```
# data gathering
import pandas as pd
data = pd.read_csv("/Housing_Modified (1).csv")
```

```
# shpae of data
print(data.shape)
print("data has %d rows and %d columns"%data.shape)
```

```
(546, 12)
data has 546 rows and 12 columns
```

```
#Check top 5 values in data
data.head(5)
#check bottom 5 values in data
data.tail(5)
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	ga
541	91500.0	4800	3	2	four	yes	yes	no	
542	94000.0	6000	3	2	four	yes	no	no	
543	103000.0	6000	3	2	four	yes	yes	no	
544	105000.0	6000	3	2	two	yes	yes	no	
545	105000.0	6000	3	1	two	yes	no	no	

```
#Access any random row from data
columns=data.columns
data["price"][2]
```

```
49500.0
```

```
data[["price","lotsize","airco"]]
```

	price	lotsize	airco
0	42000.0	5850	no
1	38500.0	4000	no
2	49500.0	3060	no
3	60500.0	6650	no
4	61000.0	6360	no
...	...	...	...
541	91500.0	4800	yes
542	94000.0	6000	yes
543	103000.0	6000	yes
544	105000.0	6000	yes
545	105000.0	6000	yes

546 rows × 3 columns

```
# what happens in data ?
import matplotlib.pyplot as plt
plt.scatter(data["lotsize"],data["price"],color="black")
plt.xlabel("lot size(Area of house)")
plt.ylabel("house prices(Area of house)")
plt.title("house prices vs lotsize")
```

```
Text(0.5, 1.0, 'house prices vs lotsize')
```


```
#why did it happens?
data.corr() #calculate correlations
```

	price	lotsize	bedrooms	bathrms	garagepl
price	1.000000	0.535796	0.366447	0.516719	0.383302
lotsize	0.535796	1.000000	0.151851	0.193833	0.352872
bedrooms	0.366447	0.151851	1.000000	0.373769	0.139117
bathrms	0.516719	0.193833	0.373769	1.000000	0.178178
garagepl	0.383302	0.352872	0.139117	0.178178	1.000000

```
data.head(3)
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw
0	42000.0	5850	3	1	two	yes	no	yes	no
1	38500.0	4000	2	1	one	yes	no	no	no
2	49500.0	3060	3	1	one	yes	no	no	no

```
# Start
import pandas as pd
data = pd.read_csv("/Housing_Modified (1).csv")
data.head(3)
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw
0	42000.0	5850	3	1	two	yes	no	yes	no
1	38500.0	4000	2	1	one	yes	no	no	no
2	49500.0	3060	3	1	one	yes	no	no	no

```
# stage 1 Selection
# selection a dependent and independent variables
# check the influence
print(data.columns)
data.corr()
```

```
Index(['price', 'lotsize', 'bedrooms', 'bathrms', 'stories', 'driveway',
       'recroom', 'fullbase', 'gashw', 'airco', 'garagepl', 'prefarea'],
      dtype='object')
```

	price	lotsize	bedrooms	bathrms	garagepl
price	1.000000	0.535796	0.366447	0.516719	0.383302
lotsize	0.535796	1.000000	0.151851	0.193833	0.352872
bedrooms	0.366447	0.151851	1.000000	0.373769	0.139117
bathrms	0.516719	0.193833	0.373769	1.000000	0.178178
garagepl	0.383302	0.352872	0.139117	0.178178	1.000000

```
data.columns
temp_data=data[["stories","driveway","recroom","fullbase","gashw","airco","prefarea"]]
temp_data.head(3)
print("stories" ,temp_data["stories"].unique())
print("driveway", temp_data["driveway"].unique())
```

```
stories ['two' 'one' 'three' 'four']
driveway ['yes' 'no']
```

```
# stage 2 - preprocessing
# a.missing Data
# b.convert text to Numbers
```

```
#check Missing data
data.isna()
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashu
0	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	..
541	False	False	False	False	False	False	False	False	Fals
542	False	False	False	False	False	False	False	False	Fals
543	False	False	False	False	False	False	False	False	Fals
544	False	False	False	False	False	False	False	False	Fals
545	False	False	False	False	False	False	False	False	Fals

546 rows × 12 columns

```
# what to do if data is missing ?
# a. drop the null values
data_withoutNA =data.dropna()
# b. fil misising values with mean,median,max,min
# check the mean, median,max and min value of bedrooms
print("max of bedrooms",data["bedrooms"].max())
print("min of bedrooms",data["bedrooms"].min())
print("mean of bedrooms",round(data["bedrooms"].mean()))
print("median of bedrooms",round(data["bedrooms"].median()))
```

```
max of bedrooms 6
min of bedrooms 1
mean of bedrooms 3
median of bedrooms 3
```

```
# fill the missing value with max,min,mean,median
data_clean= data.fillna(data["bedrooms"].mean())
data_clean
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	ga
0	42000.0	5850	3	1	two	yes	no	yes	
1	38500.0	4000	2	1	one	yes	no	no	
2	49500.0	3060	3	1	one	yes	no	no	
3	60500.0	6650	3	1	two	yes	yes	no	
4	61000.0	6360	2	1	one	yes	no	no	
...	...	...	...	...	...	...	...	...	
541	91500.0	4800	3	2	four	yes	yes	no	
542	94000.0	6000	3	2	four	yes	no	no	
543	103000.0	6000	3	2	four	yes	yes	no	
544	105000.0	6000	3	2	two	yes	yes	no	
545	105000.0	6000	3	1	two	yes	no	no	

546 rows × 12 columns

```
# convert data into numbers
# a.convert binary category to number(0/1)
# label Binarizer
import sklearn.preprocessing as pp
print(temp_data.columns)

lb =pp.LabelBinarizer()
data.driveway =lb.fit_transform(data.driveway)
data.recroom =lb.fit_transform(data.recroom)
data.fullbase =lb.fit_transform(data.fullbase)
data.gashw =lb.fit_transform(data.gashw)
data.airco =lb.fit_transform(data.airco)
data.prefarea =lb.fit_transform(data.prefarea)
data.head(3)
```

```
Index(['stories', 'driveway', 'recroom', 'fullbase', 'gashw', 'airco',
       'prefarea'],
      dtype='object')
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw
0	42000.0	5850	3	1	two	1	0	1	0
1	38500.0	4000	2	1	one	1	0	0	0
2	49500.0	3060	3	1	one	1	0	0	0

```
#convert n-category into numbers using one Hot Encoding
stories_data = pd.get_dummies(data["stories"],prefix="stories")
data=pd.concat([data,stories_data],axis=1)
data.head(3)
```

	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw
0	42000.0	5850	3	1	two	1	0	1	0
1	38500.0	4000	2	1	one	1	0	0	0
2	49500.0	3060	3	1	one	1	0	0	0

```
data.columns
```

```
Index(['price', 'lotsize', 'bedrooms', 'bathrms', 'stories', 'driveway',
       'recroom', 'fullbase', 'gashw', 'airco', 'garagepl', 'prefarea',
       'stories_four', 'stories_one', 'stories_three', 'stories_two'],
      dtype='object')
```

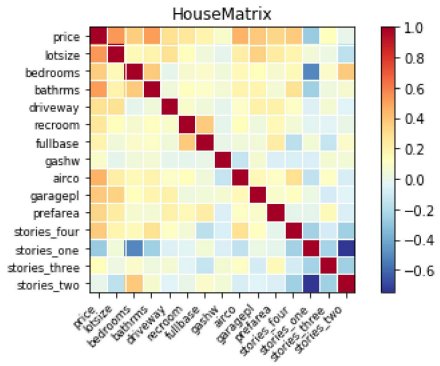
```
print("before delete",data.columns)
# delete stories columns
del data["stories"]
print("after delete", data.columns)
```

```
before delete Index(['price', 'lotsize', 'bedrooms', 'bathrms', 'stories', 'driveway',
       'recroom', 'fullbase', 'gashw', 'airco', 'garagepl', 'prefarea',
       'stories_four', 'stories_one', 'stories_three', 'stories_two'],
      dtype='object')
after delete Index(['price', 'lotsize', 'bedrooms', 'bathrms', 'driveway', 'recroom',
       'fullbase', 'gashw', 'airco', 'garagepl', 'prefarea', 'stories_four',
       'stories_one', 'stories_three', 'stories_two'],
      dtype='object')
```

```
# check rhe correlation
data.corr()
```

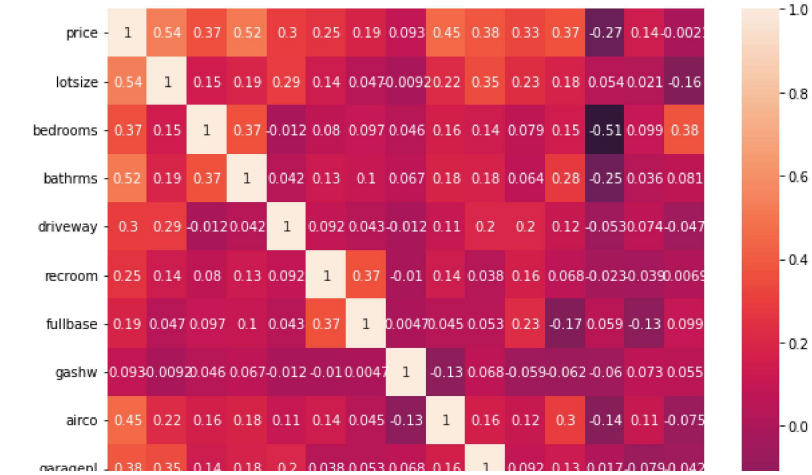
	price	lotsize	bedrooms	bathrms	driveway	recroom	fullbase
price	1.000000	0.535796	0.366447	0.516719	0.297167	0.254960	0.186218
lotsize	0.535796	1.000000	0.151851	0.193833	0.288778	0.140327	0.047487
bedrooms	0.366447	0.151851	1.000000	0.373769	-0.011996	0.080492	0.097201
bathrms	0.516719	0.193833	0.373769	1.000000	0.041955	0.126892	0.102791
driveway	0.297167	0.288778	-0.011996	0.041955	1.000000	0.091959	0.043428
recroom	0.254960	0.140327	0.080492	0.126892	0.091959	1.000000	0.372434
fullbase	0.186218	0.047487	0.097201	0.102791	0.043428	0.372434	1.000000
gashw	0.092837	-0.009201	0.046028	0.067365	-0.011942	-0.010119	0.004677
airco	0.453347	0.221765	0.160412	0.184955	0.106290	0.136626	0.045248
garagepl	0.383302	0.352872	0.139117	0.178178	0.203682	0.038122	0.052524
prefarea	0.329074	0.234782	0.078953	0.064013	0.199378	0.161292	0.228651

```
# create a correlation maxtrix(color Grid)
import statsmodels.api as sm
sm.graphics.plot_corr(data.corr(),xnames=data.columns,title="HouseMatrix")
```



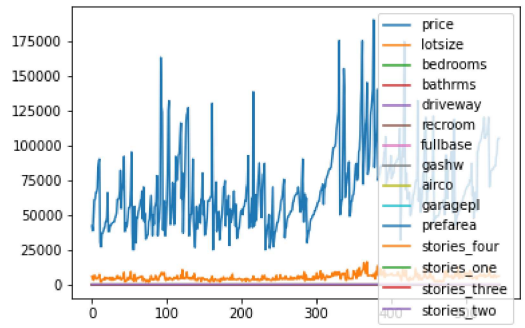

```
# create a correlation maxtrix using seaborn
import seaborn as sb
import matplotlib.pyplot as plt
fig, ax =plt.subplots(figsize=(10,10))
sb.heatmap(data.corr(),annot =True,ax =ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f86cabaf040>
```




```
data.plot()
print("Price ranges from", data.price.min(), "-", data.price.max())
print("Bedroom ranges from", data.bedrooms.min(), "-", data.bedrooms.max())
```

```
Price ranges from 25000.0 - 190000.0
Bedroom ranges from 1 - 6
```




```
# Transformation using Standardization
# Formula = (X - Xmean) / Xstd

X = data["price"]
Xmean = X.mean()
print("Mean of price is", Xmean)
Xstd = X.std()
print('Standard deviation is', Xstd)

Xnorm = (X - Xmean) / Xstd
print("Normalized price is", Xnorm)
```

```
Mean of price is 68121.59706959708
Standard deviation is 26702.670925794933
Normalized price is 0     -0.978239
1     -1.109312
2     -0.697368
3     -0.285425
4     -0.266700
         ...
541    0.875508
542    0.969132
543    1.306177
544    1.381075
545    1.381075
Name: price, Length: 546, dtype: float64
```

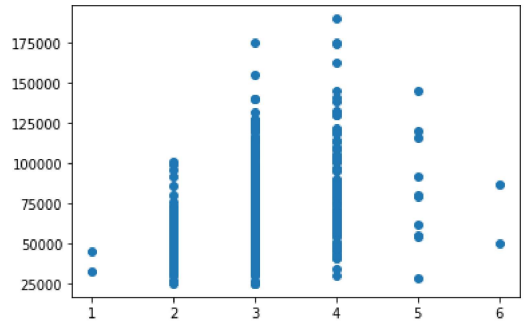
```
# Standardization of data using Standard Score
standard_data = (data - data.mean())/data.std()

# Plot standardized values
standard_data.plot(figsize=(300, 2))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f86dba5bcd0>
```

```
plt.scatter(data["bedrooms"],data["price"])
```

```
<matplotlib.collections.PathCollection at 0x7f86c8c99ca0>
```




```
ss = pp.StandardScaler()
ss.fit_transform(data)
```

```
array([[-0.97913617,  0.32302806,  0.0472349 , ..., -0.84356313,
        -0.28116078,  1.13759292],
       [-1.11032939, -0.53101296, -1.31014696, ...,  1.18544774,
        -0.28116078, -0.87904907],
       [-0.69800783, -0.96495812,  0.0472349 , ...,  1.18544774,
        -0.28116078, -0.87904907],
       ...,
       [ 1.30737434,  0.39227462,  0.0472349 , ..., -0.84356313,
        -0.28116078, -0.87904907],
       [ 1.3823419 ,  0.39227462,  0.0472349 , ..., -0.84356313,
        -0.28116078,  1.13759292],
       [ 1.3823419 ,  0.39227462,  0.0472349 , ..., -0.84356313,
        -0.28116078,  1.13759292]])
```

```
# Transfrom using min - max scaler
# xnorm = (x-xmin)/ (xmax-xmin)
norm_data  = (data - data.min()) / (data.max()-data.min())
norm_data
```

	price	lotsize	bedrooms	bathrms	driveway	recroom	fullbase	gashw	air
0	0.103030	0.288660	0.4	0.000000	1.0	0.0	1.0	0.0	
1	0.081818	0.161512	0.2	0.000000	1.0	0.0	0.0	0.0	
2	0.148485	0.096907	0.4	0.000000	1.0	0.0	0.0	0.0	
3	0.215152	0.343643	0.4	0.000000	1.0	1.0	0.0	0.0	
4	0.218182	0.323711	0.2	0.000000	1.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
541	0.403030	0.216495	0.4	0.333333	1.0	1.0	0.0	0.0	
542	0.418182	0.298969	0.4	0.333333	1.0	0.0	0.0	0.0	
543	0.472727	0.298969	0.4	0.333333	1.0	1.0	0.0	0.0	
544	0.484848	0.298969	0.4	0.333333	1.0	1.0	0.0	0.0	
545	0.484848	0.298969	0.4	0.000000	1.0	0.0	0.0	0.0	

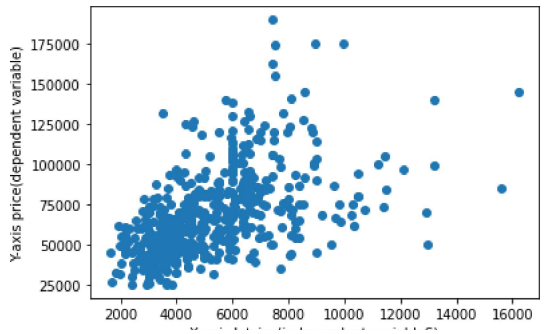
546 rows × 15 columns

```
# print min max ranges from price and bedrooms
print("price ranges", norm_data["price"].min(),norm_data["price"].max())
print("bedrooms ranges", norm_data["bedrooms"].min(),norm_data["bedrooms"].max())
```

```
price ranges 0.0 1.0
bedrooms ranges 0.0 1.0
```

```
plt.scatter(data["lotsize"],data["price"])
plt.xlabel("X-axis lotsize(independent variableS)")
plt.ylabel("Y-axis price(dependent variable)")
```

```
Text(0, 0.5, 'Y-axis price(dependent variable)')
```




```
Y = data["price"]
Y # dependent variable
X = data #independent variables
del X["price"]


regression = sm.OLS(Y, X) # refression equation /model

# train the model
# calculate the value of attribute using actual values of
# dependent  and independent variables
# for regrrssion-caclulate the value of coefficient and
# intercept using the actual values of X and Y
model = regression.fit() # train the model


model.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.674
Model:	OLS	**Adj. R-squared:**	0.666
Method:	Least Squares	**F-statistic:**	84.47
Date:	Mon, 16 Jan 2023	**Prob (F-statistic):**	4.12e-120
Time:	11:27:49	**Log-Likelihood:**	-6033.7
No. Observations:	546	**AIC:**	1.210e+04
Df Residuals:	532	**BIC:**	1.216e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
lotsize	3.4853	0.357	9.760	0.000	2.784	4.187
bedrooms	2207.4007	1126.826	1.959	0.051	-6.174	4420.976
bathrms	1.423e+04	1501.004	9.479	0.000	1.13e+04	1.72e+04
driveway	6744.5906	2049.077	3.292	0.001	2719.317	1.08e+04
recroom	4452.7280	1905.476	2.337	0.020	709.547	8195.909
fullbase	5611.2079	1602.026	3.503	0.000	2464.134	8758.281
gashw	1.298e+04	3244.074	4.002	0.000	6610.754	1.94e+04
airco	1.246e+04	1568.474	7.944	0.000	9379.277	1.55e+04
garagepl	4207.8472	847.752	4.964	0.000	2542.495	5873.200
prefarea	9339.4245	1695.216	5.509	0.000	6009.286	1.27e+04
stories_four	2.265e+04	5140.674	4.406	0.000	1.25e+04	3.27e+04
stories_one	2381.0771	3517.544	0.677	0.499	-4528.903	9291.057
stories_three	1.512e+04	4848.135	3.119	0.002	5596.507	2.46e+04
stories_two	7666.2541	4153.290	1.846	0.065	-492.606	1.58e+04

Omnibus:	98.599	**Durbin-Watson:**	1.603
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	266.624
Skew:	0.891	**Prob(JB):**	1.27e-58
Kurtosis:	5.923	**Cond. No.**	7.07e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.07e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

```
data["predictedprice"]=model.predict()


data
```

	lotsize	bedrooms	bathrms	driveway	recroom	fullbase	gashw	airco	garage
0	5850	3	1	1	0	1	0	0	
1	4000	2	1	1	0	0	0	0	
2	3060	3	1	1	0	0	0	0	
3	6650	3	1	1	1	0	0	0	
4	6360	2	1	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	
541	4800	3	2	1	1	0	0	1	
542	6000	3	2	1	0	0	0	1	
543	6000	3	2	1	1	0	0	1	
544	6000	3	2	1	1	0	0	1	
545	6000	3	1	1	0	0	0	1	

546 rows × 15 columns

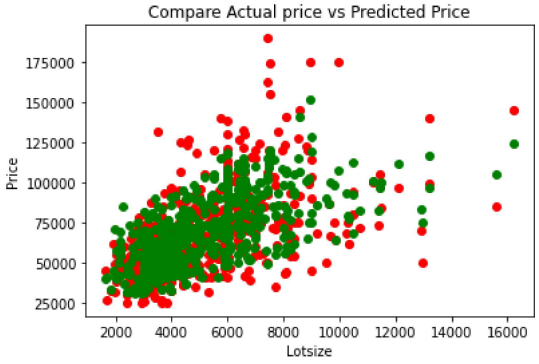
```
data["price"] = Y
data[["price", "lotsize", "predictedprice"]]
```

	price	lotsize	predictedprice
0	42000.0	5850	65469.387145
1	38500.0	4000	41709.888312
2	49500.0	3060	40641.075980
3	60500.0	6650	62891.326583
4	61000.0	6360	49935.274323
...	...	...	...
541	91500.0	4800	98113.739019
542	94000.0	6000	97843.410668
543	103000.0	6000	106503.985852
544	105000.0	6000	91522.232970
545	105000.0	6000	72841.418102

546 rows × 3 columns

```
# Comparing Actual prices vs predicted prices
plt.scatter(data["lotsize"], data["price"], color="red")
plt.scatter(data["lotsize"], data["predictedprice"], color="green")
plt.title("Compare Actual price vs Predicted Price")
plt.xlabel("Lotsize")
plt.ylabel("Price")
```

```
Text(0, 0.5, 'Price')
```



```
del X["predictedprice"]
del X["price"]


# Create an AI Console App
user_data = {}
for column in X.columns:
  temp_val = int(input("Enter "+column+": "))
  user_data[column] = temp_val
user_data
input_data = pd.DataFrame(data = user_data, index=[0])
output = model.predict(input_data)
print("Price of the house is USD", output)
```

```
Enter lotsize: 2000
Enter bedrooms: 3
Enter bathrms: 4
Enter driveway: 1
Enter recroom: 0
Enter fullbase: 0
Enter gashw: 0
Enter airco: 1
Enter garagepl: 0
Enter prefarea: 0
Enter stories_four: 1
Enter stories_one: 0
Enter stories_three: 0
Enter stories_two: 0
Price of the house is USD 0    112358.252144
dtype: float64
```

✓ 1m 22s completed at 4:59 PM