$$\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i'=1}^{n} x_{i'}^2}$$

$$\Rightarrow \hat{y}_i = x_i \hat{\beta} = \frac{x_i \sum_{i'=1}^{n} x_{i'} y_{i'}}{\sum_{j=1}^{n} x_j^2} = \sum_{i'=1}^{n} \underbrace{\boxed{\frac{x_i x_{i'}}{\sum_{j=1}^{n} x_j^2}}}_{= a_{i'}} y_{i'}$$

5. Consider the fitted values that result from performing linear regression without an intercept. In this setting, the ith fitted value takes the form

$$\hat{y}_i = x_i \hat{\beta},$$

where

$$\hat{\beta} = \left(\sum_{i=1}^{n} x_i y_i\right) / \left(\sum_{i'=1}^{n} x_{i'}^2\right). \tag{3.38}$$

Show that we can write

$$\hat{y}_i = \sum_{i'=1}^{n} a_{i'} y_{i'}.$$

What is $a_{i'}$?

[2번 문제] ESL 3.4

Ex. 3.4 Show how the vector of least squares coefficients can be obtained from a single pass of the Gram–Schmidt procedure (Algorithm 3.1). Represent your solution in terms of the QR decomposition of $\mathbf{X}$.

$$\hat{\beta} = (X^T X)^{-1} X^T Y \quad \Leftarrow X = QR$$

$$= ((QR)^T QR)^{-1} (QR)^T Y$$

$$= (R^T Q^T Q R)^{-1} R^T Q^T Y$$

$$= R^{-1} (R^T)^{-1} R^T Q^T Y$$

$$= R^{-1} Q^T Y$$

$$\therefore R\hat{\beta} = Q^T Y$$

$$\cdot\ R\hat{\beta} = \begin{bmatrix} (x_1 \cdot q_1) & (x_2 \cdot q_1) & \cdots & (x_p q_1) \\ 0 & (x_2 \cdot q_2) & \cdots & (x_p q_2) \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & (x_p q_p) \end{bmatrix} \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{bmatrix} = \begin{bmatrix} (x_1 \cdot q_1)\hat{\beta}_1 + (x_2 \cdot q_1)\hat{\beta}_2 + \cdots + (x_p q_1)\hat{\beta}_p \\ (x_2 \cdot q_2)\hat{\beta}_2 + \cdots + (x_p q_2)\hat{\beta}_p \\ \vdots \\ (x_p \cdot q_p)\hat{\beta}_p \end{bmatrix}$$

$$\cdot\ Q^T Y = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_p \end{bmatrix} Y = \begin{bmatrix} q_1 Y \\ q_2 Y \\ \vdots \\ q_p Y \end{bmatrix}$$

$\Rightarrow$ 하나씩 비교해서 계수추출가능

```python
import ssl
import pandas as pd
ssl._create_default_https_context = ssl._create_unverified_context #Github에서 데이터를 바로 불러오도록 하는 세팅입니다. 해당 코드 무시하고 데이터 받아서 쓰셔도 됩니다!
data = pd.read_csv('https://github.com/YonseiESC/ESC-21SUMMER/blob/main/week1/HW/week1_data.csv?raw=True')
y = data['mpg']
x = data.drop(['mpg'],axis=1)
```

```python
import numpy as np
# numpy 모듈만을 이용해주세요.
```

```python
sum(data['horsepower']=='?')
```

```
5
```

```python
data=data[data['horsepower']!='?']
```

결측치제거

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 396
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    object 
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   year          392 non-null    int64  
dtypes: float64(3), int64(3), object(1)
memory usage: 24.5+ KB
```

```python
data['horsepower']=data['horsepower'].astype('float')
```

형식맞추기

```python
y=data['mpg']
x=data.drop('mpg',axis=1)
```

변수설정

```python
def YourOwnRegression(x,y):
    beta = np.dot(np.dot(np.linalg.inv(np.dot(x.T, x)), x.T), y)
    yhat = np.dot(x, beta)
    print(beta, yhat)
```

함수설정

```python
YourOwnRegression(x,y)
```

```
[-0.5226089   0.01022108 -0.020873   -0.00639456 -0.05202195  0.61025869] [15.93081361 14.45720405 16.11263762 15.93670419 16.10071197 10.51021782
 10.27543153 10.53128391  9.67525181 13.49634208 15.59946068 15.17857818
 14.95056695 18.23756934 23.85149163 20.70116218 21.04691637 22.47738545
 25.4075601  27.85849004 21.93938914 23.54965655 23.61026333 24.5700506
 22.02475307  7.48992449  8.73758029  8.68094775  6.39448743 26.01781879
 25.50668622 25.43458909 22.95714525 17.50355598 18.56684981 18.98997897
 18.64504732 11.74185816 10.43958016 12.2762232  12.39844199  7.02172716
  8.71466792  6.09084578 20.89073634 24.7795066  18.89340516 20.0843144
 25.76559234 26.24104628 26.30753902 26.59195154 28.28090927 29.28279009
 28.2609542  27.13912286 25.6470868  26.69569542 26.07663641 24.9880403
 26.2074288  11.93647037 11.52899822 12.73330182 13.0723569  15.65493241
  9.60277136 10.60920749 10.79899231 10.95329347 25.45996983 14.19610698
 13.24891771 11.63170439 13.0781754  21.23759877 24.50545484 21.19494316
 26.4550875  25.15275892 25.40418994 24.27095036 26.56185652 26.71587043
 13.39863077 16.2651911  14.74101165 13.99404444 15.68483539  8.35588404
 12.1559116  12.08192325 12.63929331  9.52710571  8.09046672 15.38896808
 20.7055219  19.98343495 22.03288824 21.95534612 22.05191343 28.9278806
  8.64356655  8.94866353 10.06261966 10.76884679 23.08246175 26.05144779
 26.01877494 25.52924044 27.53361915 26.19737345 24.2284252  26.29152354
 14.1399229  11.80551478 29.17763168 27.47281611 24.48514487 22.211482
 18.18001246 23.66132966 21.80938422 16.18633836 21.37003802 22.92450763
 20.27483103 29.01987066 26.1144079  29.60478409 25.79483636 17.42925127
 18.20305761 18.16965645 13.93463354 10.62233937 11.89477047 10.6524927
 12.92654367 27.29002111 29.11881818 26.97024946 31.35712023 28.84708125
 28.08036911 28.1370547  27.59838633 25.57416431 26.13565935 28.85620844
 21.24635801 20.04621563 20.64756809 22.47771079 11.68048944 13.0191654
 12.19083021 11.60515866 16.62277687 17.11534294 18.13462735 17.75872575
 22.48538772 20.67124626 21.09405268 28.38999264 25.6178386  23.45163703
 25.94471506 25.06438254 28.04340891 25.7120626  22.52118358 30.02318129
 21.7091604  24.86417853 23.27558145 23.28386212 24.73315284 31.11419428
 27.02526368 28.70906066 26.57792197 28.30139996 28.71233562 14.76463858
 14.84864669 16.74288177 14.99473679 21.98169626 21.41233074 23.42150596
 23.09392064 29.90480163 29.05099549 30.6230356  31.72965517 19.18911918
 20.22456108 19.35231654 22.55428679 31.49394493 30.08735698 29.08175554
 27.10777245 22.52930717 16.43688261 21.62943493 23.04180738 17.16039851
 13.38138443 16.19537407 17.04478509 17.57825889 30.44250197 29.8139795
```