Lab 2

Data processing

In [32]:

```
import pandas as pd

# Load dataset
url = "https://media.geeksforgeeks.org/wp-content/uploads/20240319120216/housing.csv"
housing = pd.read_csv(url)

# Overview of dataset
print(housing.info())  # Check data types and missing values
print(housing.describe())  # Summary statistics
print(housing.head())
```

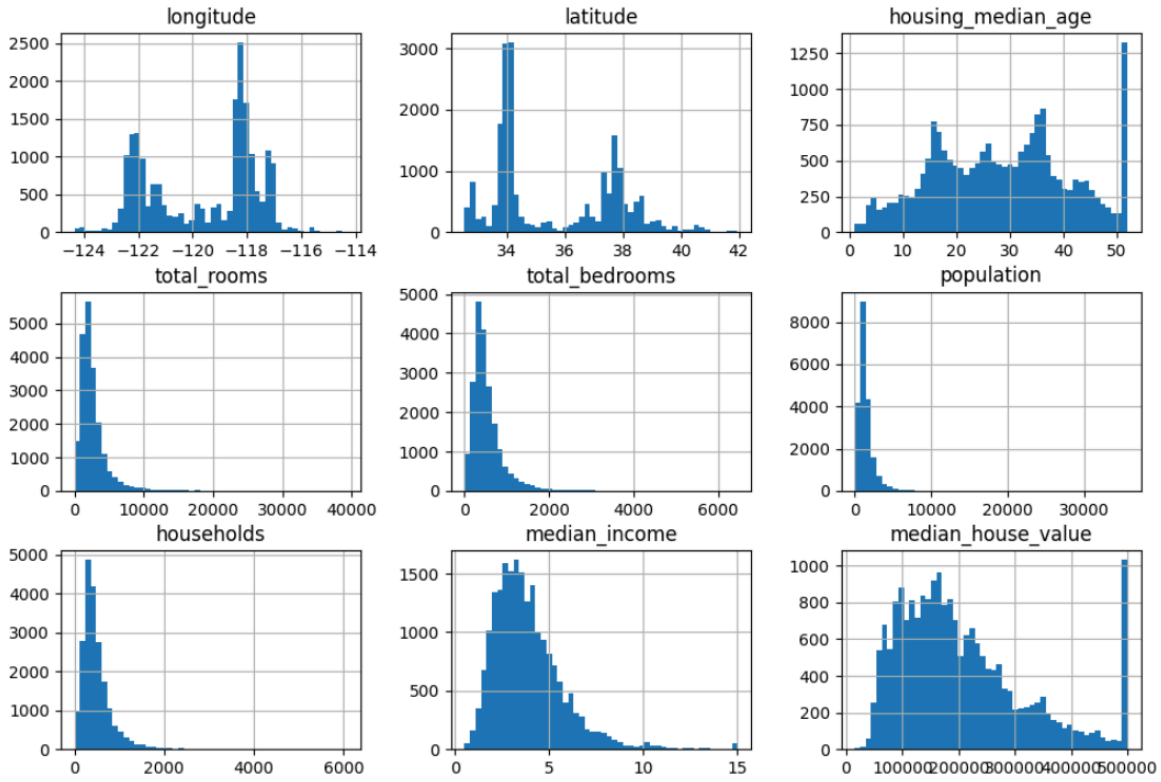
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000   
mean    -119.569704     35.631861           28.639486   2635.763081   
std        2.003532      2.135952           12.585558   2181.615252   
min     -124.350000     32.540000            1.000000      2.000000   
25%     -121.800000     33.930000           18.000000   1447.750000   
50%     -118.490000     34.260000           29.000000   2127.000000   
75%     -118.010000     37.710000           37.000000   3148.000000   
max     -114.310000     41.950000           52.000000  39320.000000   
```

```
       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100
```

```
       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

In [33]:

```
import matplotlib.pyplot as plt

housing.hist(bins=50, figsize=(12, 8))
plt.show()
```

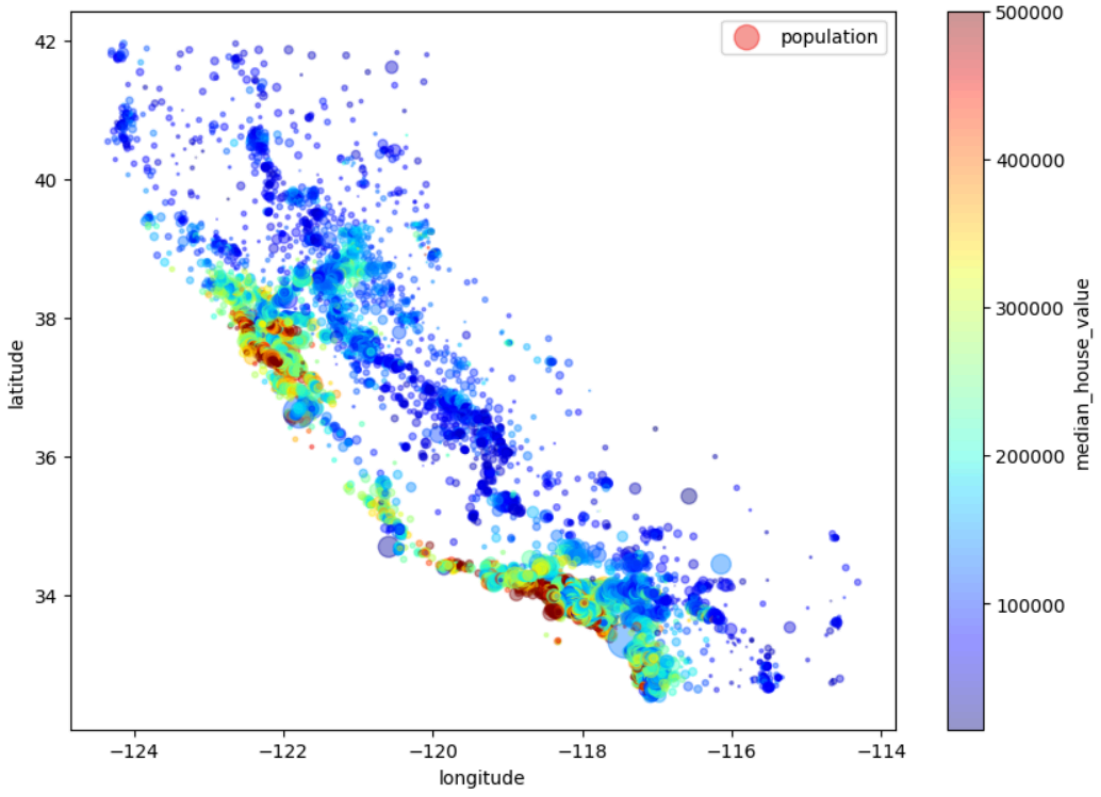


In [34]:

```
#creating test set
from sklearn.model_selection import train_test_split
import numpy as np

# Random Sampling
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)

# Stratified Sampling by median_income
housing["income_cat"] = pd.cut(housing["median_income"],
                               bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                               labels=[1, 2, 3, 4, 5])

from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)

for train_index, test_index in split.split(housing, housing["income_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

In [36]:

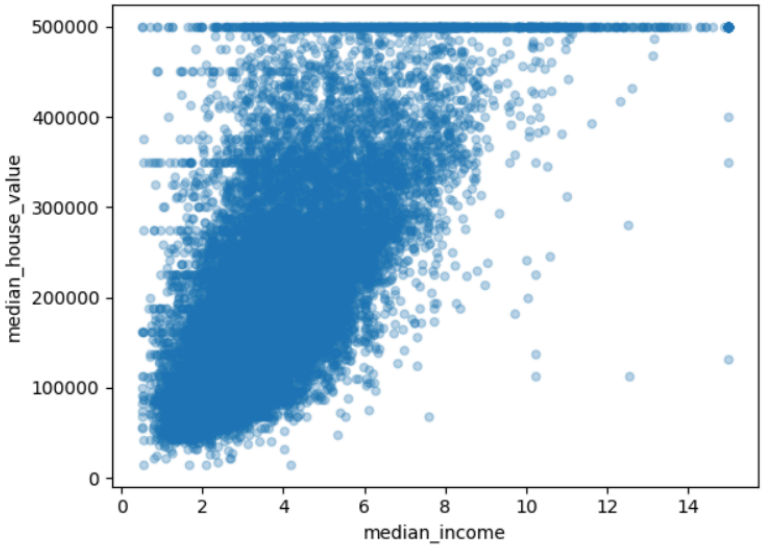
```
#geographical features visualization
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
             s=housing["population"]/100, label="population",
             c="median_house_value", cmap=plt.get_cmap("jet"),
             colorbar=True, figsize=(10,7))
plt.legend()
plt.show()
```



In [37]:
```
#feature corelation with housing prices
corr_matrix = housing.select_dtypes(include=[np.number]).corr()
print(corr_matrix["median_house_value"].sort_values(ascending=False))
housing_encoded = pd.get_dummies(housing, drop_first=True)  # Convert categorical to numerical
corr_matrix = housing_encoded.corr()
print(corr_matrix["median_house_value"].sort_values(ascending=False))
```

```
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
median_house_value            1.000000
median_income                 0.688075
income_cat_5                  0.535253
income_cat_4                  0.206218
ocean_proximity_NEAR BAY      0.160284
ocean_proximity_NEAR OCEAN    0.141862
total_rooms                   0.134153
housing_median_age            0.105623
households                    0.065843
total_bedrooms                0.049686
ocean_proximity_ISLAND        0.023416
population                   -0.024650
longitude                    -0.045967
income_cat_3                 -0.046115
latitude                     -0.144160
income_cat_2                 -0.416886
ocean_proximity_INLAND       -0.484859
Name: median_house_value, dtype: float64
```

In [38]:
```
#visualizing median_income vs median_house_value
housing.plot(kind="scatter", x="median_income", y="median_house_value", alpha=0.3)
plt.show()
```



In [39]:
```
#feature engineering-combining features
housing["rooms_per_household"] = housing["total_rooms"] / housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"] / housing["total_rooms"]

corr_matrix = housing.select_dtypes(include=[np.number]).corr()
print(corr_matrix["median_house_value"].sort_values(ascending=False))
housing_encoded = pd.get_dummies(housing, drop_first=True)  # Convert categorical to numerical
corr_matrix = housing_encoded.corr()
print(corr_matrix["median_house_value"].sort_values(ascending=False))
```

```
median_house_value     1.000000
median_income          0.688075
rooms_per_household    0.151948
total_rooms            0.134153
housing_median_age     0.105623
households             0.065843
total_bedrooms         0.049686
population            -0.024650
longitude             -0.045967
latitude              -0.144160
bedrooms_per_room     -0.255880
Name: median_house_value, dtype: float64
median_house_value            1.000000
median_income                 0.688075
income_cat_5                  0.535253
income_cat_4                  0.206218
ocean_proximity_NEAR BAY      0.160284
rooms_per_household           0.151948
ocean_proximity_NEAR OCEAN    0.141862
total_rooms                   0.134153
housing_median_age            0.105623
households                    0.065843
total_bedrooms                0.049686
ocean_proximity_ISLAND        0.023416
population                   -0.024650
longitude                    -0.045967
```

In [29]:
```
#handling categorical data
housing = pd.get_dummies(housing, columns=["ocean_proximity"], drop_first=True)
```

In [41]:
```
#creating machine learning pipeline
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer

# Separating numerical and categorical attributes
num_attribs = housing.drop("median_house_value", axis=1).select_dtypes(include=[np.number]).columns
cat_attribs = ["ocean_proximity"]

# Pipeline for numerical attributes
num_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

# Full pipeline
full_pipeline = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_attribs)
])

# Applying the pipeline
housing_prepared = full_pipeline.fit_transform(housing)
```

In [42]:

```
#mode selection and training
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Train Model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(housing_prepared, housing["median_house_value"])

# Predict
predictions = model.predict(housing_prepared)

# Evaluate
mae = mean_absolute_error(housing["median_house_value"], predictions)
print("Mean Absolute Error:", mae)
```

```
Mean Absolute Error: 11608.788181862672
```

10/3/25
Monday

Date
Page

Lab - 2

Demonstrate the steps to build a machine-learning model that predicts the median housing price using the california housing price dataset.

1. Download the dataset :

→
```
import pandas as pd
url = "https://media.geeksforgeeks.org/wp-content/uploads/20240319120216/housing.csv"
housing = pd.read_csv(url)
```

2. Perform the describe and info steps

→
```
print(housing.info())
print(housing.describe())
print(housing.head())
```

2. Plot the histogram of each feature (indicate what does histogram indicate on median_income and house_median_age).

→
```
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(12,8))
plt.show()
```

~~3. Test the progs what does the the graphs~~

3. Demonstrate the process of creating a test set (write diff b/w random and stratified test set)

→ Random test set :

- Samples are randomly selected for the test set
- Suitable for balanced datasets
- can lead to an unrepresentative test set, Especially with imbalanced data.

Stratified Test Set:
- Ensures the test set reflects the same distribution of target classes as the full dataset
- Ideal for imbalanced datasets
- Preserves class proportions in both training and test sets.

4) what does the graph indicates w.r.t housing price and location.

-> - Shows how the housing prices are distributed across different geographical location (latitude and longitude)
- locations with color and size of the points represents the housing price
- If the points cluster in specific areas, it may indicate that housing prices tend to be higher in central or popular areas.

5) Plot a graph to show feature correlation with housing price. which feature corelate to the maximum. plot the graph for that with housing price and analyze what the graph indicate.

-> - If the plot shows a strong upward trend (positive correlation), it means that larger homes (in terms of square footage) tend to have higher prices.
- A weaker or no trend indicates less of a relationship b/w the two variable

6) List the feature that could be combined to improve correlation and plot again to see if correlation had improved.

→ • total_living_space = sqft_living * num_bedrooms.
• total_area = Sqft_living + Sqft_lot
• bathroom_to_bedroom_ratio = num_bathrooms / num_bedrooms
• city_distance_factor = distance_to_city_center * elevation

7) List the feature that needs to be cleaned and demonstrate the process of cleaning

→ 1. Missing Values.
→ num_bedrooms or price might have missing values
2. outliers.
Sqft_living, num_bedrooms may have Extreme values that are far outside the Expected range
3. Duplicates:
→ Duplicates means in that dataset can artificially inflate the data, leading to biased results.

8) Is there any categorical data that needs to be converted to numerical? if so Explain the method used to convert and code the same and show the output.

→ Ocean_Proximity needs to be converted to numerical, It is done by.
housing = pd.get_dummies (housing, columns = ["ocean_proximity"], drop_first = True)
One-Hot Encoding is used to convert Column into multiple new binary

Column, Each representing one category of ocean proximity column and then one category is dropped.

Q8. Importance of feature Scaling:
→ Some ML models like linear regression, neural networks require feature Scaling
→ Standardization
→ min-max Scaling

Q9 For pipeline encoding
→ we first separate numerical and categorical data
- pipeline for numerical attributes fills missing values with median and use feature Scaling.
- Full pipeline applies num_pipeline to numerical attributes and uses one-Hot Encoding for categorical attributes to convert it to numerical format. It ignores unknown categories.

10/3/25