1. Installing Kaggle:

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.8.30)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.6)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.2.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.10)
```

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d camnugent/california-housing-prices
```

```
Dataset URL: https://www.kaggle.com/datasets/camnugent/california-housing-prices
License(s): CC0-1.0
Downloading california-housing-prices.zip to /content
  0% 0.00/400k [00:00<?, ?B/s]
100% 400k/400k [00:00<00:00, 86.7MB/s]
```

```
!unzip california-housing-prices.zip
```

```
Archive:  california-housing-prices.zip
  inflating: housing.csv
```

2. Importing relevant Python packages:

```
# for data manipulation
import pandas as pd
import numpy as np

# for data visualization
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

# for data modelling
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

# for metrics and helpful functions
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

3. Loading the dataset:

```
# loading the dataset into a dataframe
df = pd.read_csv('housing.csv')

# displaying the first few rows of the dataframe
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0

4. Initial data exploration and data cleaning:

```
# gathering basic information about the data
df.info()
```

```
<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
```

```
# gathering descriptive statistics about the data
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	2
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	1
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	1
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	1
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	2
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	5

```
# checking for missing values
df.isna().sum()
```

	0
longitude	0
latitude	0
housing_median_age	0
total_rooms	0
total_bedrooms	207
population	0
households	0
median_income	0
median_house_value	0
ocean_proximity	0

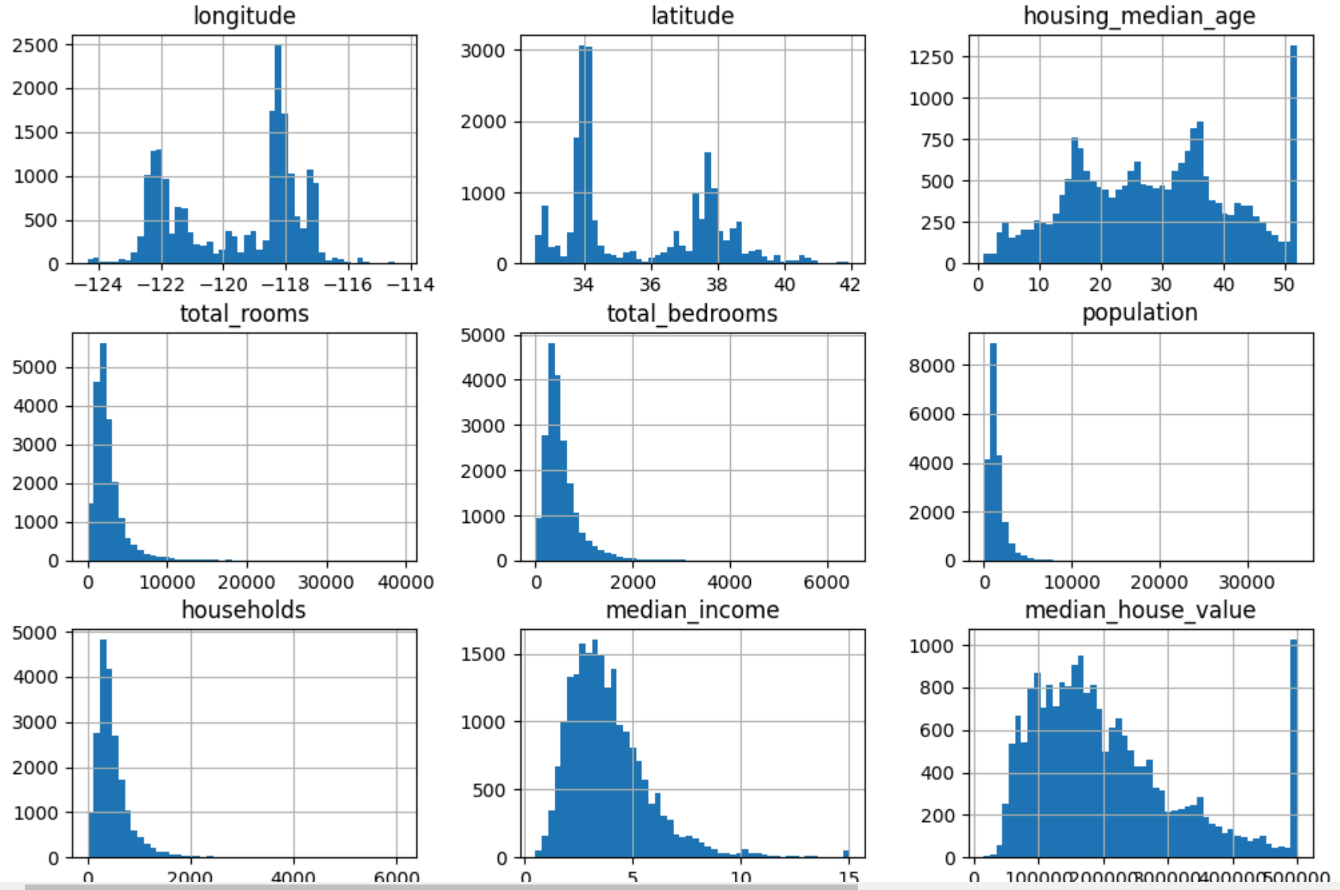
```
# dropping the missing values from the dataset
df = df.dropna()
```

```
# rechecking for missing values
df.isna().sum()
```

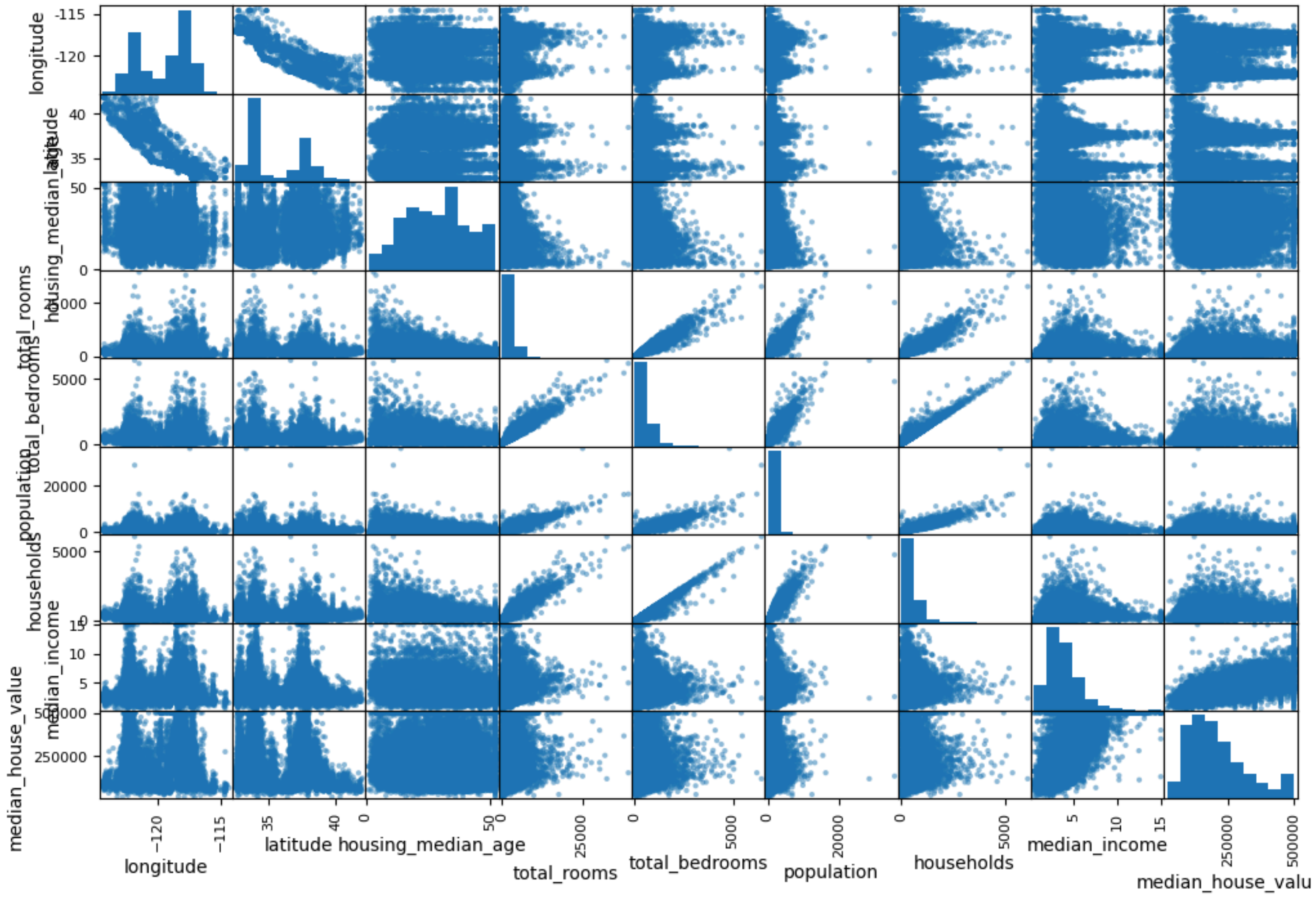
```
                     0
         longitude   0
          latitude   0
housing_median_age   0
       total_rooms   0
    total_bedrooms   0
        population   0
        households   0
     median_income   0
median_house_value   0
   ocean_proximity   0
```

5. Feature exploration through visualization:

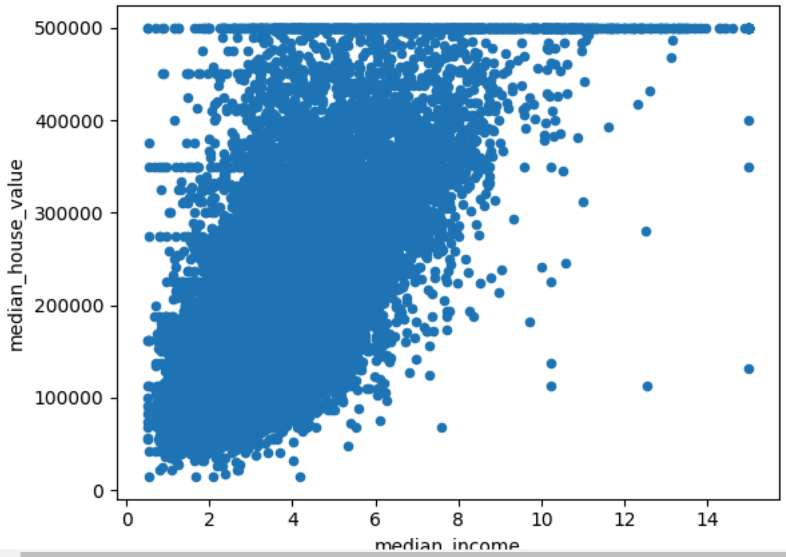
```
# visualizing the data through histograms
df.hist(bins=50, figsize=(12,8))
plt.show()
```



```
# plotting the correlations between the features against each other
attributes = ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income
scatter_matrix(df[attributes], figsize=(12,8))
plt.show()
```



```
# diving deeper into the correlation between median_income and median_house_value
df.plot(kind="scatter", x="median_income", y="median_house_value")
plt.show()
```



```
# list of all the relevant correlation values
df1 = df[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income',
corr = df1.corr()
corr['median_house_value'].sort_values(ascending=True)
```

	median_house_value
latitude	-0.144638
longitude	-0.045398
population	-0.025300
total_bedrooms	0.049686
households	0.064894
total_rooms	0.133294

6. Feature selection:

```
# features for linear regression
X = df[['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'oc

# target variable
y = df['median_house_value']
```

7. Converting the categorical variable

```
# using one-hot encoding
X = pd.get_dummies(X, columns=['ocean_proximity'], drop_first=True)
```

8. Building and testing the linear regression model:

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


# fitting the model
regression_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', LinearRegression())
])
regression_pipeline.fit(X_train, y_train)
```

```
▸ Pipeline
  ▸ StandardScaler
  ▸ LinearRegression
```

```
# prediction and evaluation
y_pred = regression_pipeline.predict(X_test)
r2_score(y_test, y_pred)
```

```
0.6400865688993737
```