In [26]:

```
# import Important Library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1. Load the data :

• Read the “housing.csv” file from the folder into the program. • Print first few rows of this data. • Extract input (X) and output (Y) data from the dataset.

In [27]:

```
#Read the “housing.csv” file from the folder into the program
dataset = pd.read_csv("housing.csv")
# Print first few rows of this data
dataset.head()
```

Out[27]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
0	-122.23	37.88	41	880	129.0	322	1
1	-122.22	37.86	21	7099	1106.0	2401	11
2	-122.24	37.85	52	1467	190.0	496	1
3	-122.25	37.85	52	1274	235.0	558	2
4	-122.25	37.85	52	1627	280.0	565	2

Handle missing values :

• Fill the missing values with the mean of the respective column

In [28]:

```
# check for missing value
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null int64
total_rooms           20640 non-null int64
total_bedrooms        20433 non-null float64
population            20640 non-null int64
households            20640 non-null int64
median_income         20640 non-null float64
ocean_proximity       20640 non-null object
median_house_value    20640 non-null int64
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

In [29]:

```
dataset.isnull().sum()
```

Out[29]:

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

Comment : there is missing value in total_bedroom

In [30]:

```
#Fill the missing value using mean of total_bedroom column
dataset['total_bedrooms']=dataset['total_bedrooms'].fillna(dataset['total_bedroo
ms'].mean())
dataset.isnull().sum()
```

Out[30]:

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
ocean_proximity       0
median_house_value    0
dtype: int64
```

3. Encode categorical data :

- Convert categorical column in the dataset to numerical data

In [31]:

```
df1 =pd.get_dummies(dataset['ocean_proximity'])
```

In [32]:

```
df1.head()
```

Out[32]:

	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0

In [33]:

```
dataset =pd.concat([dataset,df1],axis =1)
```

In [34]:

```
dataset.head()
```

Out[34]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41	880	129.0	322	1
1	-122.22	37.86	21	7099	1106.0	2401	11
2	-122.24	37.85	52	1467	190.0	496	1
3	-122.25	37.85	52	1274	235.0	558	2
4	-122.25	37.85	52	1627	280.0	565	2

In [35]:

```
dataset=dataset.drop('ocean_proximity',axis=1)
```

In [36]:

```
dataset.head()
```

Out[36]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41	880	129.0	322	1
1	-122.22	37.86	21	7099	1106.0	2401	11
2	-122.24	37.85	52	1467	190.0	496	1
3	-122.25	37.85	52	1274	235.0	558	2
4	-122.25	37.85	52	1627	280.0	565	2

In [37]:

```
dataset.tail()
```

Out[37]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hous
20635	-121.09	39.48	25	1665	374.0	845	
20636	-121.21	39.49	18	697	150.0	356	
20637	-121.22	39.43	17	2254	485.0	1007	
20638	-121.32	39.43	18	1860	409.0	741	
20639	-121.24	39.37	16	2785	616.0	1387	

In [38]:

```
# define feature variable & lable variable
x=dataset.drop('median_house_value',axis=1)
x.head()
y=dataset['median_house_value']
y.head()
```

Out[38]:

```
0    452600
1    358500
2    352100
3    341300
4    342200
Name: median_house_value, dtype: int64
```

4. Split the dataset :

• Split the data into 80% training dataset and 20% test dataset.

In [39]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state
=1)
```

5. Standardize data :

• Standardize training and test datasets

In [51]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
scaled_x_train =sc.fit_transform(x_train)
scaled_x_test = sc.transform(x_test)
```

6. Perform Linear Regression :

• Perform Linear Regression on training data. • Predict output for test dataset using the fitted model. • Print root mean squared error (RMSE) from Linear Regression. [HINT: Import mean_squared_error from sklearn.metrics]

In [53]:

```
from sklearn.linear_model import LinearRegression
lr =LinearRegression()
#Perform Linear Regression on training data.
lr.fit(scaled_x_train,y_train)
```

Out[53]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [54]:

```
#Predict output for test dataset using the fitted model
y_pred=lr.predict(scaled_x_test)
y_pred
```

Out[54]:

```
array([243626.05897437,  93442.21732292, 247630.00077113, ...,
       284944.56343132, 266458.51436442, 137361.4394673 ])
```

In [64]:

```
 #Print root mean squared error (RMSE) from Linear Regression. [ HINT: Import me
an_squared_error from sklearn.metrics]
from math import sqrt
from sklearn.metrics import mean_squared_error
rmse = sqrt(mean_squared_error(y_test, y_pred))

print(rmse)
```

```
68949.62451074278
```

Question :

I don't understand why i am getting rmse value this much high ?

i think RMSE value should be between 0 & 1 ;

please explain me

In [61]:

```
lr.coef_
```

Out[61]:

```
array([-53274.56856834, -53787.3951718 ,  13459.99994478, -11966.783
01079,
        32722.17764512, -43216.91502076,  27280.31093161,  74374.453
93177,
         5872.31094866, -12774.83915651,   2921.178774  ,   3030.786
85332,
         5988.26842074])
```

In [62]:

```
lr.intercept_
```

Out[62]:

```
207735.06419573637
```

In [ ]:

```

```