```
import pandas as pd
data = pd.read_csv("Housing_Modified_prepared.csv")
# select Y dependent variable and X Independent variable
Y = data["price"]
independent_variables = data.columns
independent_variables = independent_variables.delete(0)
X = data[independent_variables]
```

```
# Train the ordinary least square regression
import sklearn.linear_model as lm
import statsmodels.api as sm

lr =  lm.LinearRegression()
lr.fit(X,Y)
```

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

```
# Train the model using  ordinary least square
import statsmodels.api as sm
ols = sm.OLS(Y,X)
ols.fit()
```

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f8d195690f0>
```

```
# predict using the linear regression model
Ypred = lr.predict(X)
Ypred
```

```
array([ 66037.97567237,  41391.15145679,  39889.63013056,  63689.08733076,
        49760.42646619,  66387.12316802,  77632.14629398,  56904.45482064,
        56023.86666406, 103938.00821564,  99726.64585232,  31157.06959629,
        41623.60380274,  32563.51670553,  39972.63026876,  51133.80494099,
        44854.73866879,  62424.61337517,  37608.6813576 ,  63197.78116303,
        79405.33724015,  73208.66262212,  62412.99103875,  41157.09546077,
        44795.60230807,  44401.7941982 ,  40681.89086277,  62880.87825234,
        45621.96346675,  64186.14252379,  38382.60662078,  70961.50102802,
        41391.15145679,  43164.30294183,  56317.37217768,  64286.85135531,
        46957.01869128,  79676.05693673,  92035.03513952,  51476.9590496 ,
        54328.17891601,  69709.45412264,  58126.56700477, 106846.95011357,
        40855.49122811,  42019.50981801,  31298.92171509,  39865.18604074,
        54885.72177194,  47475.00370338,  72389.85989468,  44174.39533413,
        86286.28655318,  72450.57356293,  74962.59040738,  40043.55632816,
        30861.29127989,  91817.81745181,  51198.62182196,  44106.01588179,
        48373.65352469,  44050.87868435,  63506.6647019 ,  52688.0690694 ,
        66895.22792026,  67732.44549387,  71553.53166708,  70813.86234556,
        42016.80851606,  66025.78029604,  50217.70824761,  52218.12864881,
        32481.55847856,  44371.98860455,  52430.90682701,  33627.40759655,
        40210.89530326,  66789.41160837,  66760.78503778,  72090.53141479,
        63264.10375157,  40764.29513955,  48538.8945944 ,  55574.56838533,
        57131.57988042,  68764.87251847,  66405.13932567,  77140.91563658,
        42089.67749104,  48127.68511957,  60492.20840637,  75695.07386445,
        94826.6780363 , 114310.46230412,  84427.07072599,  51320.79977303,
        66784.60055397,  76481.81509775,  44129.28379785,  37259.70849664,
        77286.30764117,  81421.19103101,  77688.36739888,  85495.57535796,
        76123.24965727,  43518.93323884,  46343.29813957,  62980.84215185,
        70533.02549774,  50468.24507982,  40351.19802784,  36476.52405142,
        65959.29400883,  70619.41881606,  80900.38362634,  67692.38743948,
        70250.05219619,  86445.76405087,  74876.76453708,  90356.99677497,
        47054.50164916,  93243.46087781,  42809.67264483,  57003.82413313,
        51513.92235698,  79701.10760656,  99610.46976187, 102260.46177217,
        92942.97754023,  91877.27312391,  50926.99166315,  49248.15518842,
        46283.99441659,  51180.86748559,  50861.70021828,  90398.98893541,
        51092.23273286,  54512.47244611,  35366.43557092,  48893.52489141,
        46233.79766385,  54689.8061681 ,  81662.85760108,  36476.52405142,
        62950.74917282,  66797.48392407,  59216.87859348,  89521.99156955,
        88769.66703658,  53739.88588163,  52074.46300472,  39546.01877343,
        70562.34899237,  45918.62955981,  42336.57917992,  68666.08101744,
        64152.39736852,  54816.36605599,  44779.20949193,  42978.84004847,
        63686.01815415,  72646.75358786,  32669.90579463,  50814.02517265,
        58255.3669357 ,  61942.45501941,  53632.89339598,  38967.59135834,
        53601.93615731,  45353.03460989,  45531.40489731,  42096.86120976,
        30967.68036899,  41915.02198678,  51813.85011022,  84124.59127842,
        56576.9282577 ,  90465.7407982 ,  69697.34867547,  43718.58219933,
        40402.18578844,  49744.63760423,  44428.89793227,  48042.41217859,
        44699.47295216,  62967.47671638,  53977.26340216,  47325.07666905,
        48677.91740159,  62329.99090055,  56957.62654366,  52711.90976268,
        45493.0732034 ,  77973.81473554,  38483.18302132,  33408.97198229,
        43081.30280364,  61409.76305379,  34455.13135847,  53707.57589624,
        46049.46273874,  71616.51798454,  47324.1888924 ,  63039.69413246,
        55949.51161878,  59259.88367751,  37152.26426862,  48084.74303577,
        43035.74256569,  57023.34635175,  34125.22953139,  30093.17870526,
        31925.18217091,  45636.73884749,  58328.6106232 ,  40079.01935786,
       118660.56061707,  40077.96421894,  50874.4730818 ,  54599.35789008,
        39971.57512984,  81197.68079391,  54734.19023228,  69273.10563378,
        59718.68855814,  75502.42159635,  67313.56566322,  52442.95808428,
        80341.17477609,  71364.11107141,  80935.90495334,  96808.76393155,
        49248.15518842.  32764.98001162.  44401.7941982 .  50417.45285897.
```

```
# calculate r2 score for Linear Regression Model using
# sum of squared Total(SST) and Sum of Squared Residual (SSR)

# calculate sum of Squared Total (Sum of (y - ymean)^2)
Ymean = Y.mean()
print("Mean of Actual dependent variable", Ymean)

# Calculate square of y - ymean using numpy library
import numpy as np
squared_total = np.square(Y - Ymean)
print("Squared Total is\n ", squared_total)
sst = squared_total.sum()
print("Sum of squared total", sst)
```

```
Mean of Actual dependent variable 68121.59706959708
Squared Total is
  0      6.823378e+08
1      8.774390e+08
2      3.467639e+08
3      5.808874e+07
4      5.071714e+07
           ...
```

```
     541    5.465497e+08
     542    6.696917e+08
     543    1.216503e+09
     544    1.360017e+09
     545    1.360017e+09
     Name: price, Length: 546, dtype: float64
     Sum of squared total 388602785841.3553
```

```
# calculate the sum of squared residual (SSR)
squared_residual = np.square(Ypred - Ymean)
ssr = squared_residual.sum()
print("Sum of squared total", ssr)
```

```
     Sum of squared total 261577714196.88992
```

```
# calculate the value of R2 score(Rsquared)
r2score = ssr/sst
print("R2 score is ", r2score)
```

```
     R2 score is  0.6731236206414574
```

```
# calculate Root Mean Squared Errors(RMSE)
error = Y-Ypred
print("Errors\n",error)

#take a square of errors
squared_errors = np.square(error)
print("Squared errors \n",squared_errors)
```

```
     Errors
      0     -24037.975672
     1      -2891.151457
     2       9610.369869
     3      -3189.087331
     4      11239.573534
                ...
     541    -5710.767132
     542    -2955.046870
     543    -2711.159701
     544    12402.731722
     545    31249.574017
     Name: price, Length: 546, dtype: float64
     Squared errors
      0      5.778243e+08
     1      8.358757e+06
     2      9.235921e+07
     3      1.017028e+07
     4      1.263280e+08
                ...
     541    3.261286e+07
     542    8.732302e+06
     543    7.350387e+06
     544    1.538278e+08
```

```
545    9.765359e+08
Name: price, Length: 546, dtype: float64
```

```
# calculate mean squared errors
sse = squared_errors.sum()
mse = sse/len(Y)
print("Mean squared errors\n", mse)
print("Number of elements", len(Y))
rmse = np.sqrt(mse)
print("root mean squared errors", rmse)
```

```
Mean squared errors
 232646651.36349452
Number of elements 546
root mean squared errors 15252.758811555846
```

```
# calculate Mean Absolute Errors
# Calculate absolute errors
absolute_errors = abs(Y-Ypred)
sae = absolute_errors.sum()
mae = sae/len(Y)
print("Mean Absolute Error", mae)
```

```
Mean Absolute Error 11239.02923948371
```

```
# calculate R2Score, RMSE and MAE using sklearn
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
r2score = r2_score(Y,Ypred)
print("R2 score ",r2score)
```

```
R2 score  0.6731236206414506
```