[8]:

```
import pandas as pd
a=pd.read_csv("/content/sample_data/california_housing_test.csv")
print(a)
a1=a.describe()
print(t1)
```

```
 longitude latitude housing_median_age total_rooms total_bedrooms \
0      -122.05    37.37                27.0      3885.0          661.0
1      -118.30    34.26                43.0      1510.0          310.0
2      -117.81    33.78                27.0      3589.0          507.0
3      -118.36    33.82                28.0        67.0           15.0
4      -119.67    36.33                19.0      1241.0          244.0
…        …      …                  …         …            …

2995   -119.86    34.42                23.0      1450.0          642.0
2996   -118.14    34.06                27.0      5257.0         1082.0
2997   -119.70    36.30                10.0       956.0          201.0
2998   -117.12    34.10                40.0        96.0           14.0
2999   -119.63    34.42                42.0      1765.0          263.0

     population households median_income
     median_house_value
0        1537.0      606.0        6.6085          344700.0

1         809.0      277.0        3.5990          176500.0

2        1484.0      495.0        5.7934          270500.0

3          49.0       11.0        6.1359          330000.0

4         850.0      237.0        2.9375           81700.0

…          …        …           …                …

2995     1258.0      607.0        1.1790          225000.0

2996     3496.0     1036.0        3.3906          237200.0

2997      693.0      220.0        2.2895           62000.0

2998       46.0       14.0        3.2708          162500.0

2999      753.0      260.0        8.5608          500001.0

[3000 rows x 9 columns] longitude     latitude
        housing_median_age  total_rooms \
count 3000.000000 3000.00000     3000.000000
                                 3000.000000
mean  -119.589200   35.63539      28.845333  2599.578667
```

```
std      1.994936    2.12967       12.555396  2155.593332
min   -124.180000   32.56000         1.000000     6.000000
25%    -121.810000  33.93000         18.000000 1401.000000
50%    -118.485000  34.27000         29.000000 2106.000000
75%    -118.020000  37.69000         37.000000 3129.000000
max    -114.490000    41.92000    52.000000    30450.000000
       total_bedrooms population households median_income
       \
count   3000.000000 3000.000000 3000.00000  3000.000000
mean      529.950667 1402.798667 489.91200    3.807272
std       415.654368 1030.543012 365.42271    1.854512
min         2.000000     5.000000    2.00000    0.499900
25%       291.000000  780.000000 273.00000    2.544000
50%       437.000000 1155.000000 409.50000    3.487150
75%       636.000000 1742.750000 597.25000    4.656475
max      5419.000000 11935.000000              15.000100
       4930.00000 median_house_value

count         3000.00000

mean        205846.27500

std         113119.68747

min          22500.00000

25%         121200.00000

50%         177650.00000

75%         263975.00000

max         500001.00000
```

[9]:

```python
#2
print("DATATYPE OF EACH COLUMN ")
print(a.dtypes)
print("")
print("SHAPE OF EACH COLUMN ")
for column in a.columns:
  print(f"{column} :{a[column].shape[0]}")
```

```
DATATYPE OF EACH COLUMN
longitude          float64
latitude           float64
housing_median_agefloat64
total_rooms        float64
total_bedrooms     float64
population         float64
households         float64
```

```
median_income     float64
median_house_valuefloat64
dtype: object
SHAPE OF EACH COLUMN
longitude :3000
latitude :3000
housing_median_age
:3000 total_rooms
:3000
total_bedrooms
:3000 population
:3000 households
:3000 median_income
:3000
median_house_value :3000
```

[10]:
```
#3
nullvalues=a.isnull()
print(nullvalues)
a_mean=a.fillna(a.mean)
print(a_mean)
```

```
 longitude latitude housing_median_age total_rooms total_bedrooms \
0        False    False              False       False          False
1        False    False              False       False          False
2        False    False              False       False          False
3        False    False              False       False          False
4        False    False              False       False          False
…         …        …                …          …            …

2995     False    False              False       False          False
2996     False    False              False       False          False
2997     False    False              False       False          False
2998     False    False              False       False          False
2999     False    False              False       False          False

     population households median_income median_house_value
0         False      False         False              False

1         False      False         False              False

2         False      False         False              False

3         False      False         False              False

4         False      False         False              False

…          …        …           …                …
```

```
2995     False     False       False            False
2996     False     False       False            False
2997     False     False       False            False
2998     False     False       False            False
2999     False     False       False            False
[3000 rows x 9 columns] longitude latitude housing_median_age
     total_rooms total_bedrooms \
0      -122.05   37.37            27.0     3885.0        661.0
1      -118.30   34.26            43.0     1510.0        310.0
2      -117.81   33.78            27.0     3589.0        507.0
3      -118.36   33.82            28.0       67.0         15.0
4      -119.67   36.33            19.0     1241.0        244.0
…        …     …              …        …          …
2995   -119.86   34.42            23.0     1450.0        642.0
2996   -118.14   34.06            27.0     5257.0       1082.0
2997   -119.70   36.30            10.0      956.0        201.0
2998   -117.12   34.10            40.0       96.0         14.0
2999   -119.63   34.42            42.0     1765.0        263.0

  population households median_income median_house_value
0          1537.0      606.0 6.6085      344700.0
1          809.0  277.0 3.5990      176500.0
2          1484.0      495.0 5.7934      270500.0
3          49.0   11.0  6.1359      330000.0
4          850.0  237.0 2.9375      81700.0
…        …        …          …              …
2995  1258.0     607.0 1.1790      225000.0
2996  3496.0     1036.0      3.3906      237200.0
2997  693.0 220.0 2.2895      62000.0  2998     46.0  14.0
     3.2708     162500.0
2999     753.0     260.0       8.5608         500001.0
[3000 rows x 9 columns]
```

[11]:

```python
#4
X = a.drop(columns=["median_house_value"])
y = a["median_house_value"]

print("Features (X):\n",  .head())
print("\nTarget (y):\n",  .head())
```

```
Features (X):
  longitude latitude housing_median_age total_rooms total_bedrooms
                                                                  \
0    -122.05     37.37 27.0  3885.0      661.0
1    -118.30     34.26 43.0  1510.0      310.0
2    -117.81     33.78 27.0  3589.0      507.0
3    -118.36     33.82 28.0  67.0  15.0
4    -119.67     36.33 19.0  1241.0      244.0

   population households median_income
0        1537.0   606.0 6.6085
1        809.0    277.0 3.5990
2        1484.0   495.0 5.7934
3        49.0     11.0  6.1359
4        850.0    237.0 2.9375

Target (y):
0     344700.0
1     176500.0
2     270500.0
3     330000.0
4     81700.0
Name: median_house_value, dtype: float64
```

[15]:

```python
#5
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size=0.2,␣
  ↪random_state=42)

print("Train set shape:", X_train.shape)
print("Test set shape:", X_test.shape)
```

```
Train set shape: (2400, 8)
```

```
Test set shape: (600, 8)
```

[13]:

```
#6
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
target_column = a.columns[-1]
features = a.drop(columns=[target_column])
target = a[target_column]
scaler = MinMaxScaler()
features_normalized = scaler.fit_transform(features)
features_normalized_df = pd.DataFrame(features_normalized, columns=features.
 ↪columns)
print("Normalized Features:")
print(features_normalized_df.head())
```

```
Normalized Features: longitude latitude housing_median_age
  total_rooms total_bedrooms \
0     0.219814 0.513889     0.509804   0.127414   0.121654 1
0.606811 0.181624     0.823529   0.049402   0.056858 2
0.657379 0.130342     0.509804   0.117691   0.093225 3
0.600619 0.134615     0.529412   0.002004   0.002400 4
0.465428 0.402778     0.352941   0.040566   0.044674

  population households median_income
0     0.128416   0.122565   0.421277
1     0.067393   0.055804   0.213728
2     0.123973   0.100041   0.365064
3   0.003688   0.001826     0.388684
4    0.070830    0.047687      0.168108
```

[14]:

```
#7
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)

print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Squared Error (MSE): 4586505886.68125
Mean Absolute Error (MAE): 49554.27620826821
```

```
Root Mean Squared Error (RMSE): 67723.74684467222
```

[16]:

```
#8
weights = model.coef_
intercept = model.intercept_  # Access the intercept value if needed

print("Coefficient values (weights):", weights)
print("Intercept:", intercept)
```

```
Coefficient values (weights): [-4.40099473e+04 -4.33583030e+04
1.14711666e+03
-7.88631396e+00
  9.85275637e+01 -4.05048347e+01 6.14349440e+01 3.95481370e+04]
Intercept: -3700204.0909373183
```