

DR. ALVIN'S PUBLICATIONS

MULTIPLE REGRESSION USING PYTHON

DR. ALVIN ANG



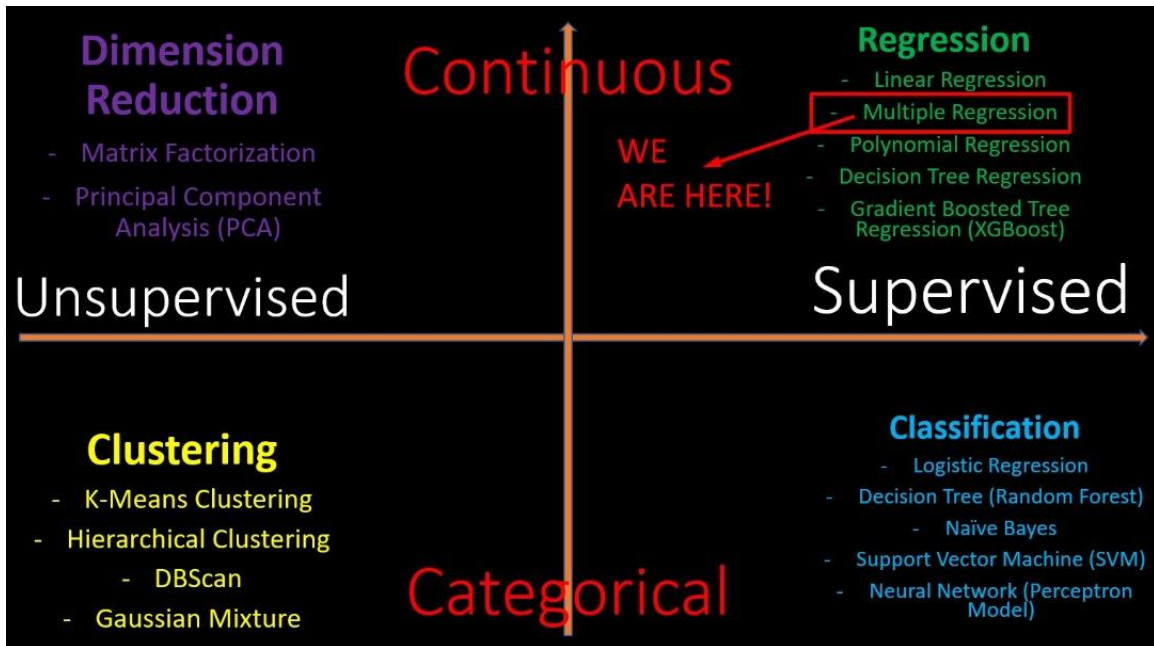
1 | PAGE

COPYRIGHTED BY DR ALVIN ANG
WWW.ALVINANG.SG

CONTENTS

I. Introduction	3
II. Python – using Scikit Learn	4
(Advertising.csv)	4
A. Load and Glance	4
B. Create the Linear Model	4
C. Produce the Model	5
D. Predict the Model	5
E. Predicting the Model Using X values	6
III. Python – Using Scikit Learn	7
(AutomobileEDA.csv)	7
A. Part I: Load and Glance the Dataset	7
B. PART II: Generate a Multiple Linear Regression Equation	8
1. Define Our Z and X	8
2. Fit the Linear Model	8
3. Step 4: Find the Z-Intercept	9
4. Step 5: Find the Gradient	9
C. Part III: Distribution Plot	10
1. Make a Prediction	10
2. Visualize the Distribution Plot.....	11
D. Part IV: Use R2 and MSE as indicators to determine the accuracy of the MR fit	12
1. Calculate the R2 for MR	12
a) Step 1: Fit the MR Model.....	12
b) Step 2: Find the R2.....	13
2. Calculate the MSE for MR	13
a) Step 1: Do a Prediction	13
b) Step 2: Find the MSE.....	13
E. Part V: Simple Linear Regression model (SLR) vs Multiple Linear Regression model (MLR)	14
1. Recall back in Simple Linear Regression (SLR).....	14
2. Now, for Multiple Linear Regression (MLR)	14
3. Comparison... ..	14
About Dr. Alvin Ang	15

I. INTRODUCTION



II. PYTHON – USING SCIKIT LEARN

(ADVERTISING.CSV)

A. LOAD AND GLANCE

- Dataset can be found here: <https://www.alvinang.sg/s/Advertising.csv>
- [https://www.alvinang.sg/s/Multiple Regression using Scikit Learn with Python by Dr Alvin Ang.ipynb](https://www.alvinang.sg/s/Multiple%20Regression%20using%20Scikit%20Learn%20with%20Python%20by%20Dr%20Alvin%20Ang.ipynb)

```
import pandas as pd

# Import and display first five rows of advertising dataset
advert = pd.read_csv('https://www.alvinang.sg/s/Advertising.csv')
advert.head()
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

B. CREATE THE LINEAR MODEL

```
from sklearn.linear_model import LinearRegression

# Build linear regression model using TV and Radio as predictors
# Split data into predictors X and output Y

X = advert[['TV', 'Radio']]
y = advert['Sales']

# Initialise and fit model
lm = LinearRegression()
model = lm.fit(X, y)
```

- $Y \sim \text{Sales}$
- $X \sim \text{TV and Radio(advertising)}$

C. PRODUCE THE MODEL

```
print(f'alpha = {model.intercept_}')
print(f'betas = {model.coef_}')

#Sales = 2.921 + 0.046*TV + 0.1880*Radio

alpha = 2.921099912405138
betas = [0.04575482 0.18799423]
```

D. PREDICT THE MODEL

```
new_X = [[300, 200]]
print(model.predict(new_X))

#If we spend $300 on TV advertising and $200 on Radio advertising,
#We should predict TV = 54 units sold.

[54.24638977]
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid
"X does not have valid feature names, but"
```

E. PREDICTING THE MODEL USING X VALUES

```
model.predict(X)
```

```
array([20.55546463, 12.34536229, 12.33701773, 17.61711596, 13.22390813,
       12.51208449, 11.71821241, 12.10551553,  3.7093792 , 12.55169696,
        7.0358597 , 17.25652015, 10.60866187,  8.81095051, 18.44466773,
       20.82891539, 12.90386507, 23.24107626,  9.94121476, 14.15384619,
       18.12139161, 14.74206357,  6.51417168, 16.54402663,  8.14035215,
       15.6080206 , 14.96769383, 17.0463346 , 19.39954145,  9.15929748,
       21.64292187, 11.35791808,  7.65045928, 18.83346334,  7.56302763,
       16.99280099, 23.36720719, 15.6258994 ,  9.91257829, 20.4405801 ,
       16.37872122, 17.29870935, 21.5621537 , 13.96692266,  8.9009974 ,
       15.16263814,  8.88644967, 21.69944046, 16.28690268,  8.18162949,
       12.64569407,  9.31962792, 20.66180115, 19.96126242, 20.35512357,
       21.30864743,  8.53774783, 12.76239488, 21.89072858, 18.10746914,
        5.74497097, 22.90418658, 16.78413768, 13.18474853, 16.96570907,
        7.82652846,  8.98703456, 12.02066194, 18.95313425, 21.09369037,
       17.78350693, 10.63329605, 10.35113844,  9.91334008, 17.30983543,
       11.90970399,  4.48014809, 13.79239059,  8.78920329,  9.67621401,
       11.43621364, 14.6638809 , 10.18272029, 14.41647235, 20.77350468,
       15.22002396, 11.58203354, 15.61872354, 11.75510286, 16.93110264,
        9.98714329,  4.51167896, 19.17972975, 21.26277229, 10.46708623,
       16.33347878, 12.62023117, 15.32904398, 24.12842563, 16.94651016.]
```

- There are 200 predicted values as shown above.
- It's a prediction of Y given the current X (TV / Radio) values.
- Note / remember that there are 200 rows of data
- In other words, we used these 200 rows of data for TV and Radio to predict the Sales.

	C1	C2	C3	C4	C5
		TV	Radio	Newspaper	Sales
184	184	287.6	43.0	71.8	26.2
185	185	253.8	21.3	30.0	17.6
186	186	205.0	45.1	19.6	22.6
187	187	139.5	2.1	26.6	10.3
188	188	191.1	28.7	18.2	17.3
189	189	286.0	13.9	3.7	15.9
190	190	18.7	12.1	23.4	6.7
191	191	39.5	41.1	5.8	10.8
192	192	75.5	10.8	6.0	9.9
193	193	17.2	4.1	31.6	5.9
194	194	166.8	42.0	3.6	19.6
195	195	149.7	35.6	6.0	17.3
196	196	38.2	3.7	13.8	7.6
197	197	94.2	4.9	8.1	9.7
198	198	177.0	9.3	6.4	12.8
199	199	283.6	42.0	66.2	25.5
200	200	232.1	8.6	8.7	13.4

III. PYTHON – USING SCIKIT LEARN

(AUTOMOBILEEDA.CSV)

- The dataset is here: <https://www.alvinang.sg/s/automobileEDA.csv>
- <https://www.alvinang.sg/s/Multiple Regression using Scikit Learn with Python Part II by Dr Alvin Ang.ipynb>

A. PART I: LOAD AND GLANCE THE DATASET

```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

path = 'https://www.alvinang.sg/s/automobileEDA.csv'
df = pd.read_csv(path)
df.head()
```

- Output:

mpg	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27	13495.0
3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27	16500.0
1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...	9.0	154.0	5000.0	19	26	16500.0
2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	...	10.0	102.0	5500.0	24	30	13950.0
2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	...	8.0	115.0	5500.0	18	22	17450.0

vs x 29 columns

B. PART II: GENERATE A MULTIPLE LINEAR REGRESSION EQUATION

```
✓ 1s ▶ from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm

LinearRegression()
```

1. DEFINE OUR Z AND X

```
✓ 0s ▶ # X1 = "horsepower"
# X2 = "curb-weight"
# X3 = "engine-size"
# X4 = "highway-mpg"
# Z = "price"
# That is  $Z = A + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4$ 

Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

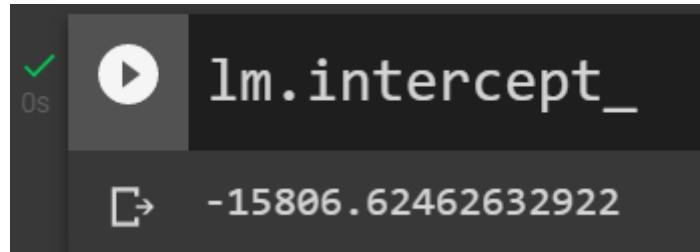
2. FIT THE LINEAR MODEL

```
✓ 0s ▶ lm.fit(Z, df['price'])

LinearRegression()
```


3. STEP 4: FIND THE Z-INTERCEPT

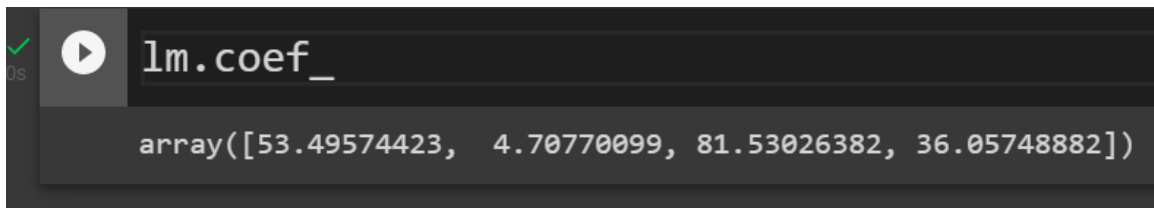
- Z-Intercept refers to the A of the $Z = A + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4$



```
✓ 0s ▶ lm.intercept_  
↳ -15806.62462632922
```

4. STEP 5: FIND THE GRADIENT

- Gradient refers to the b's of the $Z = A + b_1X_1 + b_2X_2 + b_3X_3 + b_4X_4$



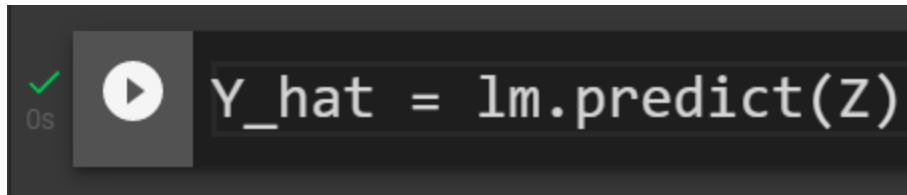
```
✓ 0s ▶ lm.coef_  
array([53.49574423, 4.70770099, 81.53026382, 36.05748882])
```

- This means that the Multiple Linear Equation is
 - Price = -15806
 - + 53 * horsepower
 - + 4.7 * curb-weight
 - + 81.5 * engine-size
 - + 36 * highway-mpg

C. PART III: DISTRIBUTION PLOT

- How do we visualize a model for Multiple Linear Regression?
- This gets a bit more complicated because you can't visualize it with regression or residual plot.
- One way to look at the fit of the model is by looking at the distribution plot:
- We can look at the distribution of the fitted values that result from the model and compare it to the distribution of the actual values.

1. MAKE A PREDICTION

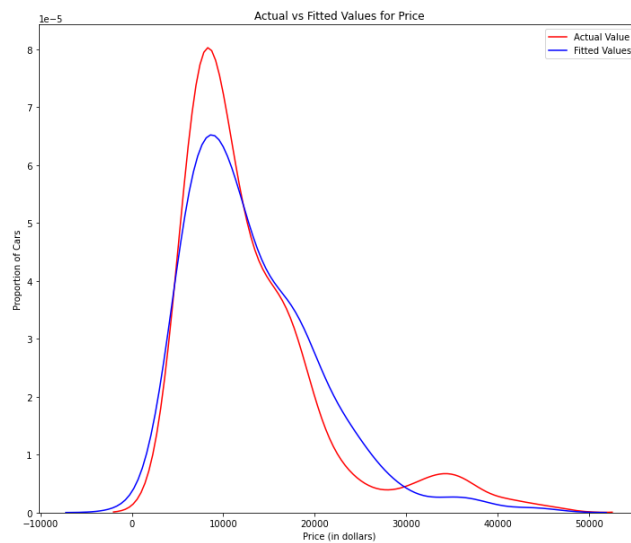


2. VISUALIZE THE DISTRIBUTION PLOT

```
import seaborn as sns
%matplotlib inline
width = 12
height = 10

plt.figure(figsize=(width, height))
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")
sns.distplot(Y_hat, hist=False, color="b", label="Fitted Values", ax=ax1)
plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion of Cars')
plt.show()
plt.close()
```

- Output:



○

- Comments:

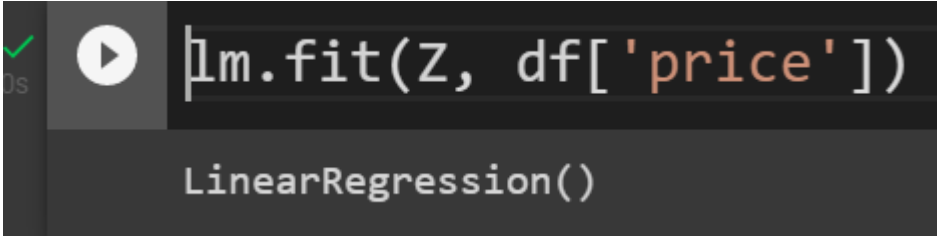
- We can see that the fitted values are reasonably close to the actual values, since the two distributions overlap a bit.
- However, there is definitely some room for improvement.
- MR is quite a good fit.

D. PART IV: USE R2 AND MSE AS INDICATORS TO DETERMINE THE ACCURACY OF THE MR FIT

- R2 has been explained here:
 - <https://www.alvinang.sg/s/How-to-Perform-Simple-Linear-Regression-using-Excel-Dr-Alvin-Ang-watermarked.pdf>
 - R squared, also known as the coefficient of determination, is a measure to indicate how close the data is to the fitted regression line.
- Mean Squared Error (MSE) has been explained here:
 - <https://www.alvinang.sg/s/Forecasting-by-Dr-Alvin-Ang-watermarked-hjr9.pdf>
 - The Mean Squared Error measures the average of the squares of errors, that is, the difference between actual value (y) and the estimated value (\hat{y}).


1. CALCULATE THE R2 FOR MR

a) Step 1: Fit the MR Model



```
lm.fit(Z, df['price'])  
LinearRegression()
```

b) *Step 2: Find the R2*

```
0s ✓  print('The R-square is: ', lm.score(Z, df['price']))
```

The R-square is: 0.8093562806577457


- Comment:
 - We can say that ~ 80.896 % of the variation of price is explained by this multiple linear regression "multi_fit".
 - 80% means that actually a MR model is a good fit...which means that the actual data is quite near the fitted line...

2. CALCULATE THE MSE FOR MR

a) *Step 1: Do a Prediction*

```
0s ✓  Y_predict_multifit = lm.predict(Z)
```

b) *Step 2: Find the MSE*

```
0s ✓  from sklearn.metrics import mean_squared_error  
print('The mean square error of price and predicted value using multifit is: ', \n      mean_squared_error(df['price'], Y_predict_multifit))
```

The mean square error of price and predicted value using multifit is: 11980366.87072649

E. PART V: SIMPLE LINEAR REGRESSION MODEL (SLR) VS MULTIPLE LINEAR REGRESSION MODEL (MLR)

1. RECALL BACK IN SIMPLE LINEAR REGRESSION (SLR)¹...

- We used “highway-mpg” vs “price”.
- The R2 for the SLR was: 0.49659118843391759
- The MSE for the SLR was: 3.16×10^7

2. NOW, FOR MULTIPLE LINEAR REGRESSION (MLR) ...

- In this article, we used Multiple Linear Regression (MLR):
 - Horsepower,
 - Curb-weight,
 - Engine-size, and
 - Highway-mpg vs Price
- The R2 for MLR was: 0.80896354913783497
- The MSE for SLR was: 1.2×10^7

3. COMPARISON...

	SLR	MLR
R2	0.497	0.809 (higher R2)
MSE	3.16×10^7	1.2×10^7 (lower MSE)

- R2 in combination with MSE show that MLR seems like the better model fit in this case, compared to SLR.

¹ <https://www.alvinang.sg/s/Simple-Linear-Regression-using-Python-Dr-Alvin-Ang.pdf>

ABOUT DR. ALVIN ANG



Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.