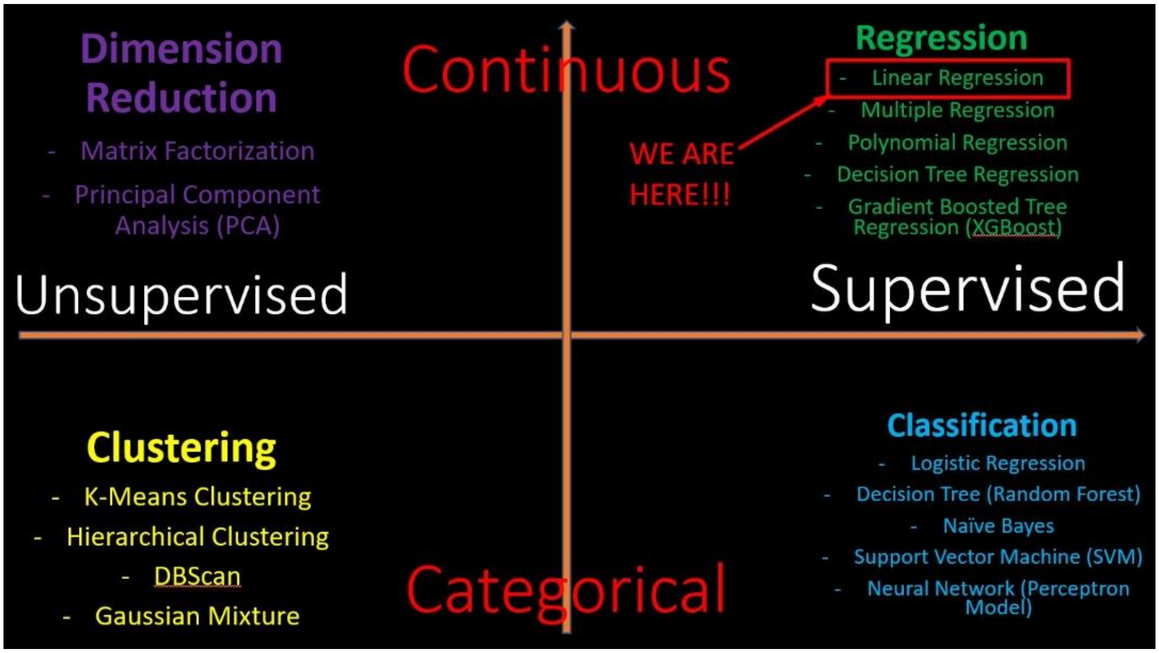
SIMPLE LINEAR REGRESSION USING PYTHON

DR. ALVIN ANG

CONTENTS

I. INTRODUCTION



II. PYTHON - USING STATSMODEL

(ADVERTISING.CSV)

A. LOAD AND GLANCE

- Dataset can be found here: https://www.alvinang.sg/s/Advertising.csv
- https://www.alvinang.sg/s/Simple_Linear_Regression_with_Statsmodel_by_Dr_Alvin_Ang.ipynb

```python
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. INITIALIZE AND FIT LINEAR MODEL

```python
import statsmodels.formula.api as smf

# Initialise and fit linear regression model using `statsmodels`
model = smf.ols('Sales ~ TV', data=advert)
model = model.fit()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the
  import pandas.util.testing as tm
```

- Y ~ Sales
- X ~ TV (advertising)

C. PRODUCE THE MODEL

```
model.params

#Sales = 7.032 + 0.047*TV
```

```
Intercept    7.032594
TV           0.047537
dtype: float64
```

D. PREDICT THE MODEL

```
new_X = 400
model.predict({"TV": new_X})

# if X (TV advertising costs) = $400,
# Then Y (Predicted Sales) will = 26 units
```

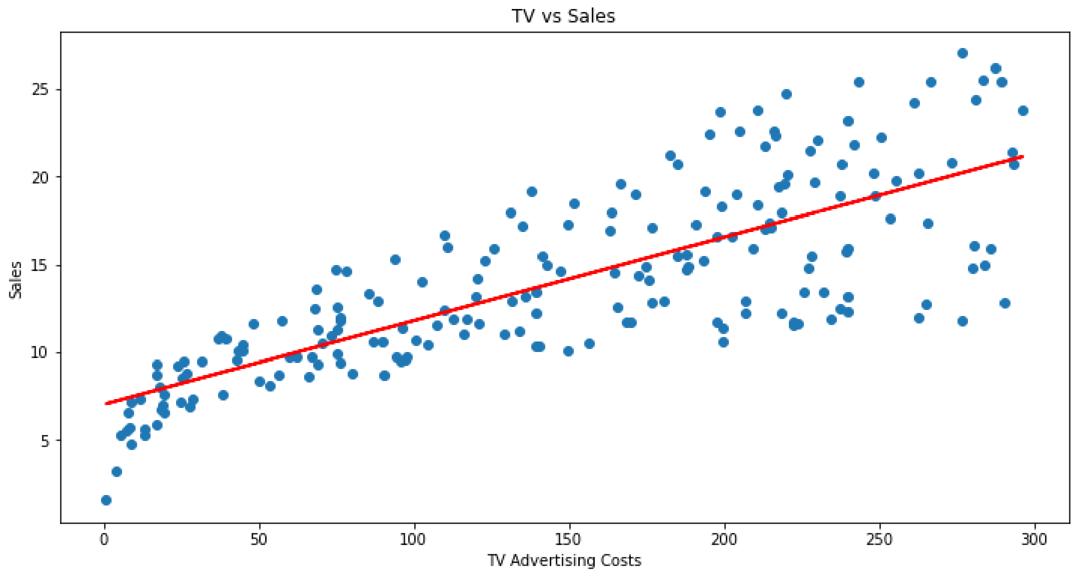
```
0    26.04725
dtype: float64
```

E. STORE THE PREDICTION MODEL

```
# Predict values
sales_pred = model.predict()
```

F. PLOT

```
from matplotlib import pyplot as plt

# Plot regression against actual data
plt.figure(figsize=(12, 6))

# scatter plot showing actual data
plt.plot(advert['TV'], advert['Sales'], 'o')

# regression line
plt.plot(advert['TV'], sales_pred, 'r', linewidth=2)

#cosmetics
plt.xlabel('TV Advertising Costs')
plt.ylabel('Sales')
plt.title('TV vs Sales')

plt.show()
```



- The dataset is here:
    - https://www.alvinang.sg/s/automobileEDA.csv
    - https://www.alvinang.sg/s/Simple_Linear_Regression_using_SKLearn_by_Dr_Alvin_Ang.ipynb

A. LOAD AND GLANCE

```
[2] 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:

ymboling	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 × 29 columns

B. PART II: VISUALIZE / PLOT THE REGRESSION MODEL

1. STEP 1: LOAD THE LR MODULES AND CREATE THE LR OBJECT

```
[4] from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
lm
```

```
LinearRegression()
```

2. STEP 2: DEFINE OUR X AND Y

```
X = df[['highway-mpg']]
Y = df['price']
```

3. STEP 3: FIT / TRAIN THE LINEAR MODEL

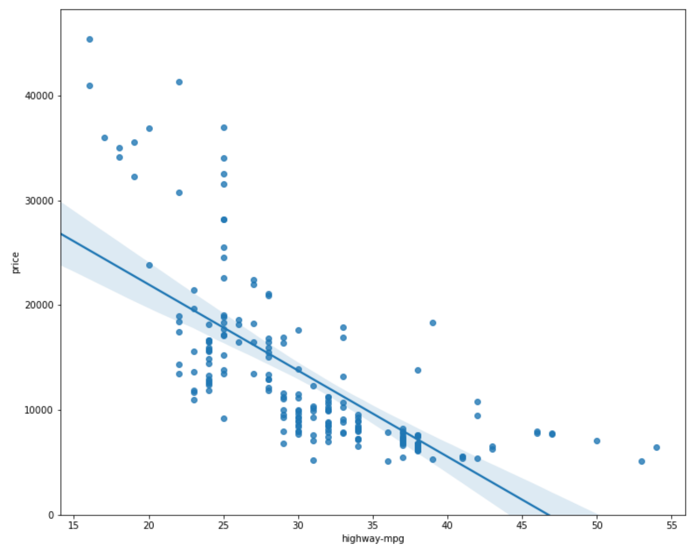
```
lm.fit(X,Y)
```

```
LinearRegression()
```

4. STEP 4: VISUALIZE PRICE VS HIGHWAY-MPG

```
import seaborn as sns
%matplotlib inline

width = 12
height = 10
plt.figure(figsize=(width, height))
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

```
(0.0, 48180.533904764896)
```

- 

- Comments:
  - Price is negatively correlated to highway-mpg.
  - The data points are scattered badly around the regression line.
  - A linear model is NOT the best fit.

C. PART III: GENERATE A LINEAR REGRESSION EQUATION

1. STEP 1: FIND THE Y-INTERCEPT

- Y-Intercept refers to the C of the Y = mX + C.

```
lm.intercept_
```

```
38423.3058581574
```

2. STEP 2: FIND THE GRADIENT

- Gradient refers to the m of the Y = mX +C

```
lm.coef_
```

```
array([-821.73337832])
```

- This means that the Linear Equation is
    - price = 38423.31 - 821.73 x highway-mpg → Y = C + mX

3. STEP 3: TEST SOME PREDICTIONS

- Since we already have the LR Equation Y = mX +C, we test it using the first 5 rows of values of the Dataset.

```python
Yhat=lm.predict(X)
Yhat[0:5]
```

```
array([16236.50464347, 16236.50464347, 17058.23802179, 13771.3045085 ,
       20345.17153508])
```

- Note that the first 5 rows of the "highway-mpg" are as follows:

highway-mpg	price
27	13495
27	16500
26	16500
30	13950
22	17450

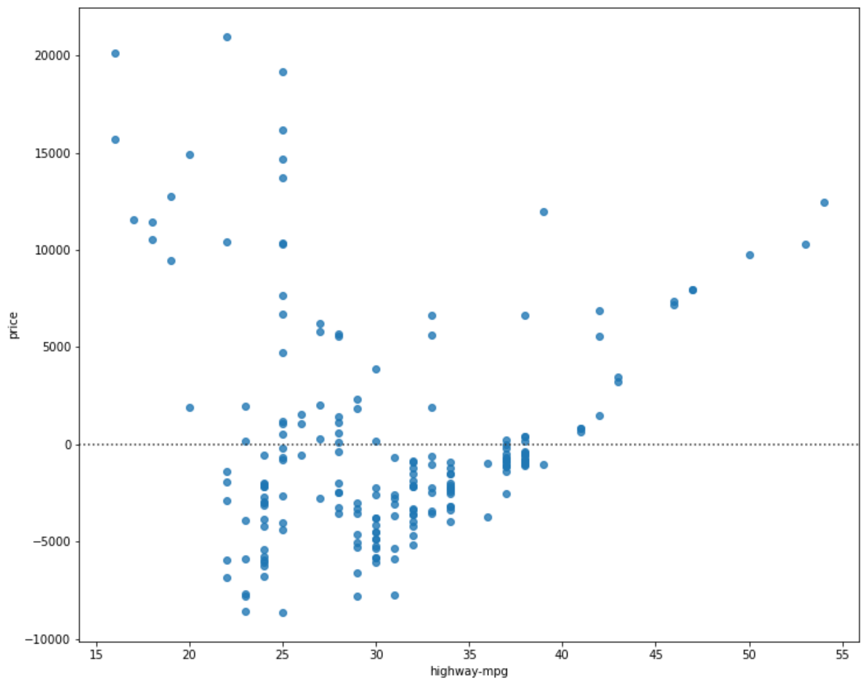
- In other words, the "forecasted" values in the prediction array were using the values
    - 27 / 27 / 26 / 30 / 22
- This differs quite a bit from the real pricings!

D. PART IV: USE A RESIDUAL PLOT TO VISUALLY INSPECT IF LINEAR REGRESSION FITS THE MODEL

- Residual plot has been described and defined here:
    - https://www.alvinang.sg/s/Multiple-Regression-MR-by-Dr-Alvin-Ang.pdf
    - A residual plot is a graph that shows the residuals on the vertical y-axis and the independent variable on the horizontal x-axis.
- What is a Residual? The difference between the observed value (y) and the predicted value (Yhat).
- If the points in a Residual Plot are randomly spread out around the x-axis, then a linear model is appropriate for the data.
- Because randomly spread out residuals means that the variance is constant, and thus the linear model is a good fit for this data.

```
width = 12
height = 10
plt.figure(figsize=(width, height))
sns.residplot(df['highway-mpg'], df['price'])
plt.show()
```

- Output:

    - 

- Comments:
    - This residual plot shows that the residuals are not randomly spread around the x-axis.
    - Maybe a non-linear model is more appropriate for this data.

E. PART V: USE R2 AND MSE AS INDICATORS TO DETERMINE THE ACCURACY OF THE LINEAR REGRESSION 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 (ŷ).

1. STEP 1: CALCULATE THE R2 FOR "HIGHWAY_MPG" VS "PRICE"

```
#highway_mpg_fit
lm.fit(X, Y)
# Find the R^2
print('The R-square is: ', lm.score(X, Y))
```

```
The R-square is:  0.4965911884339176
```

- Comment:
  - We can say that ~ 49.659% of the variation of the "price" is explained by this simple linear model "highway_mpg".
  - Below 50% means that actually a linear model is not a good fit…which means that the actual data is far from the fitted line…

2. STEP 2: CALCULATE THE MSE

a) Firstly, predict the output "yhat"

```
Yhat=lm.predict(X)
print('The output of the first four predicted value is: ', Yhat[0:4])
```

```
The output of the first four predicted value is:  [16236.50464347 16236.50464347 17058.23802179 13771.3045085 ]
```

b) "mean_squared_error"

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(df['price'], Yhat)
print('The mean square error of price and predicted value is: ', mse)
```

```
The mean square error of price and predicted value is:  31635042.944639888
```

- Comment:
  - At this point, we are unable to say if MSE is high or low.
  - MSE is used to measure against another method of fitting i.e. it cannot be used as a standlone measure.
  - That is, currently we are doing Linear Regression (LR) for model fitting and we have this MSE.
  - We can only compare this MSE with another MSE of another model fit… E.g. Multiple Regression (MR)… in which we will showcase this in another article.

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.