ID: cell-a571863327493058 Read-only

### Step 1: Select and Explain the Dataset

California Housing Prices Dataset

This dataset contains data on housing prices in California. It includes various features such as median income, housing median age, average rooms, etc., along with the corresponding median house value for different districts in California. The goal is to build a regression model to predict the median house value based on the given features.

ID: cell-014575f78d83739e Read-only

### Step 2: Read Data (1 point)

Read California Housing Prices datasetdataset into a Pandas DataFrame.

In [3]:

ID: cell-5ce183277081aa38 Autograded answer

```
import pandas as pd

# Load the California Housing Prices dataset
# data_url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.csv"
# housing_data = pd.read_csv(data_url)
housing_data = pd.read_csv('housing.csv')

housing_data.head()
```

Out[3]:

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

In [4]:

```
housing_data.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
```

In [5]:

Points: 1 ID: cell-0f7bf8d5311ac680 Autograder tests

```
#step 2
assert len(housing_data) == 20640, "Incorrect number of rows"
```

ID: cell-5ede6f97ca578e1a Read-only

### Step 3: Exploratory Data Analysis (EDA) and Numerical Results (3 points)

In [6]:

ID: cell-985e49e8cfb5ff4e Autograded answer

```
# Calculate summary statistics
median_income_mean = housing_data['median_income'].mean()
housing_median_age_max = housing_data['housing_median_age'].max()
total_rooms_sum = housing_data['total_rooms'].sum()

median_income_mean, housing_median_age_max, total_rooms_sum
```

Out[6]: (3.8706710029069766, 52.0, 54402150.0)

In [7]:

Points: 3 ID: cell-932f975bc5a37635 Autograder tests

```
#step3
assert round(median_income_mean, 2) == 3.87, "Incorrect mean median_income"
assert housing_median_age_max == 52, "Incorrect max housing_median_age"
assert total_rooms_sum == 54402150, "Incorrect total_rooms sum"
```

ID: cell-0709130f89512eb6 Read-only

## Step 4: Setup Experiment and Model Training (3 points)

In [8]:

ID: cell-c7ace6ade31e8be3 Autograded answer

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder

# Remove the target variable
X = housing_data.drop('median_house_value', axis=1)
y = housing_data['median_house_value']

# Handle categorical variables using one-hot encoding
categorical_features = ['ocean_proximity']
numeric_features = list(X.columns.difference(categorical_features))
preprocessor = ColumnTransformer(
    transformers=[
        ('num', SimpleImputer(strategy='median'), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Split 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)

# Preprocess and train the linear regression model
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', LinearRegression())
])

model.fit(X_train, y_train)
y_predict=model.predict(X_test.iloc[[100,200,-100],:])
y_predict
```

Out[8]:

```
array([134173.85407092, 171135.56880399, 215393.47592769])
```

In [9]:

Points: 3 ID: cell-8c31d16161235116 Autograder tests

```
#step4
def mycheck(yp,yt,error_percent=10):
    p=error_percent/100
    return yt*(1-p)<yp and yp < yt*(1+p)

assert len(X_train) == 16512, "Incorrect number of training samples"
assert len(X_test) == 4128, "Incorrect number of test samples"
assert mycheck(y_predict[0],134173.854), "Incorrect y_prediction[0]"
assert mycheck(y_predict[1],171135.568), "Incorrect y_prediction[1]"
assert mycheck(y_predict[2],215393.475), "Incorrect y_prediction[2]"
```

ID: cell-6e6b31589e3436b4 Read-only

## Step 5: Evaluation (3 points)

In [10]:

ID: cell-6510510e34498eab Autograded answer

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mae, mse, r2
```

Out[10]:

```
(50670.489235655565, 4908290571.346393, 0.6254382675296296)
```

In [11]:

Points: 3 ID: cell-e13ffa942cad9259 Autograder tests

```
#step5
def mycheck(yp,yt,error_percent=10):
    p=error_percent/100
    return yt*(1-p)<yp and yp < yt*(1+p)

assert  mycheck(mae,50670.489), "Incorrect Mean Absolute Error (MAE)"
assert mycheck(mse,4908290571.346), "Incorrect Mean Squared Error (MSE)"
assert mycheck(r2,0.625), "Incorrect R-squared (R2) score"
```