Linear Regression

What is Regression and Why is it called so?

Regression is an ML algorithm that can be trained to predict real numbered outputs; like temperature, stock price, etc. Regression is based on a hypothesis that can be linear, quadratic, polynomial, non-linear, etc. The hypothesis is a function based on some hidden parameters and input values. In the training phase, the hidden parameters are optimized w.r.t. the input values presented in the training. The process that does the optimization is the gradient descent algorithm. You also need a Back-propagation algorithm that can be used to compute the gradient at each layer, If you are using neural networks. Once the hypothesis parameters got trained (when they gave the least error during the training), then the same hypothesis with the trained parameters is used with new input values to predict outcomes that will be again real values.

Advantages/Features of Linear Regression Linear regression is an extremely simple method. It is very easy and intuitive to use and understand. A person with only the knowledge of high school mathematics can understand and use it. In addition, it works in most cases. Even when it doesn't fit the data exactly, we can use it to find the nature of the relationship between the two variables. Disadvantages/Shortcomings of Linear Regression By its definition, linear regression only models relationships between dependent and independent variables that are linear. It assumes there is a straight-line relationship between them which is incorrect sometimes. Linear regression is very sensitive to the anomalies in the data (or outliers). Take for example most of your data lies in the range 0-10. If due to any reason only one of the data items comes out of the range, say for example 15, this significantly influences the regression coefficients. Another disadvantage is that if we have a number of parameters than the number of samples available then the model starts to model the noise rather than the relationship between the variables.

Types of Regression

1. **Linear regression is used for predictive analysis. Linear regression is a linear approach for modeling the relationship between the criterion or the scalar response and the multiple predictors or explanatory variables. Linear regression focuses on the conditional probability distribution of the response given the values of the predictors. For linear regression, there is a danger of overfitting. The formula for linear regression is Y' = bX + A.**
2. **Logistic regression is used when the dependent variable is dichotomous. Logistic regression estimates the parameters of a logistic model and is a form of binomial regression. Logistic regression is used to deal with data that has two possible criteria and the relationship between the criteria and the predictors. The equation for logistic regression is l = \beta*{0}+\beta*{1}x*{1}+\beta*{2}x_{2}.**

   #There are many type of Linear Regression

Key Terms

1. **Estimator A formula or algorithm for generating estimates of parameters, given relevant data.**
2. **Bias An estimate is unbiased if its expectation equals the value of the parameter being estimated; otherwise, it is biased.**

TOPIC

> **Linear Regression Algorithm**
>
> **Applications of Linear Regression**
>
> **Use case of Linear Regression**

Linear Regression Algorithm

Linear Regression is a machine learning algorithm based on supervised learning. ... Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output)

Introduction

Artificial Intelligence has become prevalent to resolve complex problems in our world throughout different sectors.

Developers, data scientists and researchers across different disciplines use this technology to make their lives easier. For example, doctors use AI to classify whether a tumor is malignant or benign, commercials are using AI to predict customer features, and meteorologists use AI to predict the weather.

The impetus behind such ubiquitous use of AI is machine learning prediction. This is applicable to every real complex problem which we cannot solve manually.

In this post, I write about the linear regression problem, used to predict a real-valued output based on independent predictors.

It is a very simple approach for supervised learning. This method is mostly used for forecasting and finding out cause and effect relationship between variables (X,Y)

Mathematically, we can write a linear relationship as,

Y= beta0+ beta1 x1 + beta2 x2 + … + betan xn

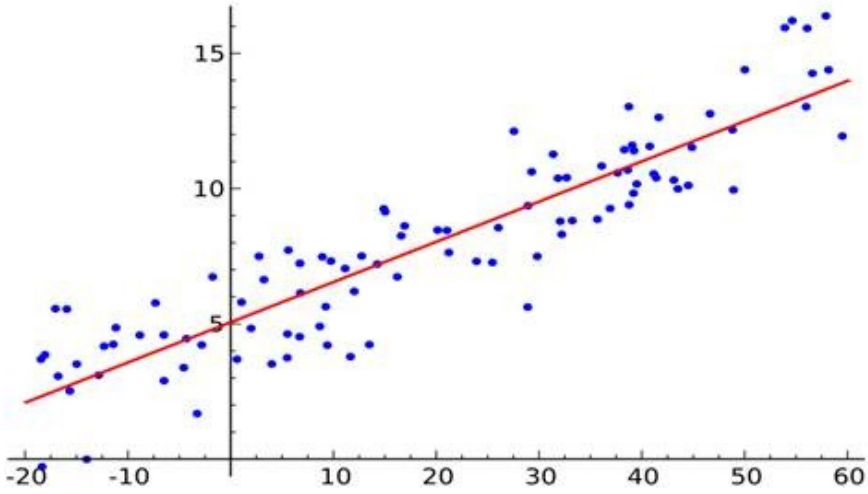
Y : The predictive result

Beta : The Model coefficients (is learned in the training step)

Beta0 : The intercept

Beta1 : first feature

Betan : the nth feature



Based on the given data points we draw the graph. Then we draw the line that regroups the max of point in the above graph is referred to as the best fit straight line.

The raining process of the model. it's way when we can find out coefficients for the linear function (beta)

The cost function is the fact of minimizing the error.

To estimate the coefficients, we use gradient Descent, Start with some values of the coefficients/parameters(2x+3y >>> 2 and 3 are coefficients), beta0=0, beta1=0, keep changing beta0 and beta1 until we obtain the best result

until we obtain the best result.

Simple Linear Regression Model

The simple linear regression model is represented like this: y = (β0 +β1 + E)

By mathematical convention, the two factors that are involved in simple linear regression analysis are designated x and y. The equation that describes how y is related to x is known as the regression model. The linear regression model also contains an error term that is represented by E, or the Greek letter epsilon. The error term is used to account for the variability in y that cannot be explained by the linear relationship between x and y. There also parameters that represent the population being studied. These parameters of the model are represented by (β0+β1x).

The simple linear regression equation is graphed as a straight line.

The simple linear regression equation is represented like this: E(y) = (β0 +β1*x) β0 is the y-intercept of the regression line. β1 is the slope. E(y) is the mean or expected value of y for a given value of x.

A regression line can show a positive linear relationship, a negative linear relationship, or no relationship. If the graphed line in a simple linear regression is flat (not sloped), there is no relationship between the two variables. If the regression line slopes upward with the lower end of the line at the y-intercept (axis) of the graph, and the upper end of the line extending upward into the graph field, away from the x-intercept (axis) a positive linear relationship exists. If the regression line slopes downward with the upper end of the line at the y-intercept (axis) of the graph, and the lower end of the line extending downward into the graph field, toward the x-intercept (axis) a negative linear relationship exists.

Applications of Linear Regression

Linear regressions can be used in business to evaluate trends and make estimates or forecasts. For example, if a company's sales have increased steadily every month for the past few years, by conducting a linear analysis on the sales data with monthly sales, the company could forecast sales in future months

Use case of Linear Regression

Linear regressions can be used in business to evaluate trends and make estimates or forecasts. ... Linear regression can also be used to analyze the marketing effectiveness, pricing and promotions on sales of a product.

Import Lib

In [26]:

```
%matplotlib inline
import pandas as pd
import numpy as np
from pandas import DataFrame
import matplotlib.pyplot as plt
import seaborn as sns
```

Import the Data set

Using House Data set

In [27]:

```
data = pd.read_csv("/content/kc_house_data.csv")
```

In [28]:

```
data
```

Out[28]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	3
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	3
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	3
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	5
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	3
...	...	...	...	...	...	...	...	...	...	...	...
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	3
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	3
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	3
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	3
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	3

21613 rows × 21 columns

Find Shape

In [29]:

```
data.shape
```

Out[29]:

```
(21613, 21)
```

Describe function is magic command, calculate all mean standar div. percetage etc

In [35]:

```
df=data.describe()
df
```

Out[35]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	216
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000

[1:100] 1 to 100 data['price']

yaxis=2.00 is 2lac

xaxis=index no range

x and y find the middle avrage of regression

Find the regression on both

first is price

scond is sqft_living >> squree feet

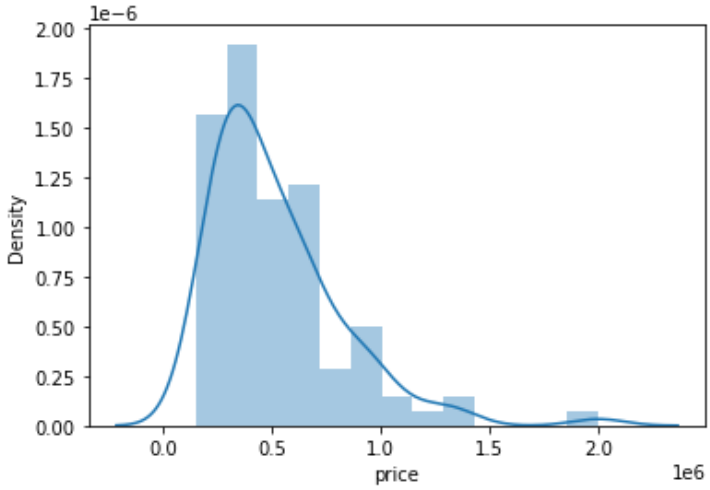
In [38]:

```
sns.distplot(data['price'][1:100])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f026388a610>
```



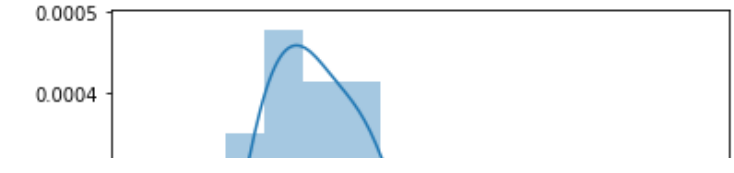
In [40]:

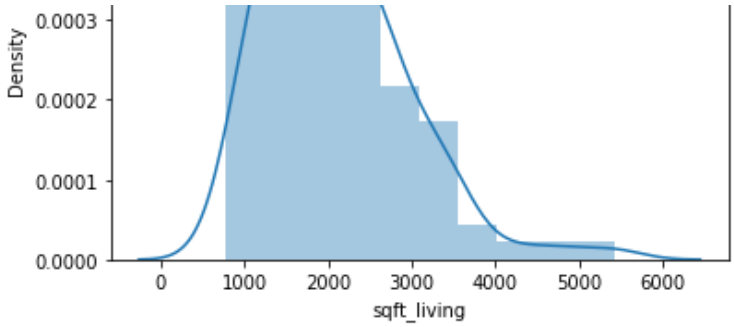
```
sns.distplot(data['sqft_living'][1:100])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0263990450>
```





Regression Formulas

> **find the mean x and y**
>
> **calculating cross-deviation and deviation about x**

```
SS_xy = np.sum(y*x) - n*mean y*mean x
SS_xx = np.sum(x*x) - n*mean x*mean y
```

> **calculating regression coefficients**

```
b_1 = SS_xy / SS_xx
b_0 = m_y - b_1*m_x
```

$$b_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$b_0 = \bar{y} - b_1\bar{x}$$

$$SS_{xy} = \sum x_i y_i - \frac{\left(\sum x_i\right)\left(\sum y_i\right)}{n}$$

$$SS_{xx} = \sum x_i^2 - \frac{\left(\sum x_i\right)^2}{n} = (n-1)s_x^2$$

Analysis

In [46]:

```
x= data['sqft_living'][1:100]
y=data['price'][1:100]
```

number of observations/points

In [47]:

```
nx = np.size(x)
```

In [48]:

```
nx
```

Out[48]:

```
99
```

In [49]:

```
ny=np.size(y)
```

In [50]:

```
ny
```

Out[50]:

```
99
```

mean of x and y vector

In [51]:

```
x_mean=np.mean(x)
```

In [52]:

```
x_mean
```

Out[52]:

```
2097.89898989899
```

In [53]:

```
y_mean=np.mean(y)
```

In [54]:

```
y_mean
```

Out[54]:

```
522151.8686868687
```

calculating cross-deviation and deviation about x

The cross-product of deviations is equal to the sum of the products of mean-corrected variables. This is the numerator of the Pearson correlation coefficient. The covariance is an unstandardized measure of the relationship between two variables

In [61]:

```
ss_xy = np.sum(y*x) - nx*y_mean*x_mean      #nx is total number
```

In [62]:

```
ss_xy
```

Out[62]:

```
16604634588.686874
```

In [63]:

```
ss_xx = np.sum(x*x) - nx*x_mean*x_mean
```

In [64]:

```
ss_xx
```

Out[64]:

```
72476882.98989898
```

calculating regression coefficients

In [67]:

```
b1 = ss_xy / ss_xx
```

In [68]:

```
b1
```

Out[68]:

```
229.10249314945074
```

In [69]:

```
b0 = y_mean - b1*x_mean
```

In [70]:

```
b0
```

Out[70]:

```
41517.979725295736
```

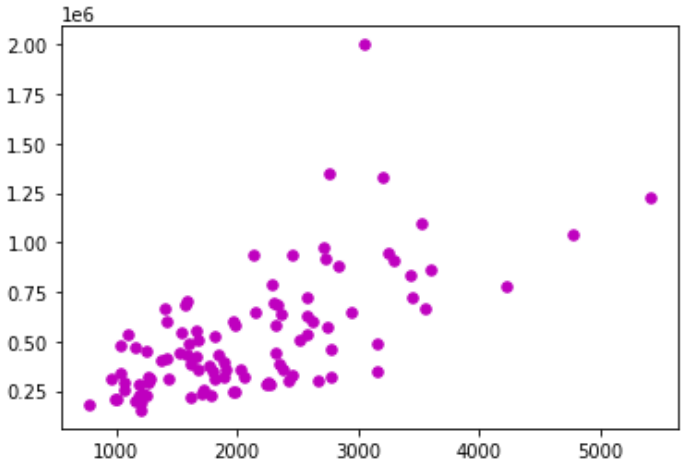
Know find the plotting the actual points as scatter plot

In [72]:

```
plt.scatter(x, y, color = "m",marker = "o", s = 30)  #this x and y plot
```

Out[72]:

```
<matplotlib.collections.PathCollection at 0x7f0263446110>
```



The simple linear regression equation is graphed as a straight line.

The simple linear regression equation is represented like this: E(y) = (β0 +β1*x) β0 is the y-intercept of the regression line. β1 is the slope.

find the prediction x axis in future

In [102]:

```
pre_x = b0 + b1*y
```

In [103]:

```
pre_x
```

Out[103]:

```
1     1.232987e+08
2     4.127997e+07
3     1.384194e+08
4     1.168838e+08
5     2.806921e+08
          ...
95    2.073793e+08
96    5.674439e+07
97    4.563291e+07
98    7.197970e+07
99    1.002739e+08
Name: price, Length: 99, dtype: float64
```

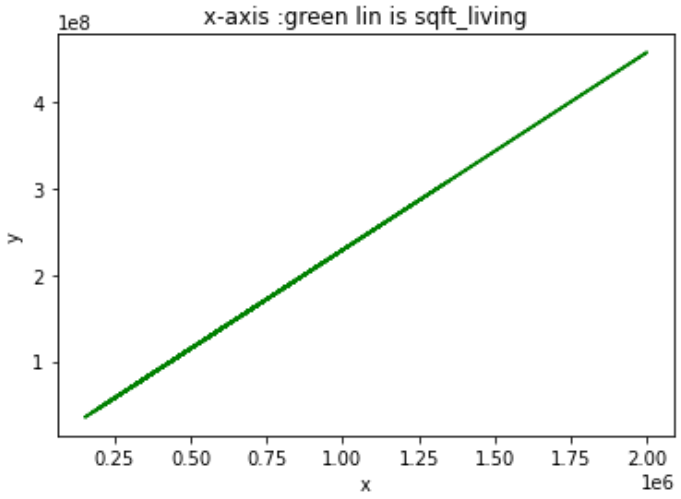
plotting the regression line x prediction

In [104]:

```
plt.plot(y, pre_x, color = "g")
plt.title('x-axis :green lin is sqft_living')
plt.xlabel('x')
plt.ylabel('y')
```

Out[104]:

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



in future squre feet is progress

find the prediction y axis in future

In [105]:

```
pre_y = b0 + b1*x
```

In [106]:

```
pre_y
```

Out[106]:

```
1     6.303114e+05
```

```
2     2.179269e+05
3     4.905589e+05
4     4.264102e+05
5     1.283253e+06
          ...
95    7.975562e+05
96    4.905589e+05
97    3.072769e+05
98    4.561935e+05
99    5.730358e+05
Name: sqft_living, Length: 99, dtype: float64
```

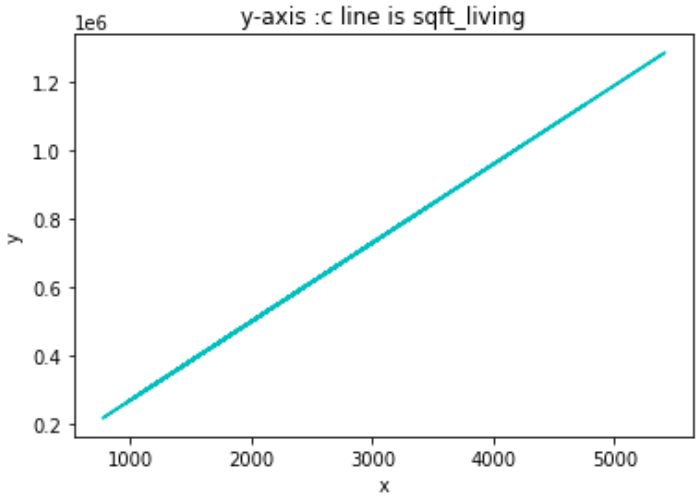
plotting the regression line x prediction

In [107]:

```
plt.plot(x,pre_y,color='c')
plt.title('y-axis :c line is sqft_living')
plt.xlabel('x')
plt.ylabel('y')
```

Out[107]:

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



In future(prediction) price grether

estimating coefficients

In [111]:

```
b = estimate_coef(x,y)
```

```
c 41517.979725295736 229.10249314945074
```

In [113]:

```
print('estimate_sqft_living::41517.979725295736\n','estimate_price::229.10249314945074')
```

```
estimate_sqft_living::41517.979725295736
 estimate_price::229.10249314945074
```

In [ ]:

```

```