In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df=pd.read_csv("housing.csv")
```

In [3]:
```
df
```

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
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	INLAND
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	INLAND
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	INLAND
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	INLAND
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	INLAND

20640 rows × 10 columns

In [4]:
```
# to find number of rows and columns
df.shape
```

Out[4]:
```
(20640, 10)
```

In [5]:
```
#general info
df.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 [6]:
```
#show top 5 rows
df.head()
```

Out[6]:

	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 [7]:
```
#extract all columns of dataset
df.columns
```

Out[7]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

In [8]:
```
#check for all null values
df.isna().sum()
```

Out[8]:
```
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
```

In [9]:
```
df.dropna(inplace=True)
```

In [10]:
```
df.isna().sum()
```

Out[10]:
```
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
```

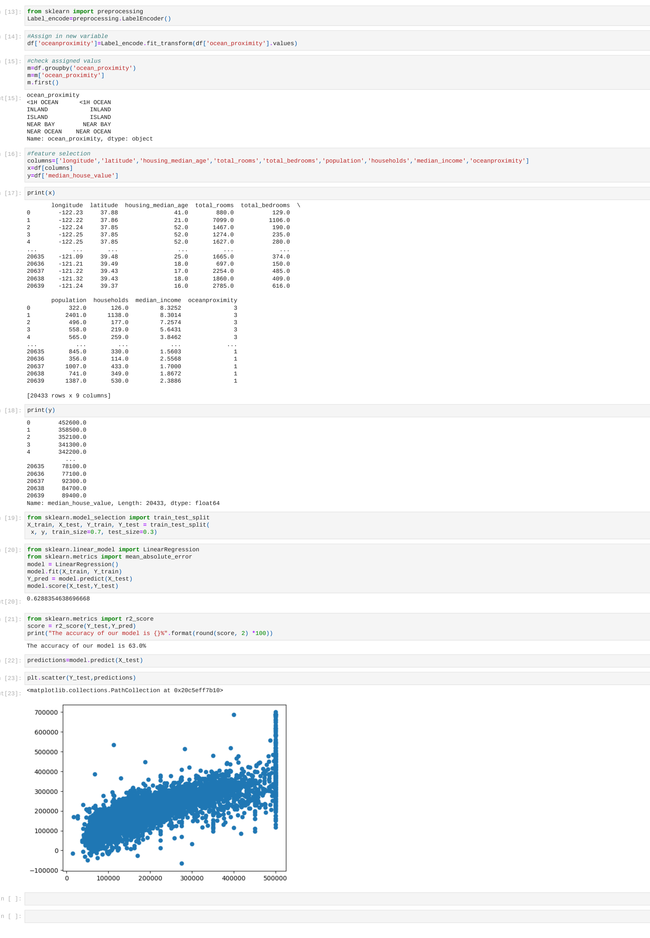
In [11]:
```
df.describe()
```

Out[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000
mean	-119.570689	35.633221	28.633094	2636.504233	537.870553	1424.946949	499.433465	3.871162	206864.413155
std	2.003578	2.136348	12.591805	2185.269567	421.385070	1133.208490	382.299226	1.899291	115435.667099
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1450.000000	296.000000	787.000000	280.000000	2.563700	119500.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.536500	179700.000000
75%	-118.010000	37.720000	37.000000	3143.000000	647.000000	1722.000000	604.000000	4.744000	264700.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

In [12]:
```
sns.pairplot(df)
```

```
C:\ProgramData\anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[12]:


In [13]:
```
from sklearn import preprocessing
Label_encode=preprocessing.LabelEncoder()
```

In [14]:
```
#Assign in new variable
df['oceanproximity']=Label_encode.fit_transform(df['ocean_proximity'].values)
```

In [15]:
```
#check assigned value
m=df.groupby('ocean_proximity')
m=m['ocean_proximity']
m.first()
```

Out[15]:
```
ocean_proximity
<1H OCEAN      <1H OCEAN
INLAND          INLAND
ISLAND          ISLAND
NEAR BAY      NEAR BAY
NEAR OCEAN    NEAR OCEAN
Name: ocean_proximity, dtype: object
```

In [16]:
```
#feature selection
columns=['longitude','latitude','housing_median_age','total_rooms','total_bedrooms','population','households','median_income','oceanproximity']
x=df[columns]
y=df['median_house_value']
```

In [17]:
```
print(x)
```

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

[20433 rows x 9 columns]
```

In [18]:
```
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: 20433, dtype: float64
```

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```

In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
model = LinearRegression()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
model.score(X_test,Y_test)
```

Out[20]:
```
0.6288354638696668
```

In [21]:
```
from sklearn.metrics import r2_score
score = r2_score(Y_test,Y_pred)
print("The accuracy of our model is {}%".format(round(score, 2) *100))
```

```
The accuracy of our model is 63.0%
```

In [22]:
```
predictions=model.predict(X_test)
```

In [23]:
```
plt.scatter(Y_test,predictions)
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x20c5efff7b10>
```


In [ ]:
```

```

In [ ]:
```

```