitative assessment of M2's predictive accuracy and the variance explained by the model.



Conclusion

M2, constructed upon the dataset imputed with MICE, offers a nuanced perspective on the efficacy of this imputation technique within the realm of linear regression. While M2 demonstrates a level of predictive power, its performance metrics, such as the R-squared value, are marginally lower compared to M1, which utilized KNN imputation. This slight difference prompts further investigation into the dataset's characteristics and the appropriateness of the imputation method in relation to the model's predictive capability.

Model Performance Evaluation for M2:

Overview

The M2 model, also using Linear Regression, was designed to predict the median house price following MICE imputation of sparse features. It aims to discern the intricate patterns between housing characteristics and their median value. Assessing M2's performance is pivotal for determining the effectiveness of MICE imputation in predictive modelling.

Statistical Analysis

M2's performance was appraised using the same statistical metrics as M1:

- Mean Squared Error (MSE): The MSE for M2 is **7088726842.969596**, which represents the mean of the squares of the errors between actual and forecasted values. A high MSE suggests that the model may have inaccuracies in prediction.
- R-squared: The R-squared value for M2 is around **0.459**, signifying that about **45.9%** of the variability in the median house value is explained by the independent variables in the model. This indicates a moderate level of predictive ability, similar to M1, but slightly lower in explanatory power.

Graphical Analysis

The **scatter plot** for M2 shows the distribution of predicted versus actual median house values. Similar to M1, the plot illustrates some deviation of predictions from the actual values, particularly for houses at the higher price range, which can be inferred from the spread of points around the regression line. The regression line in the plot illustrates the average prediction across the value spectrum but may not capture the nuances in the higher-end market segment.

By juxtaposing the actual and predicted values, the scatter plot provides visual evidence of the model's performance. Although the regression line suggests a positive correlation between the features and the median house value, the spread of the points indicates variability in the model's accuracy, especially for properties with higher median values.

Performance Assessment for M2

The M2 model's performance reflects a similar trend to the M1 model, with both statistical metrics and graphical analysis pointing to a moderate predictive capability. The Mean Squared Error for M2 is slightly higher than that of M1, implying that the predictions may have marginally increased variance from the actual values. This could suggest that while MICE imputation provides a robust method for dealing with missing data, it does not significantly improve the predictive accuracy of the model over KNN imputation in this context.

The R-squared value for M2, marginally lower than M1's, reiterates the model's moderate explanatory power, with a large proportion of variance in housing prices unexplained by the predictor variables. This underscores the inherent complexity of housing market data and indicates that the factors influencing house prices may extend beyond the scope of the variables considered in the model.

The slightly diminished performance of M2 compared to M1 could stem from the intricacies of the MICE imputation process, which involves multiple imputations and may introduce variability that does not necessarily align with the underlying data distribution. Unlike KNN, which relies on the nearest neighbours and may capture local similarities more directly, MICE generate multiple complete datasets based on the distribution of the data, which are then averaged out. This approach, while theoretically sound, may not always translate into improved predictive performance, particularly when the underlying patterns in data are complex and non-linear.

Additionally, the linear regression model may not fully capture the nuances of the housing market, where price determinants can interact in non-linear and intricate ways. The presence of high-value outliers and the spread of points around the regression line in the scatter plot for M2 suggest that alternative modelling techniques that can account for non-linearities and complex interactions may yield better results.

In essence, the M2 model, although utilizing a sophisticated imputation technique, still faces challenges in accurately predicting house prices, which could be attributed to the limitations of the linear regression approach and the potential disconnect between imputed values and real-world data complexities. To enhance predictive accuracy, a more detailed analysis involving feature engineering, the inclusion of additional relevant data, or the adoption of more advanced modelling techniques might be necessary.

Comparison of M1 and M2

Using Graphical and Statistical Methods:

When comparing the performance of M1 and M2 models, it's important to note that both used the same regression method but differed in their imputation techniques—KNN for M1 and MICE for M2. Here's a comparative analysis based on the statistical and graphical information provided:

Statistical Analysis

Mean Squared Error (MSE):

- M1: MSE = 6,984,343,314.80
- M2: MSE = 7,088,726,842.97

The MSE for M2 is slightly higher than for M1, which indicates that, on average, the predictions from M2 are further from the actual values than those from M1.

R-squared (R^2):

- M1: $R^2 = 0.467$
- M2: $R^2 = 0.459$

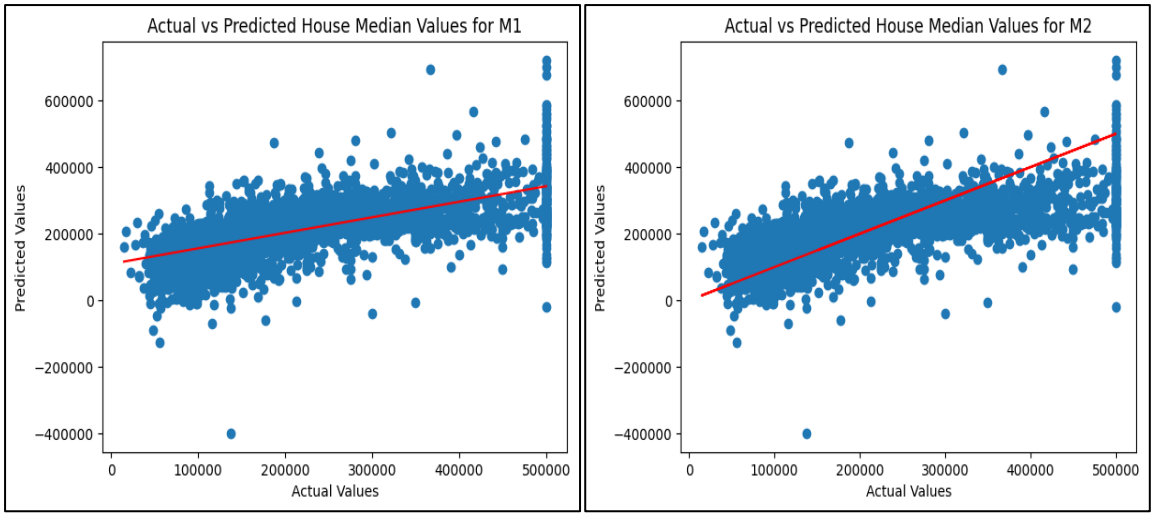
The R-squared value for M1 is marginally higher than for M2, suggesting that M1's model can explain a slightly larger proportion of the variance in the median house value.

When we compare the statistical metrics between M1 and M2, we observe that both models have a similar level of predictive power, as evidenced by their R-squared values: M1 at approximately 0.467 and M2 at around 0.459. This indicates that both models explain less than half of the variance in the median house values with their predictions. The Mean Squared Error (MSE) for M1 is around 6.98 billion, while for M2, it is slightly higher at approximately 7.09 billion. The higher MSE in M2 suggests that, on average, its predictions are slightly more dispersed from the actual values compared to M1.

Graphical Analysis:

Scatter Plots:

- Both models show a moderate fit with a tendency to underpredict higher values.
- The scatter plot for M1 shows a spread of predictions that indicates moderate predictive power, with a notable spread for high-value houses.
- M2's scatter plot also displays a spread of predictions with a slightly more pronounced deviation for higher-priced houses, implying larger errors for these values.



The scatter plots for both M1 and M2 show the actual versus predicted median house values with a fitted regression line. In both graphs, we see that as the house values increase, the models tend to underpredict, evident from the spread of points below the regression line. This underprediction is more pronounced for higher-value houses, where the points are more scattered, suggesting larger prediction errors for both models.

Comparison on basis of KNN and MICE Imputation:

The comparison between the M1 and M2 models, following KNN and MICE imputation respectively, reveals minor differences in performance. Statistically, both models show a moderate capability to predict median house values, with M1 slightly outperforming M2 in terms of R-squared, although M2 has a marginally higher MSE. These results suggest that neither imputation technique offered a significant advantage over the other for this particular dataset when used as a precursor to linear regression modelling.

The rationale behind the performance differences can be understood in the context of how each imputation method handles the missing data. KNN imputation works on the premise that similar data points can be found within the closer proximity of the feature space. It then averages the values of the nearest neighbours to fill in the missing data. This approach can be highly effective if the dataset has a strong local structure and if the missing values are not systematically biased.

MICE, on the other hand, uses a more sophisticated approach that iteratively models each feature with missing values as a function of other features, in a chained equation system. It is particularly useful when the missingness pattern is complex and potentially related to other variables in a non-random way.

The slight performance edge of M1 could suggest that the local averaging inherent in KNN was slightly more aligned with the underlying patterns in this dataset, or it could simply be a result of random variation in the performance measures. On the other hand, the complexity

of the MICE approach did not seem to capture additional patterns that would significantly improve the prediction accuracy in this instance.

In choosing between the two for future tasks, the decision would likely depend on the nature of the missing data and the computational resources available. If the missing data are believed to be MAR (Missing At Random) and the dataset is large, KNN could be a good choice due to its simplicity and lower computational cost. If the missing data are MNAR (Missing Not At Random) or if there are complex relationships between variables, MICE might be more appropriate despite its higher computational demands. The performance difference in this context may not be substantial enough to favour one method conclusively over the other without further analysis and validation.

Method to choose:

In determining the most suitable imputation method for this dataset, one would need to consider the characteristics of the missing data, the computational resources, and the ultimate goal of the analysis.

Given the observed performances, KNN imputation (M1) provided a slightly higher R-squared value, which means it was marginally better at capturing the variability of the median house values with the given features. This could indicate that the patterns within the data are more local, and the similarities between the nearest neighbours provide a good proxy for the missing values.

On the other hand, MICE imputation (M2) showed a slightly higher Mean Squared Error, which could suggest that this iterative approach, despite being more complex and potentially more accurate for datasets with complex missingness patterns, did not significantly enhance the prediction for this particular dataset.

For this reason, if the missing data is assumed to be Missing At Random (MAR) and the dataset not exceedingly large, KNN imputation could be the preferred method. It is relatively straightforward, less computationally intensive, and in this case, offered a marginally better fit. Moreover, KNN is a non-parametric method, meaning it does not assume an underlying distribution for the input variables, which can be advantageous when such assumptions are hard to justify.

However, if there is a suspicion that the data might be Missing Not At Random (MNAR), or if there are complex inter-feature relationships that the model needs to capture, then despite the slightly lower performance in this instance, MICE would be the method of choice. It is more robust to different missing data mechanisms and can handle complexities in the data that simpler methods like KNN might miss.

In conclusion, given the current dataset and the performance metrics obtained, **KNN would be chosen for its simplicity and marginally better performance in capturing the variability of house prices**. However, this choice might be revisited if further analysis suggests complex missingness patterns or if there is a significant change in the dataset size or feature relationships.