```
#Import Necessary Libraries:
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import
LinearRegression,Ridge,Lasso,ElasticNet
from sklearn.tree import DecisionTreeRegressor
import statsmodels.formula.api as smf

from sklearn.metrics import mean_squared_error,r2_score
from math import sqrt

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

```
C:\Users\Nageswaran B\Anaconda3\lib\site-packages\statsmodels\tools\
_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use
the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Read the "housing.csv" file from the folder into the program.

```
df_house = pd.read_excel("C:/Users/Nageswaran
B/Documents/machine_learning/machine_learning_project/simplilearn/
hoursing_price_prediction/1553768847_housing.xlsx")
```

Print first few rows of this data.

```
df_house.head()
```

```
   longitude  latitude  housing_median_age  total_rooms
total_bedrooms  \
0    -122.23     37.88                  41          880
129.0
1    -122.22     37.86                  21         7099
1106.0
2    -122.24     37.85                  52         1467
190.0
3    -122.25     37.85                  52         1274
235.0
4    -122.25     37.85                  52         1627
280.0

   population  households  median_income ocean_proximity
```

```
median_house_value
0         322         126         8.3252        NEAR BAY
452600
1        2401        1138         8.3014        NEAR BAY
358500
2         496         177         7.2574        NEAR BAY
352100
3         558         219         5.6431        NEAR BAY
341300
4         565         259         3.8462        NEAR BAY
342200
```

```
df_house.columns
```

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

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

```
df_house.isnull().sum()
```

```
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
```

```
df_house.total_bedrooms=df_house.total_bedrooms.fillna(df_house.total_
bedrooms.mean())
df_house.isnull().sum()
```

```
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
```

Encode categorical data :

```
le = LabelEncoder()
df_house['ocean_proximity']=le.fit_transform(df_house['ocean_proximity
'])
```

Standardize data :

```
# Get column names first
names = df_house.columns
# Create the Scaler object
scaler = StandardScaler()
# Fit your data on the scaler object
scaled_df = scaler.fit_transform(df_house)
scaled_df = pd.DataFrame(scaled_df, columns=names)
scaled_df.head()
```

```
   longitude  latitude  housing_median_age  total_rooms
total_bedrooms  \
0  -1.327835  1.052548            0.982143    -0.804819       -
0.975228
1  -1.322844  1.043185           -0.607019     2.045890
1.355088
2  -1.332827  1.038503            1.856182    -0.535746       -
0.829732
3  -1.337818  1.038503            1.856182    -0.624215       -
0.722399
4  -1.337818  1.038503            1.856182    -0.462404       -
0.615066

   population  households  median_income  ocean_proximity
median_house_value
0   -0.974429   -0.977033       2.344766         1.291089
2.129631
1    0.861439    1.669961       2.332238         1.291089
1.314156
2   -0.820777   -0.843637       1.782699         1.291089
1.258693
3   -0.766028   -0.733781       0.932968         1.291089
1.165100
4   -0.759847   -0.629157      -0.012881         1.291089
1.172900
```

Extract input (X) and output (Y) data from the dataset.

```
X_Features=['longitude', 'latitude', 'housing_median_age',
'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity']
X=scaled_df[X_Features]
Y=scaled_df['median_house_value']
```

```
print(type(X))
print(type(Y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

```
print(df_house.shape)
print(X.shape)
print(Y.shape)
```

```
(20640, 10)
(20640, 9)
(20640,)
```

Split the dataset :

```
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,rando
m_state=1)

print (x_train.shape, y_train.shape)
print (x_test.shape, y_test.shape)
```

```
(16512, 9) (16512,)
(4128, 9) (4128,)
```

Perform Linear Regression :

```
linreg=LinearRegression()
linreg.fit(x_train,y_train)
```

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

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

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

```
y_predict = linreg.predict(x_test)

print(sqrt(mean_squared_error(y_test,y_predict)))
print((r2_score(y_test,y_predict)))
```

```
0.6056598120301221
0.6276223517950296
```

Perform Linear Regression with one independent variable :

```
x_train_Income=x_train[['median_income']]
x_test_Income=x_test[['median_income']]

print(x_train_Income.shape)
print(y_train.shape)
```

```
(16512, 1)
(16512,)
```

```
linreg=LinearRegression()
linreg.fit(x_train_Income,y_train)
y_predict = linreg.predict(x_test_Income)
```

```
#print intercept and coefficient of the linear equation
print(linreg.intercept_, linreg.coef_)
print(sqrt(mean_squared_error(y_test,y_predict)))
print((r2_score(y_test,y_predict)))
```

```
0.005623019866893162 [0.69238221]
0.7212595914243148
0.47190835934467734
```

```
#plot least square line
scaled_df.plot(kind='scatter',x='median_income',y='median_house_value'
)
plt.plot(x_test_Income,y_predict,c='red',linewidth=2)
```

```
[<matplotlib.lines.Line2D at 0x19fd77a9668>]
```