Objective:

To apply Multiple Linear Regression and Polynomial Regression on the California Housing dataset, handle missing values and categorical variables, and compare the performance of both models using appropriate evaluation metrics.

Tasks:

Task 1: Dataset Loading and Preprocessing

- Load the California Housing dataset using Pandas.
- Inspect the dataset to understand feature types and structure.
- Handle missing (NaN) values by replacing numerical missing values with column means.
- Convert categorical features into numerical form using One-Hot Encoding.

Task 2: Feature and Target Preparation

- Identify the target variable (`median_house_value`).
- Separate input features and the target variable.
- Split the dataset into training and testing sets (80% training, 20% testing).

Task 3: Multiple Linear Regression

- Train a Multiple Linear Regression model using the training data.
- Predict housing prices on the test dataset.
- Evaluate the model using the following metrics:
  - Mean Squared Error (MSE)
  - R^2 Score

Task 4: Polynomial Regression

- Transform input features using Polynomial Features of degree 2.
- Train a Polynomial Regression model using the transformed features.
- Predict housing prices on the test dataset.
- Evaluate the model using the following metrics:
  - Mean Squared Error (MSE)
  - R^2 Score

Task 5: Model Comparison

- Compare the performance of Multiple Linear Regression and Polynomial Regression models.
- Analyze the impact of polynomial feature expansion on model accuracy.

In [3]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
import kagglehub
import os
```

```
c:\Users\adity\OneDrive\Documents\GitHub\MLLab\.venv\Lib\site-packages\tqdm\auto.
py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. Se
e https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [4]:
```
# Download dataset from Kaggle
dataset_path = kagglehub.dataset_download(
    "camnugent/california-housing-prices"
)

# Load CSV file
csv_path = os.path.join(dataset_path, "housing.csv")
df = pd.read_csv(csv_path)

print("First 5 rows of dataset:")
df.head()
```

```
Downloading to C:\Users\adity\.cache\kagglehub\datasets\camnugent\california-hous
ing-prices\1.archive...
100%|██████████| 400k/400k [00:01<00:00, 378kB/s]
Extracting files...
First 5 rows of dataset:
```

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

In [5]:
```
# Fill NaN values with column mean
df.fillna(df.mean(numeric_only=True), inplace=True)

print("Missing values after handling:")
df.isnull().sum()
```

```
Missing values after handling:
```

Out[5]:
```
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
dtype: int64
```

In [6]:
```
# Convert categorical column to numerical
df_encoded = pd.get_dummies(df, drop_first=True)

print("Columns after encoding:")
df_encoded.columns
```

```
Columns after encoding:
```

Out[6]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity_INLAND',
       'ocean_proximity_ISLAND', 'ocean_proximity_NEAR BAY',
       'ocean_proximity_NEAR OCEAN'],
      dtype='str')
```

In [7]:
```
# Target variable
y = df_encoded['median_house_value']

# Feature variables
X = df_encoded.drop('median_house_value', axis=1)
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

print("Training and testing data split completed.")
```

```
Training and testing data split completed.
```

In [9]:
```
linear_model = LinearRegression()

# Train the model
linear_model.fit(X_train, y_train)

# Predict on test data
y_pred_linear = linear_model.predict(X_test)

# Evaluate performance
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

print("--- Multiple Linear Regression Results ---")
print("Mean Squared Error (MSE):", mse_linear)
print("R² Score:", r2_linear)
```

```
--- Multiple Linear Regression Results ---
Mean Squared Error (MSE): 4904399775.949299
R² Score: 0.6257351821159687
```

In [10]:
```python
poly = PolynomialFeatures(degree=2, include_bias=False)

# Transform features
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Train polynomial regression model
poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train)

# Predict on test data
y_pred_poly = poly_model.predict(X_test_poly)

# Evaluate polynomial model
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print("--- Polynomial Regression (Degree 2) Results ---")
print("Mean Squared Error (MSE):", mse_poly)
print("R² Score:", r2_poly)
```

```
--- Polynomial Regression (Degree 2) Results ---
Mean Squared Error (MSE): 4426103431.798885
R² Score: 0.6622349582997737
```

In [11]:
```python
results = pd.DataFrame({
    "Model": [
        "Multiple Linear Regression",
        "Polynomial Regression (Degree 2)"
    ],
    "MSE": [
        mse_linear,
        mse_poly
    ],
    "R² Score": [
        r2_linear,
        r2_poly
    ]
})

results
```

Out[11]:

	Model	MSE	R² Score
0	Multiple Linear Regression	4.904400e+09	0.625735
1	Polynomial Regression (Degree 2)	4.426103e+09	0.662235

In [12]:
```python
print("Program executed successfully.")
```

```
Program executed successfully.
```