Linear Regression in Python

Dr. Junya Michanan

What is Linear Regression

Linear regression is used to predict the value of an outcome variable *Y* based on one or more input predictor variables *X*. The aim is to establish a linear relationship (a mathematical formula) between the predictor variable(s) and the response variable, so that, we can use this formula to estimate the value of the response *Y*, when only the predictors (*Xs*) values are known.

Introduction

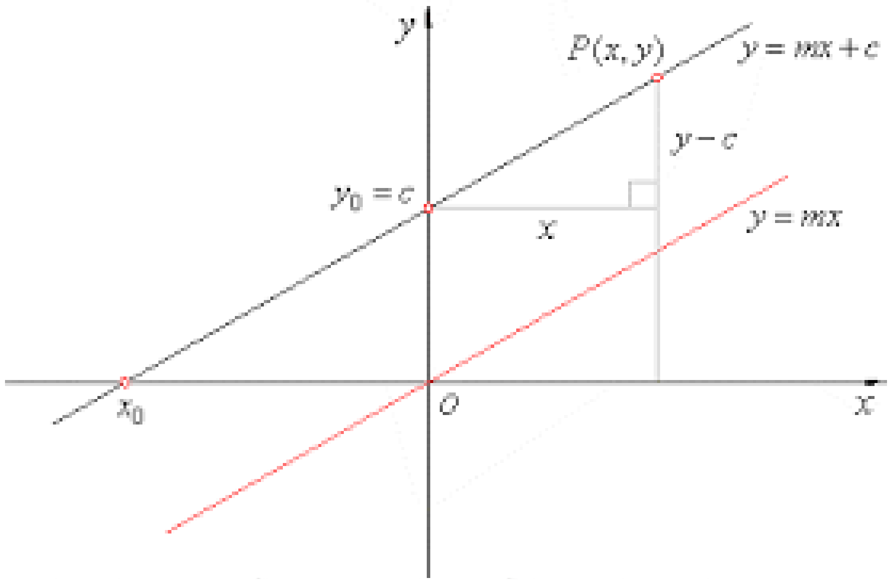
The aim of linear regression is:

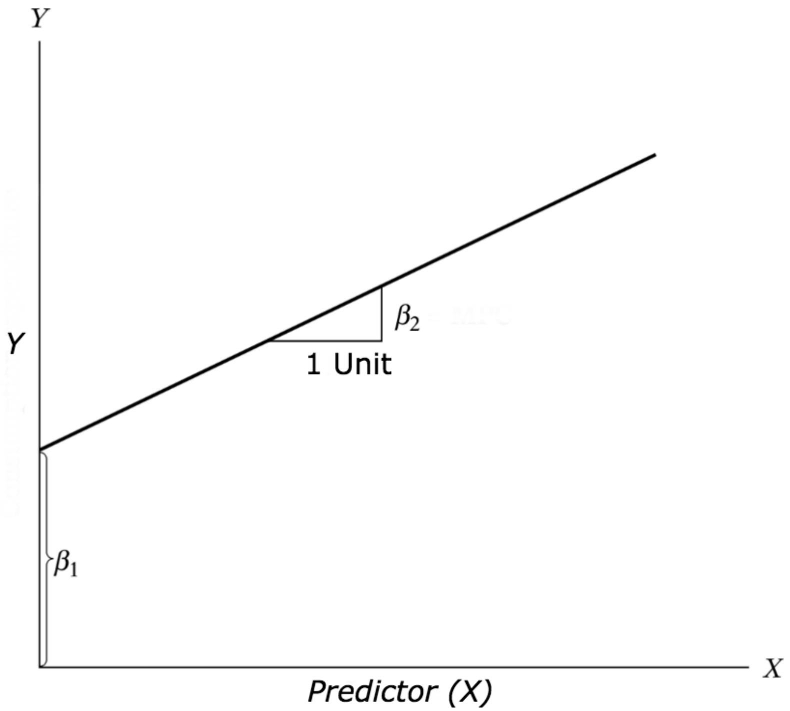
To model a continuous variable *Y* as a mathematical function of one or more *X* variable(s), so that we can use this regression model to predict the *Y* when only the *X* is known. This mathematical equation can be generalized as follows:

$Y = mX + c$

$Y = \beta_1 + \beta_2 X + \epsilon$

where, β_1 is the intercept and β_2 is the slope. Collectively, they are called *regression coefficients*. ϵ is the error term, the part of *Y* the regression model is unable to explain.





mtcars dataset

- ***Goal:*** to build a simple regression model that we can use to predict MPG by establishing a statistically significant linear relationship with DISP and other mtcars variables
- ***MTCars Dataset:***
    - *The 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).*
    - *A data frame with 32 observations on 11 (numeric) variables.*
        1. *mpg* *Miles/(US) gallon*
        2. *cyl* *Number of cylinders*
        3. *disp* *Displacement (cu.in.)*
        4. *hp* *Gross horsepower*
        5. *drat* *Rear axle ratio*
        6. *wt* *Weight (1000 lbs)*
        7. *qsec* *1/4 mile time*
        8. *vs* *Engine (0 = V-shaped, 1 = straight)*
        9. *am* *Transmission (0 = automatic, 1 = manual)*
        10. *gear* *Number of forward gears*
        11. *carb* *Number of carburetors*.

model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1
Duster 360	14.3	8	360	245	3.21	3.57	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.44	18.3	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.44	18.9	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.07	17.4	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.73	17.6	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.78	18	0	0	3	3
Cadillac Fleetwood	10.4	8	472	205	2.93	5.25	17.98	0	0	3	4

Step1: Importing packages

```
# Step 1: Importing packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn import linear_model
from sklearn import metrics
import numpy as np
from math import log
```

- ***pandas*** for DataFrame manipulation
- ***matplotlib*** and ***seaborn*** for graphs and data plots
- ***sklearn*** for Linear Regression modeling and Model Evaluation
- ***numpy*** for numeric calculation
- ***math*** for mathematics calculation

Step2: Import data and Basic Statistical Analysis

- Read data from CSV file
- Peak the first 5 rows
- Get the data info
- Get data statistical info
- Check null data
- Print keys

```
# Loading mtcars dataset and printing information about it
data = pd.read_csv('mtcarsDataset.csv')
print(type(data))
print(data.head())
print(data.info())
print(data.describe())
print(data.isnull().sum())
# Printing keys
print(data.keys())
```

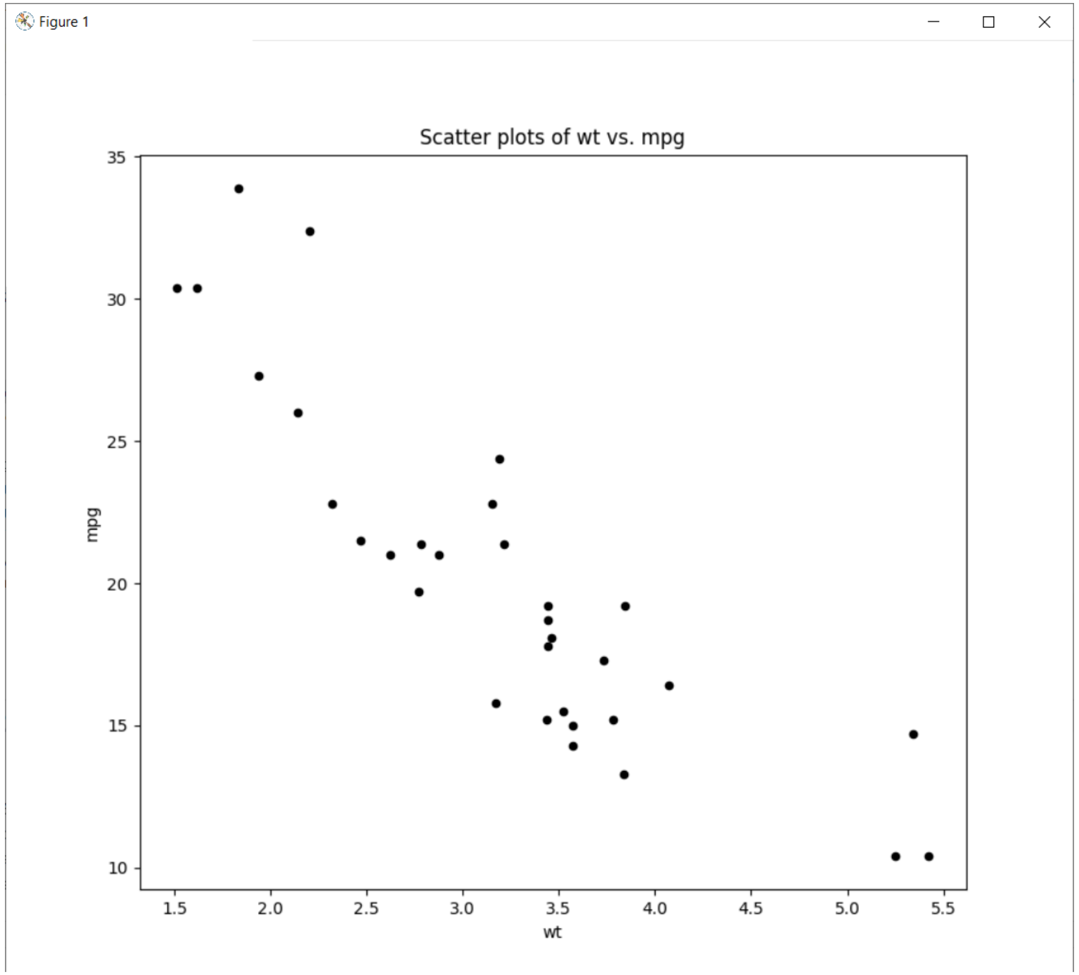
Step3: Graphical Analysis

- We can display each of the independent variables (predictors), the following plots are drawn to visualize the following behavior:

1. **Scatter plot**: Visualize the linear relationship between the predictor and response
2. **Box plot**: To spot any ***outlier*** observations in the variable. Having outliers in your predictor can drastically affect the predictions as they can easily affect the direction/slope of the line of best fit.
3. **Density plot**: To see the **distribution** of the predictor variable. Ideally, a close to **normal distribution** (a bell shaped curve), without being skewed to the left or right is preferred. Let us see how to make each one of them.

Step3: Graphical Analysis – Scatter plot

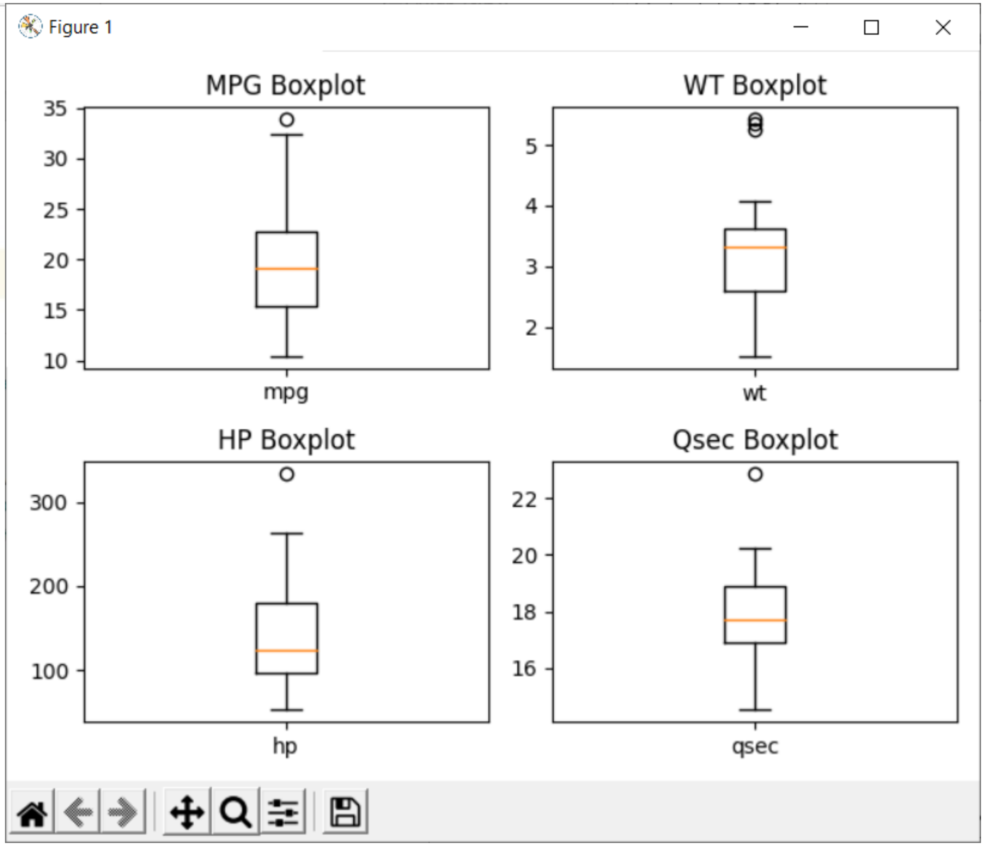
```
#Scatter plot
data.plot(kind="scatter",
          x="wt",
          y="mpg",
          figsize=(9,9),
          color="black",
         title='Scatter plots of wt vs. mpg');

# Showing plot
plt.show()
```



Scatter plots can help visualize any linear relationships between the dependent (response) variable and independent (predictor) variables.

Step3: Graphical Analysis – Box or whisker plot

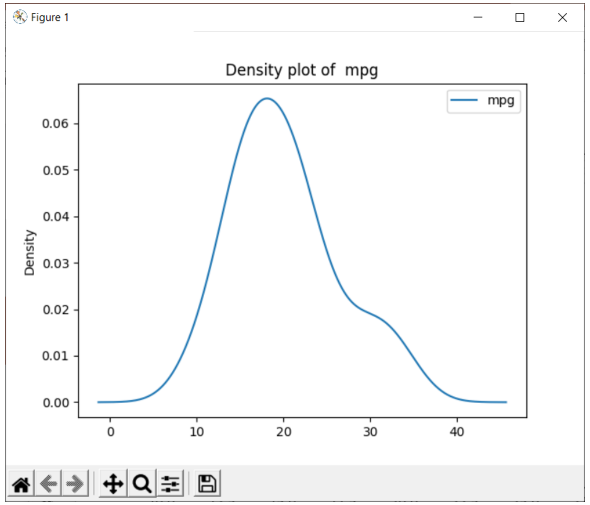
```
#Box plots
#4x4 Layouts of plots ax1, ax2, ax3, ax4
fig, ((ax1, ax2),(ax3, ax4)) = plt.subplots(nrows=2, ncols=2)

#mpg
ax1.set_title('MPG Boxplot')
ax1.boxplot(data['mpg'], labels=['mpg'])

#wt
ax2.set_title('WT Boxplot')
ax2.boxplot(data['wt'], labels=['wt'])

#qsec
ax3.set_title('HP Boxplot')
ax3.boxplot(data['hp'], labels=['hp'])

#qsec
ax4.set_title('Qsec Boxplot')
ax4.boxplot(data['qsec'], labels=['qsec'])

#Set a tight layout
plt.tight_layout()
plt.show()
```

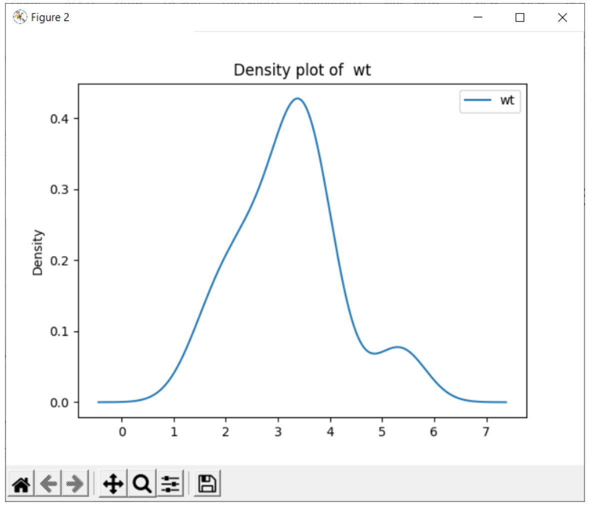


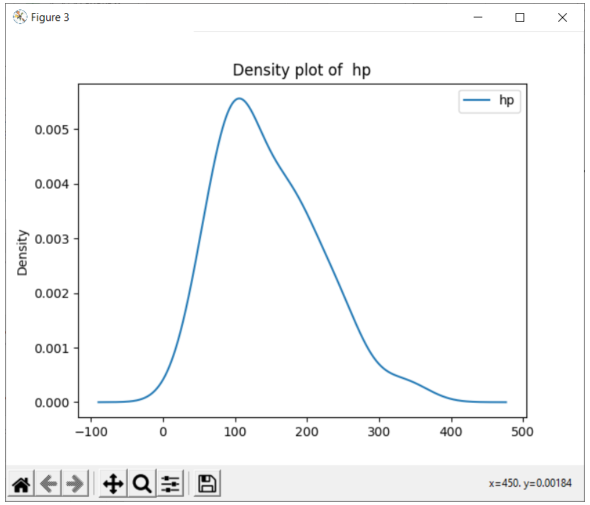
Outliers -- any datapoint that lies outside the 1.5 * interquartile-range (1.5 * *IQR*) is considered an outlier, where IQR is calculated as the distance between the 25th percentile and 75th percentile values for that variable. Outlier ค่าสุดโต่ง เป็นค่าที่สูงหรือต่ำผิดปกติ โปรแกรมทางสถิติส่วนใหญ่นิยม Plot ข้อมูลเป็น Outlier เมื่อข้อมูลนั้นมีค่าน้อยกว่า Q1-1.5*(Q3-Q1) หรือมีค่าสูงกว่า Q3+1.5*(Q3-Q1)

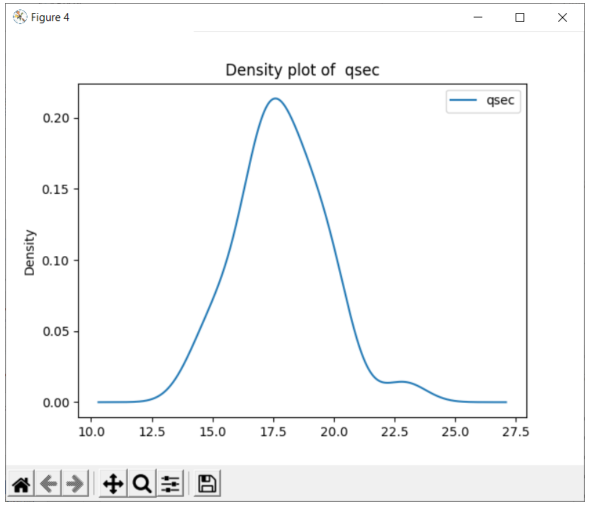
Step3: Graphical Analysis – Density plots

```
#Density plot
plt1=pd.DataFrame(data['mpg'])
plt1.plot(kind="density",title='Density plot of  mpg');

plt2=pd.DataFrame(data['wt'])
plt2.plot(kind="density",title='Density plot of  wt');

plt3=pd.DataFrame(data['hp'])
plt3.plot(kind="density",title='Density plot of  hp');

plt4=pd.DataFrame(data['qsec'])
plt4.plot(kind="density",title='Density plot of  qsec');
plt.show()
```





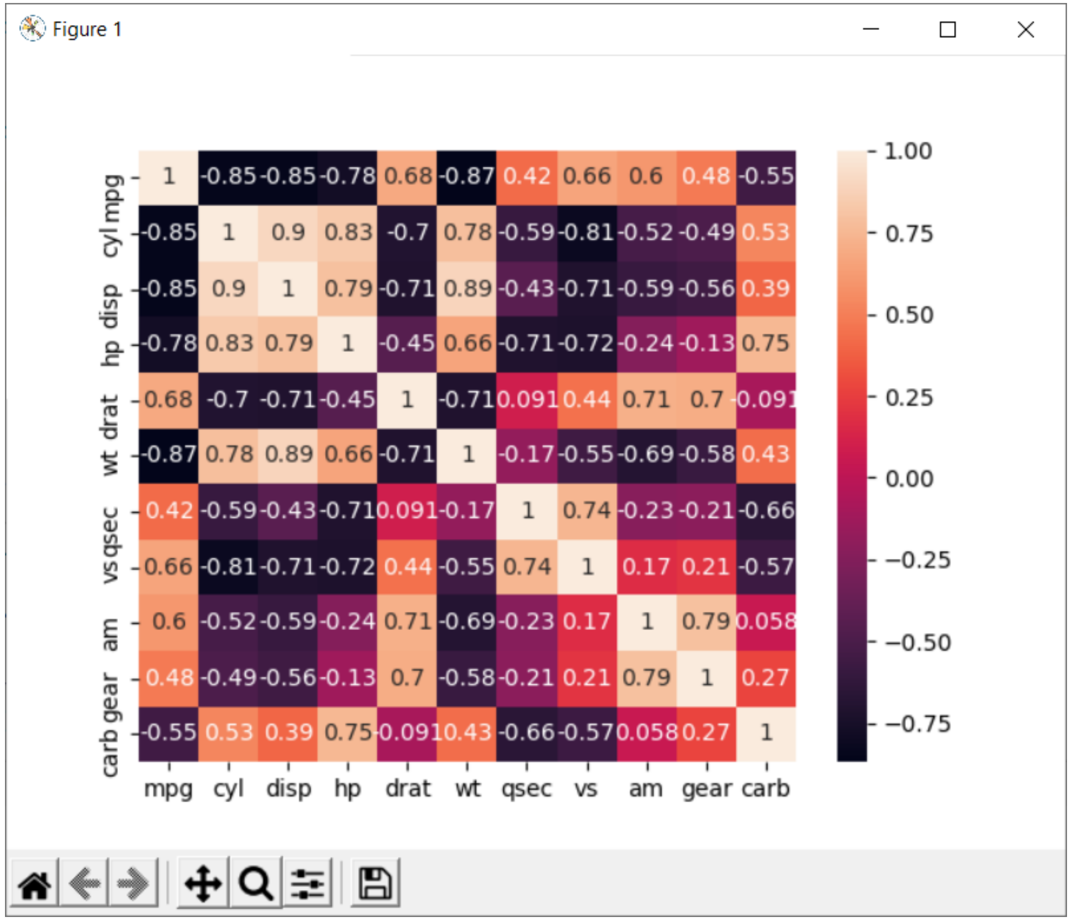




Check if the response variable is close to normality

Step3: Graphical Analysis – Correlation matrix

```python
#Corellation matrix
print("Correlation matrix")
corrMatrix = data.corr()
print(corrMatrix )
sn.heatmap(corrMatrix, annot=True)
plt.show()
```

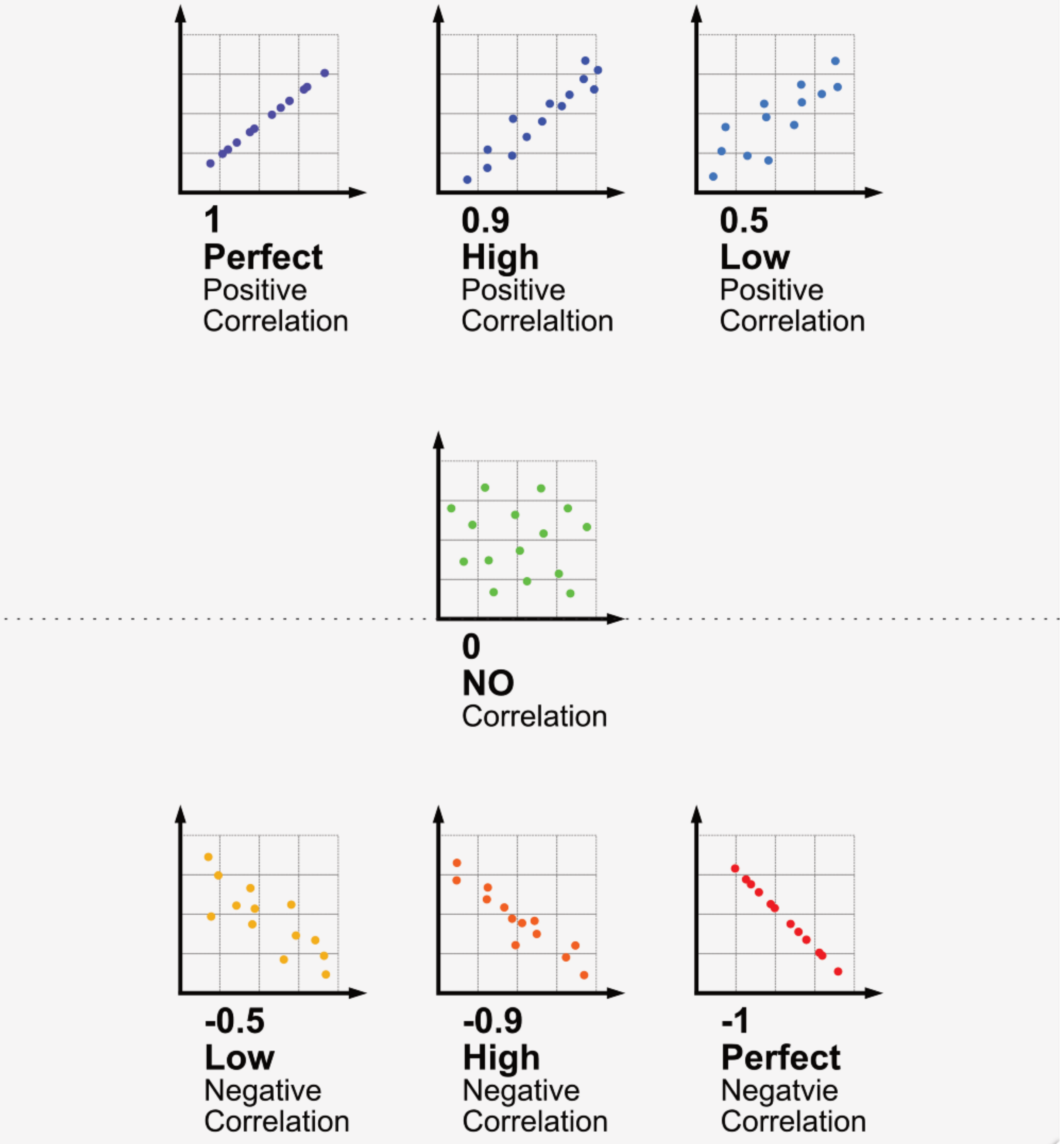


Correlation (ค่าสหสัมพันธ์)

- Correlation is a statistical measure that suggests the level of linear dependence between two variables, that occur in pair – just like what we have here in speed and dist.
- Correlation can take values between -1 to +1.
- A high **positive correlation** between them and therefore the correlation between them will be closer to 1.
- The opposite is true for an **inverse relationship or negative correlation**, in which case, the correlation between the variables will be close to -1.

Coefficient of Correlation

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$



Step 4: Machine Learning modeling using a Simple Linear Regression

```
# Step 4:ML Modeling using a Simple Linear Regression method
#Linear regression is used to fit the best line to the data
print('#######1 Simple Linear Regression#########')
# Initialize model
regression_model = linear_model.LinearRegression()

# Train the model using the mtcars data
regression_model.fit(X = pd.DataFrame(data["wt"]),
                     y = data["mpg"])

# Check trained model y-intercept
print(regression_model.intercept_)

# Check trained model coefficients
print(regression_model.coef_)

#It can be seen that the y-intercept term is set to 37.2851 and the coefficient for the weight variable is -5.3445,
# making the equation => mpg = 37.2851 - 5.3445 * wt.

# Check R-squared
score = regression_model.score(X = pd.DataFrame(data["wt"]),
                               y = data["mpg"])

print("R-squared value: ")
print(score)
```

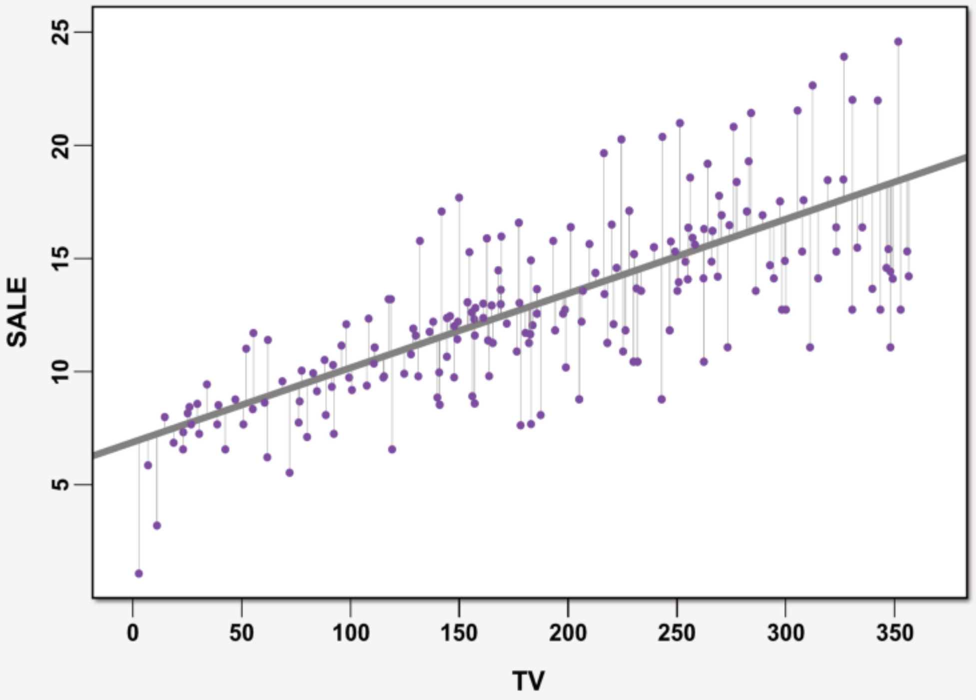
```
37.28512616734204
[-5.34447157]
```

The output above shows the model intercept and coefficients used to create the best fit line. In this case the y-intercept term is set to 37.2851 and the coefficient for the weight variable is -5.3445. In other words,

mpg = y_intercept + (β ∗ wt)

=> mpg = 37.2851 - 5.3445 * wt

Simple Linear Regression



$$y = ax + b$$

a = Slope
b = Y-Intercept

Multiple Linear Regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i + \varepsilon$$

β คือ Coefficient หรือ ค่าสัมประสิทธิของค่าประมาณการตัวนั้นๆ

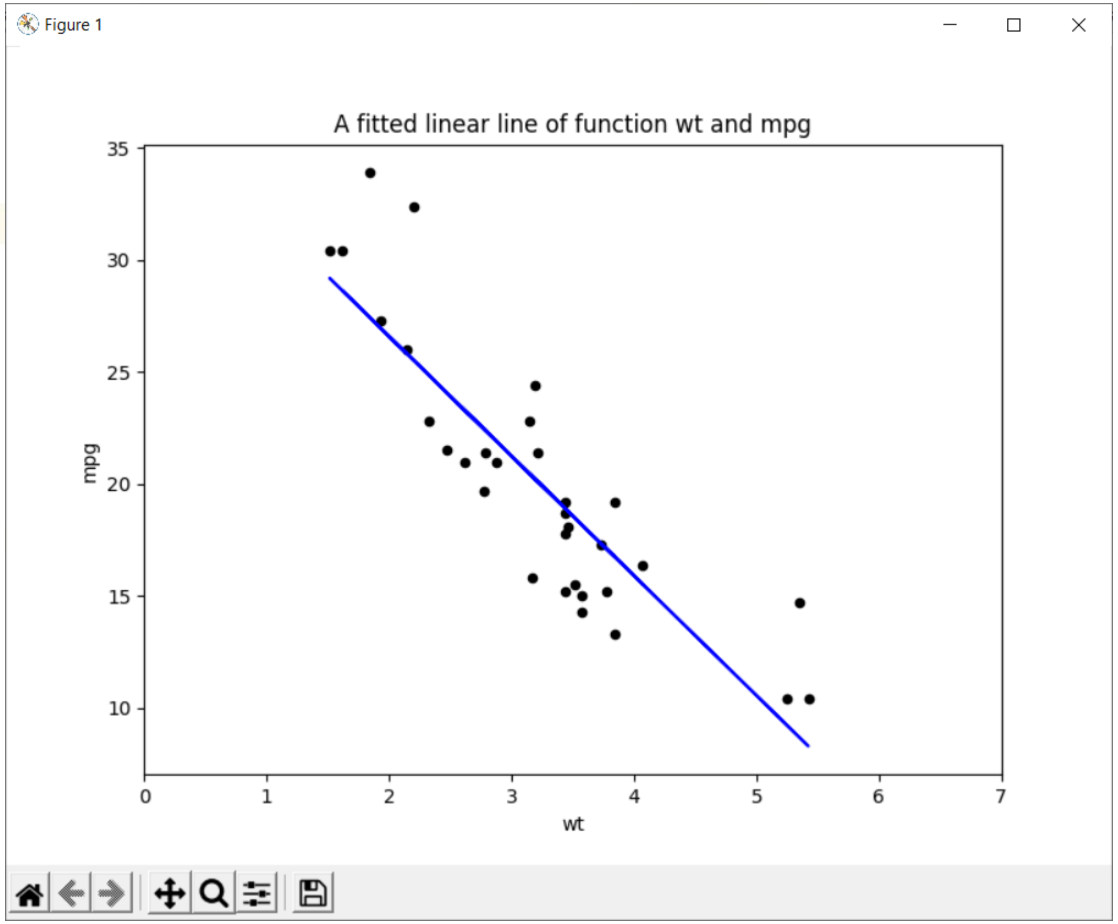
Step 5: Making prediction

```python
#Step 5: Making predictions
print("###Prediction 1 ###")
WT1 = pd.DataFrame([3.52])
print(WT1)
predicted_MPG = regression_model.predict(WT1)
print(predicted_MPG)
print("Predicted MPG of WT = {0} is {1}".format(WT1[0][0], predicted_MPG[0]))

print("###Prediction 2###")
WT2 = pd.DataFrame([4.32, 6.55])
print(WT2)
predicted_MPG2 = regression_model.predict(WT2)
print(predicted_MPG2)
print("Predicted MPG of WT = {0} is {1}".format(WT2[0][0], predicted_MPG2[0]))
print("Predicted MPG of WT = {0} is {1}".format(WT2[0][1], predicted_MPG2[1]))

#Prediction from wt values
train_prediction = regression_model.predict(X = pd.DataFrame(data["wt"]))
```

```
#######1 Simple Linear Regression#########
37.28512616734204
[-5.34447157]
Score:
0.752832793658264б
###Prediction 1 ###
      0
0  3.52
[18.47258623]
Predicted MPG of WT = 3.52 is 18.472586231358218
###Prediction 2###
      0
0  4.32
1  6.55
[14.19700897  2.27883737]
Predicted MPG of WT = 4.32 is 14.197008973180072
Predicted MPG of WT = 6.55 is 2.278837366008503
```

Plotting linear regression line

```
#Prediction from wt values
train_prediction = regression_model.predict(X = pd.DataFrame(data["wt"]))

# Actual - prediction = residuals
residuals = data["mpg"] - train_prediction
print(residuals.describe())

data.plot(kind="scatter",
           x="wt",
           y="mpg",
           figsize=(9,9),
           color="black",
           xlim = (0,7),
          title='A fitted linear line of function wt and mpg')

# Plot regression line
plt.plot(data["wt"],      # Explanitory variable
         train_prediction, # Predicted values
         color="blue");

plt.show()
```



Step 6: Model evaluation – R-squared, MAE, MSE, RMSE, AIC, BIC

```
#Step 6: Model Evaluation
#R-squared value
print("R-squared value: ")
print(score)
#Calculate MAE, MSE, RMSE
y_true = data["mpg"]
y_pred = train_prediction
print("Calculated MAE, MSE, RMSE:")
print(metrics.mean_absolute_error(y_true, y_pred))
print(metrics.mean_squared_error(y_true, y_pred))
print(np.sqrt(metrics.mean_squared_error(y_true, y_pred)))

#Calculating AIC
# number of parameters
num_params = len(regression_model.coef_) + 1
n = len(y_true)
mse = metrics.mean_squared_error(y_true, y_pred)

print("AIC:")
aic = n * log(mse) + 2 * num_params
print(aic)

print("BIC:")
bic = n * log(mse) + num_params * log(n)
print(bic)
```

```
R-squared value:
0.7528327936582646
Calculated MAE, MSE, RMSE:
2.340641858325169
8.697560548229477
2.949162685955028
AIC:
73.21736286677714
BIC:
76.1488346723766
37.22727011644721
```

โดยวิธีอ่านตารางข้างบนเบื้องต้น จะมองที่ค่าตัวแปรต่าง ๆ ดังนี้

- **R-Square หรือ Coefficient of determination** เป็นค่าที่ใช้พิสูจน์ว่า Model ที่ได้นั้นเหมาะสมหรือไม่ โดยค่า R-Square จะอยู่ระหว่าง 0-1 ซึ่งยิ่งเข้าใกล้ 1 ยิ่งดี และโดยทั่วไปควรมีค่า 0.6 ขึ้นไป และจะดีมากเมื่อมีค่ามากกว่า 0.8 ขึ้นไป แต่ก็ไม่ได้มีกฎเกณฑ์แน่นอนตายตัวแล้วแต่กรณี

- **Adjusted R-Square** เป็นค่าที่บ่งบอกว่า R-Square ที่ได้นั้นเหมาะสมหรือไม่ โดยการลดจำนวนชุดข้อมูลลง 1 ชุด แล้วคำนวณ R-Square ใหม่ ซึ่งหากมีค่าต่ำกว่า R-Square มากผิดปกติก็สามารถสรุปได้ว่าใช้ชุดข้อมูลน้อยเกินไป หรือ โมเดลนั้นมีผลสืบเนื่องจากการเปลี่ยนแปลงจำนวนชุดข้อมูลเป็นอย่างมาก มีโอกาสที่ Model จะผิดพลาดสูง ค่า Adjusted R-Square สำหรับโมเดลที่ดีจะมีค่าต่ำกว่า R-Square เล็กน้อย

- **Coefficients** คือ ค่าสัมประสิทธ์ที่มีผลต่อตัวแปรนั้นๆ ดังที่ได้อธิบายไว้ข้างต้น

- **P-value** คือ ความน่าจะเป็นที่จะได้ผลลัพธ์เท่ากับหรือเกินกว่าที่สังเกตได้ภายใต้สมมุติฐานหลัก มีความสำคัญอย่างยิ่งในการทำ Regression เพราะมันเป็นตัวบ่งบอกว่า ตัวแปรนั้นๆ มีความสำคัญต่อระบบความสัมพันธ์หรือสมการหรือไม่ ซึ่งค่า P-value จะอยู่ระหว่าง 0-1 วิธีการอ่าน P-value ง่ายๆ คือ ค่า P-value ที่มากกว่า 0.05 บ่งบอกว่าตัวแปรนั้นอาจจะมีไม่ความสำคัญต่อระบบสมการ ซึ่งเราสามารถลองตัดออกแล้วทำการ Regression ใหม่ เพื่อหาสมการที่เหมาะสมต่อไปได้

Model Evaluation Metrics for Regression

For classification problems, we have only used classification accuracy as our evaluation metric.
What metrics can we used for regression problems?
Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

AIC and BIC

- The Akaike's information criterion - AIC (Akaike, 1974) and the Bayesian information criterion - BIC (Schwarz, 1978) are measures of the goodness of fit of an estimated statistical model and can also be used for model selection. Both criteria depend on the maximized value of the likelihood function L for the estimated model.
- The AIC is defined as:

  $$\boldsymbol{AIC = (-2) \times ln(L) + (2 \times k)}$$

- where, k is the number of model parameters and the BIC is defined as:

  $\boldsymbol{BIC = (-2) \times ln(L) + k \times ln(n)}$ where, n is the sample size.
- For model comparison, the model with the lowest AIC and BIC score is preferred

How to know if the model is best fit for your data?

STATISTIC	**CRITERION**
R-Squared	Higher the better (> 0.70)
Adj R-Squared	Higher the better
F-Statistic	Higher the better
Std. Error	Closer to zero the better
t-statistic	Should be greater 1.96 for p-value to be less than 0.05
AIC	Lower the better
BIC	Lower the better
Mallows cp	Should be close to the number of predictors in model
MAPE (Mean absolute percentage error)	Lower the better
MSE (Mean squared error)	Lower the better
Min_Max Accuracy => mean(min(actual, predicted)/max(actual, predicted))	Higher the better

Step 7: Multiple Linear Regression

What is multiple linear regression?

- Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.
- The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

Formula and Calculation of Multiple Linear Regression

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

$y_i =$ dependent variable

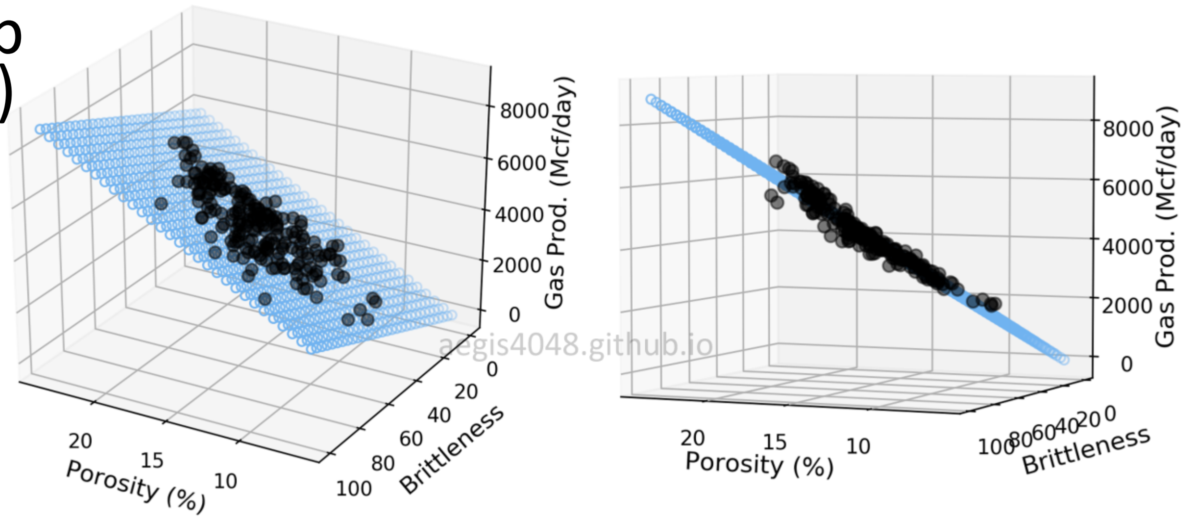
$x_i =$ explanatory variables

$\beta_0 =$ y-intercept (constant term)

$\beta_p =$ slope coefficients for each explanatory variable

$\epsilon =$ the model's error term (also known as the residuals)

3D multiple linear regression model



Step 7: Multiple Linear Regression

```
#Step 7: ML Modeling of Multiple Linear Regression
#When more x variables are added, a multiple linear regression model is used.
#The multiple linear regression model produced is
# mpg = 37.22727011644721 -3.87783074wt -0.03177295hp

print('#######2 Multiple Linear Regression#########')
# Initialize model
multi_reg_model = linear_model.LinearRegression()

# Train the model using the mtcars data
multi_reg_model.fit(X = data.loc[:,["wt","hp"]],
                    y = data["mpg"])

# Check trained model y-intercept
print("Y-intercept and slope: ")
print(multi_reg_model.intercept_)

# Check trained model coefficients (scaling factor given to "wt")
print(multi_reg_model.coef_)

# Check R-squared
print("R-squared value: ")
score = multi_reg_model.score(X = data.loc[:,["wt","hp"]],
                              y = data["mpg"])

print(score)
```

Step 8: Exercise – 15 points

Create linear multiple regression to predict mpg from:

1. multi_reg_model1 ==> mpg = m1*wt+ m2*qsec + c1
2. multi_reg_model2 ==> mpg = m1*wt+ m2*qsec+ m3*hp + c1
3. multi_reg_model3 ==> mpg = m1*wt+ m2*qsec+ m3*hp+ m4*drat + c1