


```
import pandas as pd
file_path = '/content/housing.csv'
df = pd.read_csv(file_path)
print("First few rows of the data:")
print(df.head())
print("\nSummary statistics:")
print(df.describe())
print("\nDataFrame info:")
print(df.info())
```



	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

Summary statistics:

	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

```
DataFrame 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
None
```

```
import pandas as pd
file_path = '/content/housing.csv'
df = pd.read_csv(file_path)
print("Data types of each column:")
print(df.dtypes)
print("\nShape of the DataFrame:")
print(df.shape)
```

Data types of each column:

longitude	float64
latitude	float64
housing_median_age	float64

```

total_rooms      float64
total_bedrooms   float64
population       float64
households       float64
median_income    float64
median_house_value float64
ocean_proximity  object
dtype: object

```

```

Shape of the DataFrame:
(20640, 10)

```

```

import pandas as pd
file_path = '/content/housing.csv'
df = pd.read_csv(file_path)
print("Null values in the DataFrame:")
print(df.isnull().sum())
df_filled_zero = df.fillna(0)
df_filled_mean = df.fillna(df.mean())
print("\nDataFrame with null values filled with '0':")
print(df_filled_zero.head())
print("\nDataFrame with null values filled with the mean of each column:")
print(df_filled_mean.head())

```

```

Null values in the DataFrame:
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
dtype: int64

```

DataFrame with null values filled with '0':

```

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

```

DataFrame with null values filled with the mean of each column:

```

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

```

```

<ipython-input-4-462ed931f106>:7: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version,
df_filled_mean = df.fillna(df.mean())

```

```

df=df.fillna(0)
y=df['median_house_value']
x=df.drop('median_house_value',axis=1)
x1=x.drop('ocean_proximity',axis=1)
print(x1)

```

```

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
...
20635	-121.09	39.48	25.0	1665.0	374.0
20636	-121.21	39.49	18.0	697.0	150.0
20637	-121.22	39.43	17.0	2254.0	485.0
20638	-121.32	39.43	18.0	1860.0	409.0
20639	-121.24	39.37	16.0	2785.0	616.0

	population	households	median_income
0	322.0	126.0	8.3252
1	2401.0	1138.0	8.3014
2	496.0	177.0	7.2574
3	558.0	219.0	5.6431
4	565.0	259.0	3.8462
...
20635	845.0	330.0	1.5603
20636	356.0	114.0	2.5568
20637	1007.0	433.0	1.7000
20638	741.0	349.0	1.8672
20639	1387.0	530.0	2.3886

[20640 rows x 8 columns]

print(y)

0	452600.0
1	358500.0
2	352100.0
3	341300.0
4	342200.0
...	...
20635	78100.0
20636	77100.0
20637	92300.0
20638	84700.0
20639	89400.0

Name: median_house_value, Length: 20640, dtype: float64

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x1,y,test_size=0.20,random_state=30)

print(x_train)

print(x_test)

print(y_train)

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
7186	-118.18	34.03	39.0	609.0	145.0	
7686	-118.10	33.93	35.0	1622.0	302.0	
6332	-117.95	33.99	24.0	1219.0	177.0	
14192	-117.07	32.69	20.0	2192.0	406.0	
6611	-118.11	34.18	52.0	3571.0	510.0	
...	
500	-122.27	37.85	52.0	1974.0	426.0	
12077	-117.64	33.87	2.0	17470.0	2727.0	
15277	-117.34	33.06	17.0	2718.0	518.0	
4517	-118.20	34.04	44.0	1399.0	386.0	
5925	-117.80	34.15	14.0	7876.0	1253.0	

	population	households	median_income
7186	690.0	134.0	2.9167
7686	845.0	284.0	4.5769
6332	610.0	185.0	6.7978
14192	1766.0	393.0	4.0921
6611	1434.0	490.0	5.9009
...
500	875.0	363.0	1.5817
12077	5964.0	1985.0	6.2308
15277	815.0	403.0	4.3182
4517	1419.0	373.0	1.8224
5925	3699.0	1162.0	5.5423

[16512 rows x 8 columns]

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
19449	-121.03	37.68	20.0	3204.0	625.0	
10452	-117.66	33.46	26.0	2073.0	370.0	

18982	-122.01	38.26	12.0	4132.0	710.0
8187	-118.11	33.78	16.0	3985.0	567.0
15759	-122.44	37.77	52.0	2994.0	736.0
...
12704	-121.41	38.58	18.0	6955.0	1882.0
18742	-122.34	40.57	26.0	2187.0	472.0
19142	-122.69	38.32	15.0	2536.0	414.0
1027	-120.55	38.46	16.0	1443.0	249.0
17830	-121.85	37.41	25.0	1837.0	278.0

	population	households	median_income
19449	2016.0	605.0	2.6567
10452	952.0	340.0	5.0877
18982	2087.0	633.0	4.5987
8187	1327.0	564.0	7.9767
15759	1428.0	700.0	3.0766
...
12704	2803.0	1740.0	3.0890
18742	1339.0	463.0	2.0395
19142	1400.0	426.0	5.6613
1027	435.0	181.0	3.2031
17830	1006.0	271.0	6.6842

[4128 rows x 8 columns]

7186 145800.0

7686 186100.0

6332 325000.0

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaling=MinMaxScaler()
```

```
housing_scaled_df=scaling.fit_transform(df[['median_house_value', 'population']])
```

```
housing_normalized_df=pd.DataFrame(housing_scaled_df,columns=['median_house_value', 'population'])
```

```
housing_normalized_df.head()
```

```
print(y_test)
```

19449 110400.0

10452 288100.0

18982 139700.0

8187 500001.0

15759 438900.0

...

12704 141400.0

18742 67900.0

19142 172400.0

1027 129200.0

17830 265300.0

Name: median_house_value, Length: 4128, dtype: float64

```
import numpy as np
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
```

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

```
import math
```

```
lin_reg = LinearRegression()
```

```
lin_reg.fit(x_train, y_train)
```

```
y_pred = lin_reg.predict(x_test)
```

```
mse = mean_squared_error(y_test, y_pred)
```

```
mae = mean_absolute_error(y_test, y_pred)
```

```
rmse = math.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): 5371308873.230868

Mean Absolute Error (MAE): 52486.39360780328

Root Mean Squared Error (RMSE): 73289.2138942073

```
coefficients = lin_reg.coef_
```

```
intercept = lin_reg.intercept_
```

```
print("Intercept:", intercept)
```

```
print("Coefficient (Weight):", coefficients[0])
```

```
print(lin_reg.coef_)
```

Intercept: -3466246.7043957342

Coefficient (Weight): -41577.30377414892

```
[-4.15773038e+04 -4.18177918e+04  1.14464383e+03 -5.01967848e+00  
 4.92067893e+01 -4.44012137e+01  1.16069437e+02  3.89419169e+04]
```

```
import numpy as np  
from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression  
from sklearn.metrics import mean_squared_error, mean_absolute_error  
import math  
lin_reg = LinearRegression()  
lin_reg.fit(x_train, y_train)  
y_pred = lin_reg.predict(x_train)  
mse = mean_squared_error(y_train, y_pred)  
mae = mean_absolute_error(y_train, y_pred)  
rmse = math.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): 4743701682.935274  
Mean Absolute Error (MAE): 50605.64822763461  
Root Mean Squared Error (RMSE): 68874.53580921816
```