ML LAB 4

Explore and implement Linear Regression Using Gradient Descent in a given business scenario and comment on its efficiency and performance.

In [ ]:
```
# Making the imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 9.0)
from sklearn.linear_model import LinearRegression # To work on Linear Regression
from sklearn.metrics import r2_score # To Calculate Performance matrix
import statsmodels.api as sm # To calculatestats modles
```

```
/usr/local/lib/python3.7/dist-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
```

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kc_house_data.csv to kc_house_data.csv
```

In [ ]:
```
import io
df = pd.read_csv(io.BytesIO(uploaded['kc_house_data.csv']))
```

In [ ]:
```
df.head(20)
```

Out[6]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	
5	7237550310	20140512T000000	1225000.0	4	4.50	5420	101930	1.0	
6	1321400060	20140627T000000	257500.0	3	2.25	1715	6819	2.0	
7	2008000270	20150115T000000	291850.0	3	1.50	1060	9711	1.0	
8	2414600126	20150415T000000	229500.0	3	1.00	1780	7470	1.0	
9	3793500160	20150312T000000	323000.0	3	2.50	1890	6560	2.0	
10	1736800520	20150403T000000	662500.0	3	2.50	3560	9796	1.0	
11	9212900260	20140527T000000	468000.0	2	1.00	1160	6000	1.0	
12	114101516	20140528T000000	310000.0	3	1.00	1430	19901	1.5	
13	6054650070	20141007T000000	400000.0	3	1.75	1370	9680	1.0	
14	1175000570	20150312T000000	530000.0	5	2.00	1810	4850	1.5	
15	9297300055	20150124T000000	650000.0	4	3.00	2950	5000	2.0	
16	1875500060	20140731T000000	395000.0	3	2.00	1890	14040	2.0	
17	6865200140	20140529T000000	485000.0	4	1.00	1600	4300	1.5	
18	16000397	20141205T000000	189000.0	2	1.00	1200	9850	1.0	
19	7983200060	20150424T000000	230000.0	3	1.00	1250	9774	1.0	

In [ ]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
None
```

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

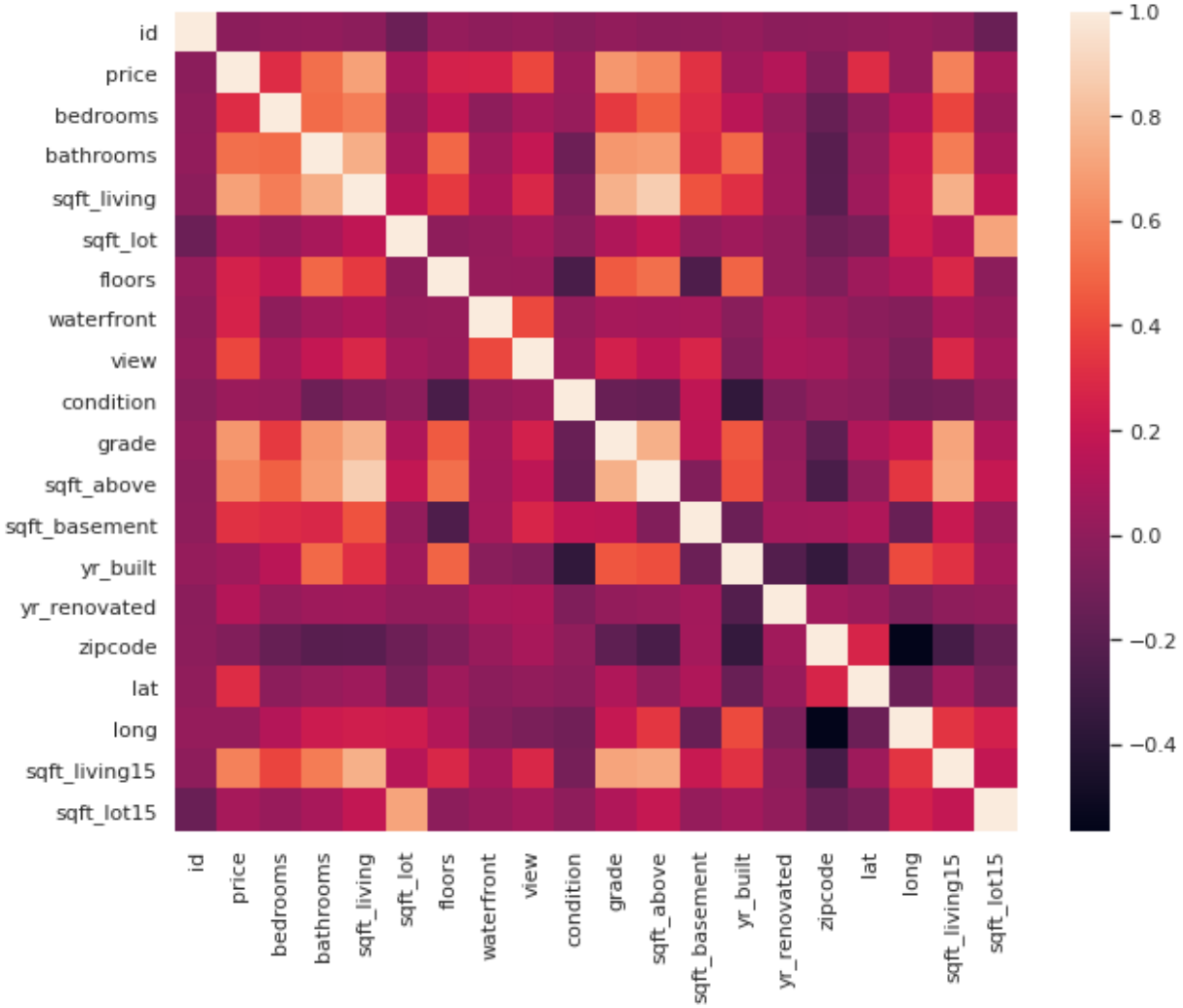
Out[8]:
```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

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

Out[9]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3

In [ ]:
```
import seaborn as sns
sns.set(rc={'figure.figsize':(10,8)})
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
plt.show()
```



In [ ]:
```
data = [df["sqft_living"], df["price"]]
headers = ["sqft_living", "price"]
df1 = pd. concat(data, axis=1, keys=headers)
```

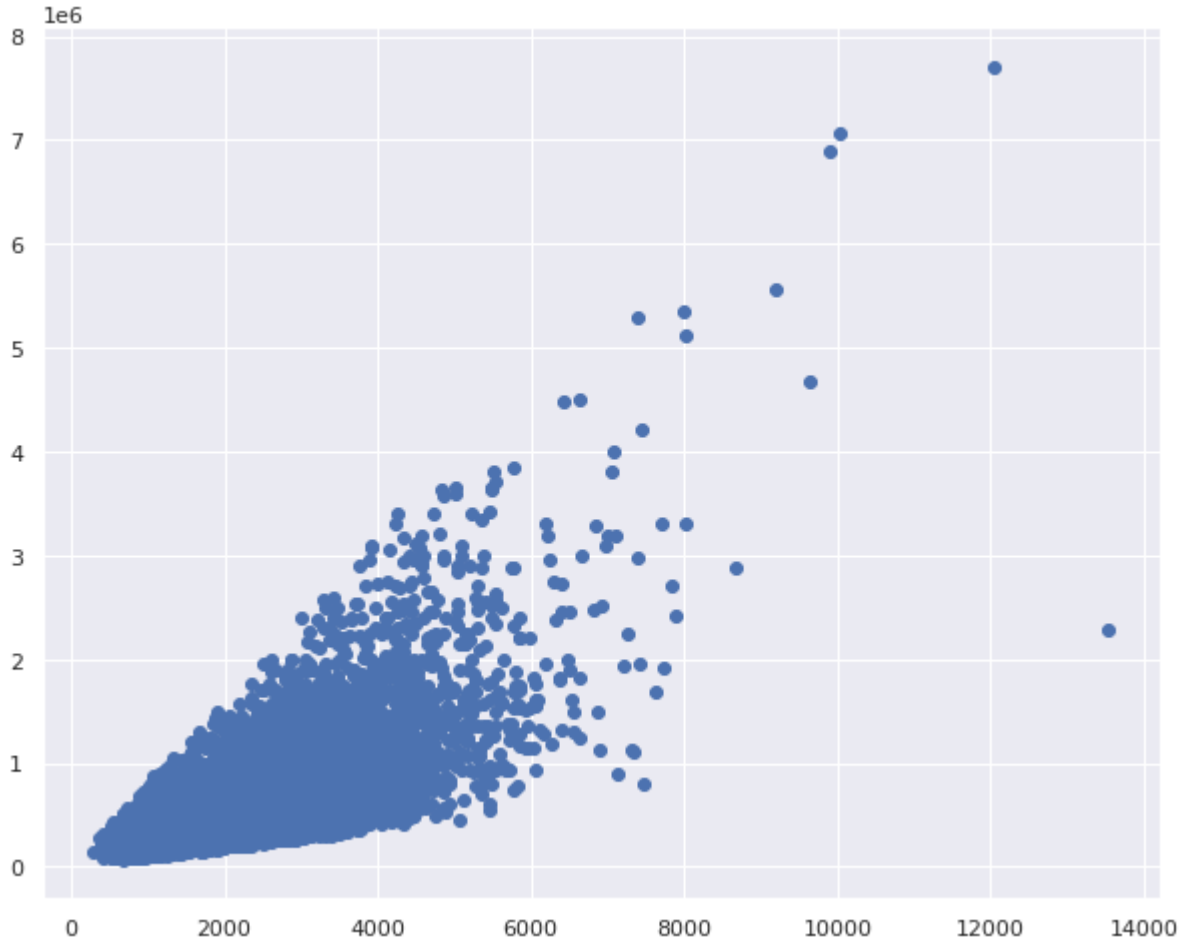
In [ ]:
```
df1
```

Out[24]:

	sqft_living	price
0	1180	221900.0
1	2570	538000.0
2	770	180000.0
3	1960	604000.0
4	1680	510000.0
...	...	...
21608	1530	360000.0
21609	2310	400000.0
21610	1020	402101.0
21611	1600	400000.0
21612	1020	325000.0

21613 rows × 2 columns

In [ ]:
```
plt.scatter("sqft_living","price",data=df1)
plt.show()
```



In [ ]:
```
from sklearn.model_selection import train_test_split, KFold, cross_val_score
training, testing =train_test_split(df1, test_size= 0.30, random_state=24)
```

In [ ]:
```
training
```

Out[27]:

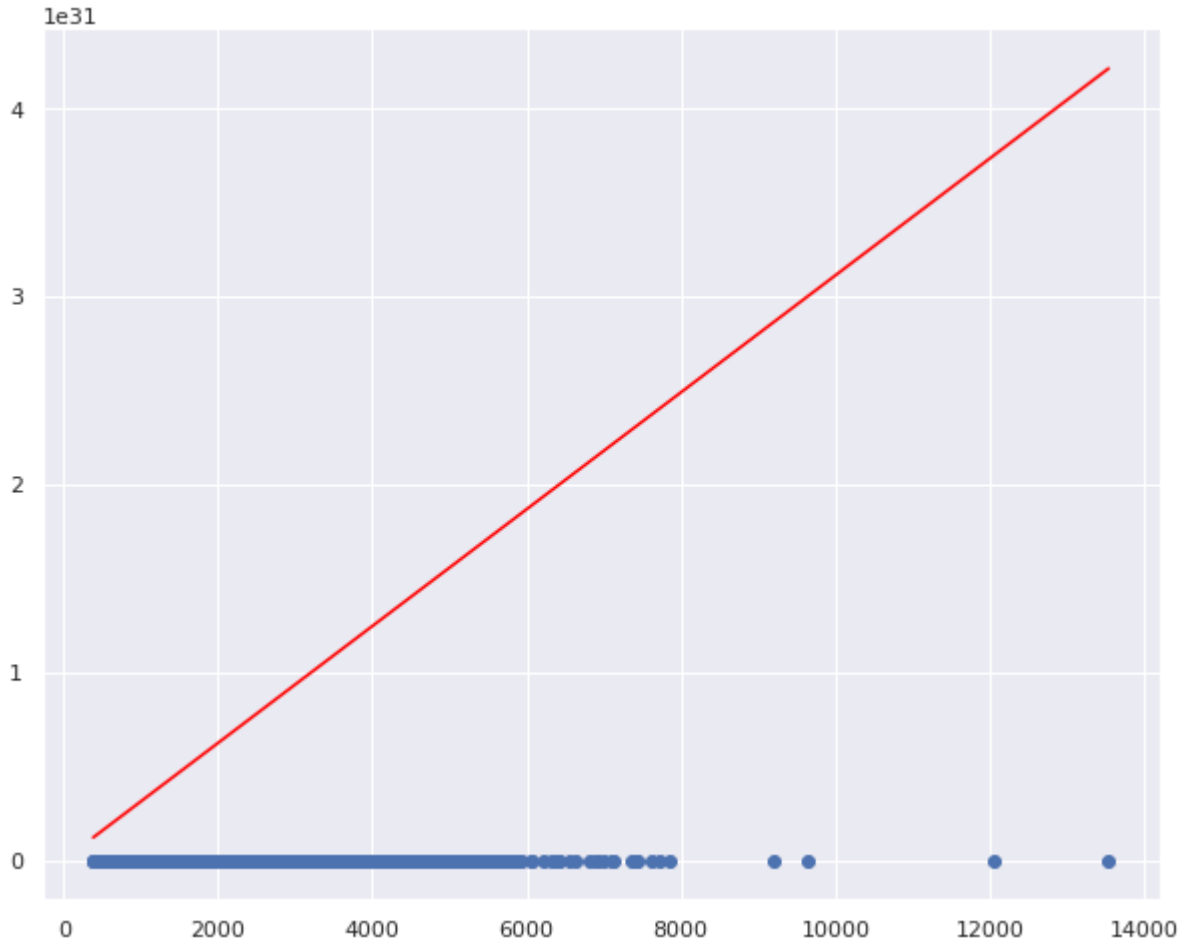
	sqft_living	price
17719	2030	572500.0
10646	3670	883000.0
1949	1008	480000.0
20322	4410	1240000.0
2072	1200	225000.0
...	...	...
6500	3450	755000.0
19857	3100	435000.0
14528	2300	294000.0
899	1260	291500.0
12706	2460	835000.0

15129 rows × 2 columns

In [ ]:
```
# Building the model
m = 0
c = 0

L = 0.01  # The learning Rate
epochs = 5  # The number of iterations to perform gradient descent

n = float(len(df1['sqft_living'])) # Number of elements in X

# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*(df1['sqft_living']) + c  # The current predicted value of Y
    D_m = (-2/n) * sum(df1['sqft_living'] * (df1['price'] - Y_pred))  # Derivativ
    D_c = (-2/n) * sum(df1['price'] - Y_pred)  # Derivative wrt c
    m = m - L * D_m  # Update m
    c = c - L * D_c  # Update c

print (m, c)
```

```
3.107889241700975e+27 1.2504354911183835e+24
```

In [ ]:
```
# Making predictions
Y_pred = m*(testing['sqft_living']) + c

plt.scatter(testing['sqft_living'],testing['price'])
plt.plot([min(testing['sqft_living']), max(testing['sqft_living'])], [min(Y_pred)
plt.show()
```



In [ ]:
```
X = testing['sqft_living'] ## Assign TV ad value to X
y = testing['price'] ## assign sales values to y

X2 = sm.add_constant(X)# Assign stat model  constant to X2
est = sm.OLS(y, X2) # Build Ordinary least square
est2 = est.fit() #Fitting OLS Regression
print(est2.summary()) # Printing OLS Results
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.507
Model:                            OLS   Adj. R-squared:                  0.507
Method:                 Least Squares   F-statistic:                     6656.
Date:                Sun, 29 Aug 2021   Prob (F-statistic):               0.00
Time:                        08:54:01   Log-Likelihood:                -90024.
No. Observations:                6484   AIC:                         1.801e+05
Df Residuals:                    6482   BIC:                         1.801e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -4.348e+04   7855.020     -5.535      0.000   -5.89e+04   -2.81e+04
sqft_living   279.8463      3.430     81.584      0.000     273.122     286.571
==============================================================================
Omnibus:                     4448.786   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           182253.906
Skew:                           2.762   Prob(JB):                         0.00
Kurtosis:                      28.379   Cond. No.                     5.59e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 5.59e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Interpretation:

House prices were predicted with Linear Regression using Gradient descent, along with scatterplot.

In [ ]:
```
```