Regression-3

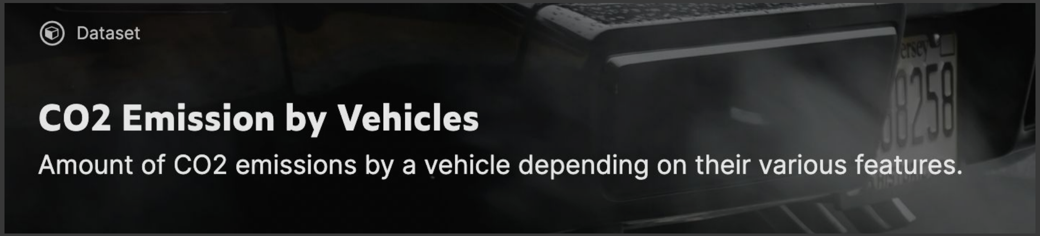
김도겸 , 류승환

목차

CO2 Emission

- 차량의 특정 요소와 CO2 발생량의 관계를 EDA 및 Linear Regression 분석

- 캐나다 정부(open.canada.ca/data) -> kaggle Dataset



데이터

	Make	Model	Vehicle Class	Transmission	Fuel Type
count	7385	7385	7385	7385	7385
unique	42	2053	16	27	5
top	FORD	F-150 FFV 4X4	SUV - SMALL	AS6	X
freq	628	32	1217	1324	3637

```
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 12 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Make                               7385 non-null   object
 1   Model                              7385 non-null   object
 2   Vehicle Class                      7385 non-null   object
 3   Engine Size(L)                     7385 non-null   float64
 4   Cylinders                          7385 non-null   int64
 5   Transmission                       7385 non-null   object
 6   Fuel Type                          7385 non-null   object
 7   Fuel Consumption City (L/100 km)   7385 non-null   float64
 8   Fuel Consumption Hwy (L/100 km)    7385 non-null   float64
 9   Fuel Consumption Comb (L/100 km)   7385 non-null   float64
 10  Fuel Consumption Comb (mpg)        7385 non-null   int64
 11  CO2 Emissions(g/km)                7385 non-null   int64
```

Make : 생산 회사

Model : 자동차 모델

Vehicle Class : 유틸리티, 용량 및 중량에 따른 차량 등급

Engine Size(L) : 엔진의 크기

Cylinders : 실린더 수

Transmission : 기어 수가있는 변속기 유형

Fuel Type : 사용된 연료의 유형

Fuel Consumption City(L/100km) : 도시 도로의 연료 소비량(L/100km)

Fuel Consumption Hwy(L/100km) : 고속도로에서 연료 소비(L/100km)

Fuel Consumption Comb(L/100km) : 복합연비 (55% 도시, 45%고속도로)

CO2_emission : CO2 배출량

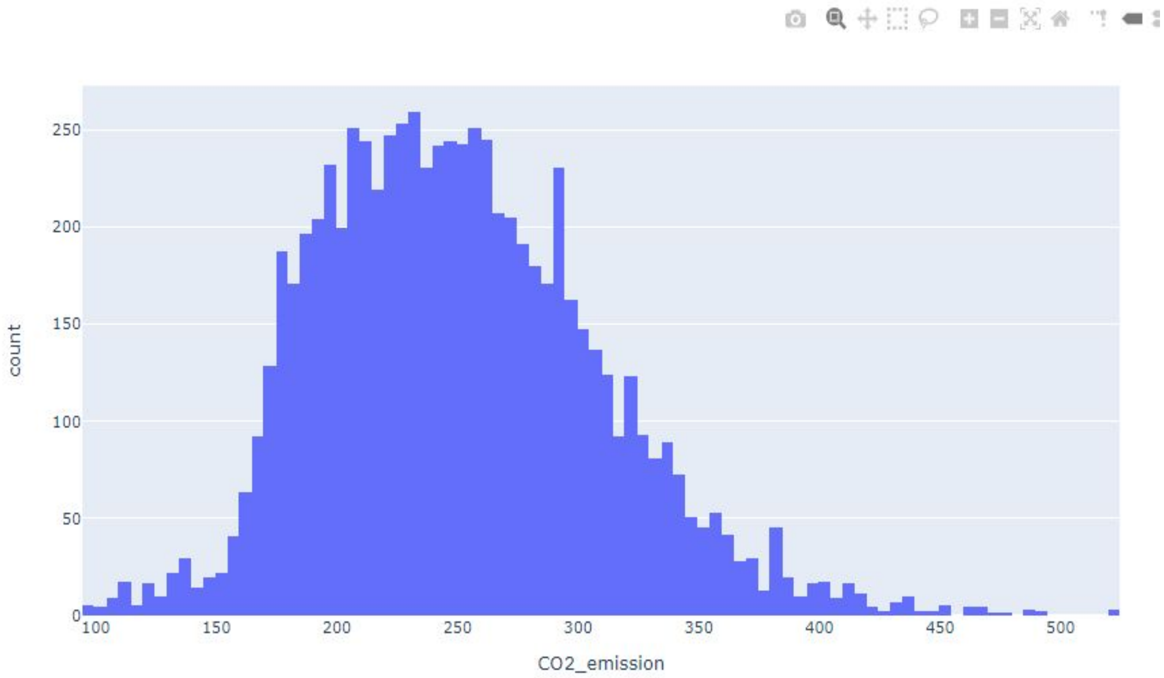
EDA

1. CO2 배출량과의 x_feature들과의 상관관계

2. 생산회사별 cylinder의 갯수와 CO2 배출량의 상관관계

CO2 배출량에 대해서 히스토그램.

In [12]:
```
import plotly.express as px

fig = px.histogram(df2, x='CO2_emission')
fig.show()
```



- CO_2 배출량에 대한 Histogram
- 정규화분포를 이루고 있음
- CO_2 배출에 가장 큰 영향을 미치는 요소 확인

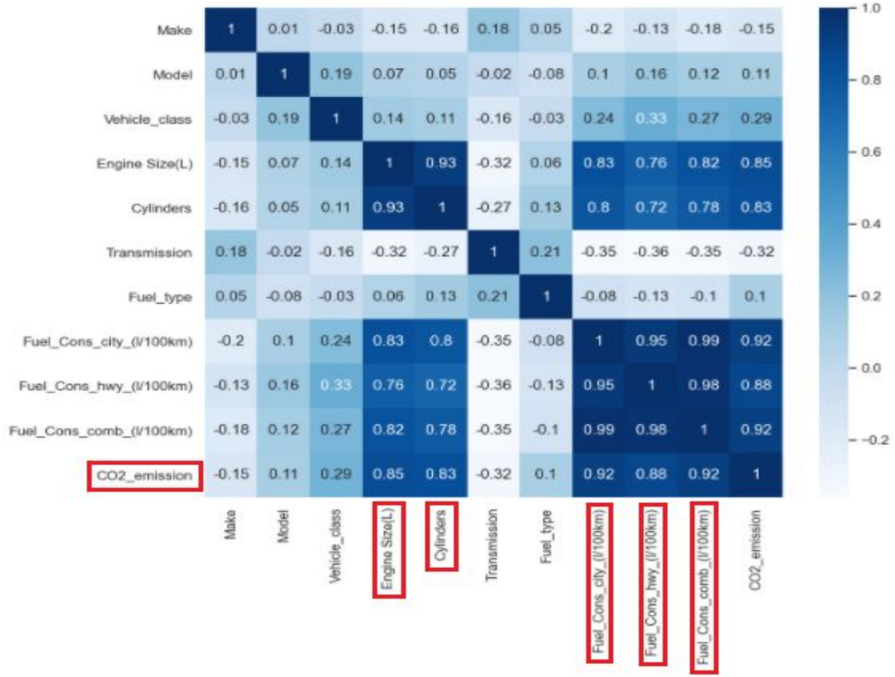
- CO2 배출량과의 상관관계

corr()

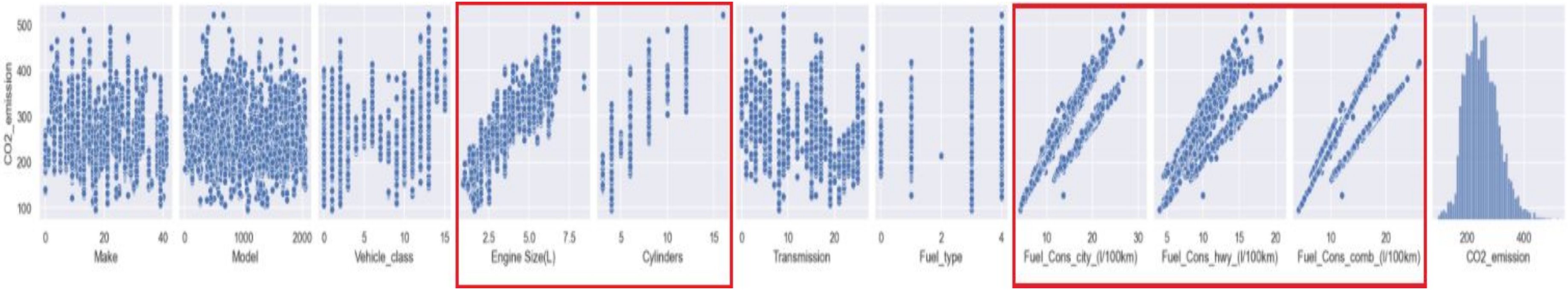
```
df2.corr()['CO2_emission'].sort_values(ascending=False)
```

```
CO2_emission               1.000000
Fuel_Cons_city_(l/100km)   0.919592
Fuel_Cons_comb_(l/100km)   0.918052
Fuel_Cons_hwy_(l/100km)    0.883536
Engine Size(L)             0.851145
Cylinders                  0.832644
Vehicle_class              0.286468
Model                      0.105847
Fuel_type                  0.100306
Make                      -0.151955
Transmission              -0.316660
Name: CO2_emission, dtype: float64
```

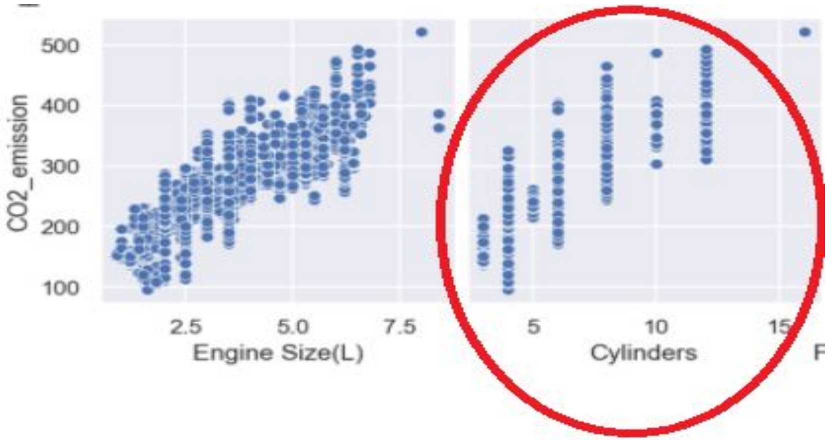
<AxesSubplot:>



- 1. CO2배출량과 다른 x_feature들과의 관계



- 2. Cylinder의 갯수와 CO2배출량과의 상관관계 분석



In [455]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

f, ax = plt.subplots(1, 2, figsize=(18,8))

df['Make'].value_counts()
```

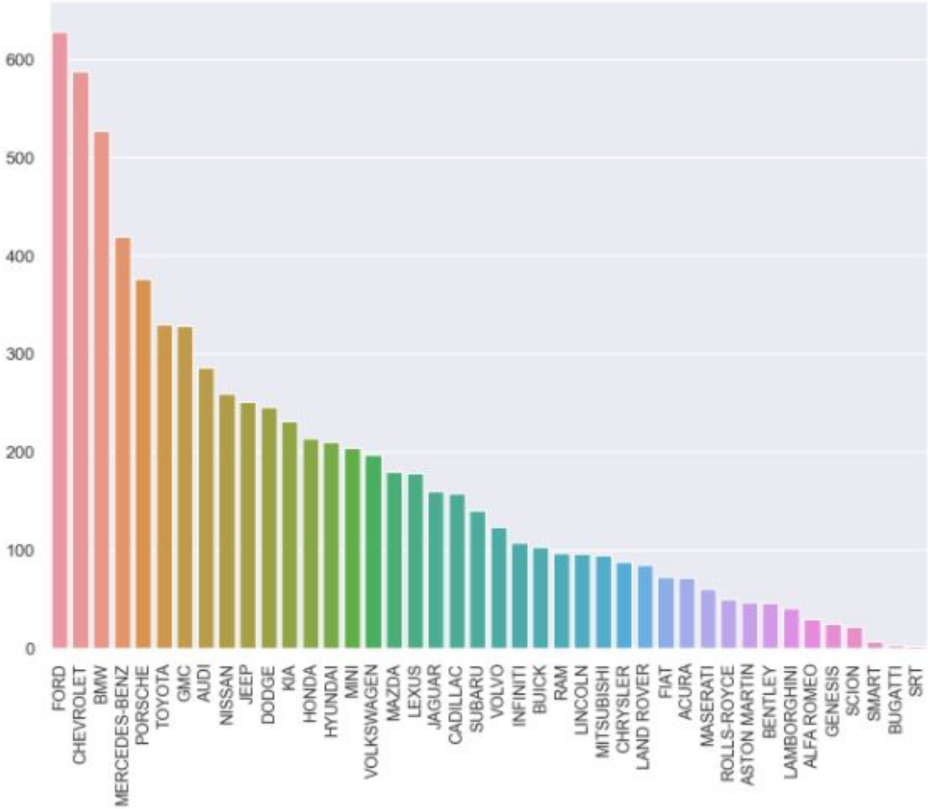
Out[455]:
```
FORD             628
CHEVROLET        588
BMW              527
MERCEDES-BENZ    419
PORSCHE          376
TOYOTA           330
GMC              328
AUDI             286
NISSAN           259
JEEP             251
DODGE            246
KIA              231
HONDA            214
HYUNDAI          210
MINI             204
VOLKSWAGEN       197
MAZDA            180
LEXUS            178
JAGUAR           160
CADILLAC         158
SUBARU           140
VOLVO            124
INFINITI         108
BUICK            103
RAM               97
LINCOLN           96
MITSUBISHI        95
CHRYSLER          88
LAND ROVER        85
FIAT              73
ACURA             72
MASERATI          61
ROLLS-ROYCE       50
ASTON MARTIN      47
BENTLEY           46
LAMBORGHINI       41
ALFA ROMEO        30
GENESIS           25
SCION             22
SMART              7
BUGATTI            3
SRT                2
Name: Make, dtype: int64
```

```python
pd.crosstab(df['Make'], df['CO2_emission'], margins=True)
```

CO2_emission Make	96	99	102	103	104	105	106	108	109	110	...	465	467	473	476	485	487	488	493	522	All
ACURA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	72
ALFA ROMEO	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	30
ASTON MARTIN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	47
AUDI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	286
BENTLEY	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	46
BMW	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	527
BUGATTI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	3	3
BUICK	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	103
CADILLAC	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	158
CHEVROLET	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	588
CHRYSLER	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	88
DODGE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	246
FIAT	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	73
FORD	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	628
GENESIS	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	25
GMC	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	328
HONDA	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	214
HYUNDAI	4	1	1	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	210
INFINITI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	108
JAGUAR	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	160
JEEP	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	251
KIA	0	0	0	0	0	0	0	0	0	4	...	0	0	0	0	0	0	0	0	0	231
LAMBORGHINI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	1	0	2	0	41
LAND ROVER	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	85
LEXUS	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	178
LINCOLN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	96
MASERATI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	61
MAZDA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	180
MERCEDES-BENZ	0	0	0	0	0	0	0	0	0	0	...	0	1	1	1	0	0	0	0	0	419
MINI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	204
MITSUBISHI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	95
NISSAN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	259
PORSCHE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	376
RAM	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	97
ROLLS-ROYCE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	50
SCION	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	22
SMART	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	7
SRT	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
SUBARU	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	140
TOYOTA	0	0	0	0	1	3	2	2	2	2	...	0	0	0	0	0	0	0	0	0	330
VOLKSWAGEN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	197
VOLVO	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	124
All	4	1	1	1	2	3	2	2	2	7	...	3	1	1	1	1	1	1	2	3	7385

- 생산 회사별 데이터와 회사별 대비 CO2 발생량 확인

브랜드 데이터



1. 각 회사별 데이터 파악
2. 상위 10개 회사별 cylinder와 CO2 상관관계 파악
3. 회사별 Rmse, coef, intercept 파악

Label Encoding

	Make	Model	Vehicle_class	Engine Size(L)	Cylinders	Transmission	Fuel_type	Fuel_Cons_city_(l/100km)	Fuel_Cons_hwy_(l/100km)	Fuel_Cons_comb_(l/100kr
0	0	1057	0	2.0	4	14	4	9.9	6.7	8
1	0	1057	0	2.4	4	25	4	11.2	7.7	9
2	0	1058	0	1.5	4	22	4	6.0	5.8	5
3	0	1233	11	3.5	6	15	4	12.7	9.1	11
4	0	1499	11	3.5	6	15	4	12.1	8.7	10
...	...	...	...	...	...	...	...	...	...	
7380	41	1951	11	2.0	4	17	4	10.7	7.7	9
7381	41	1957	11	2.0	4	17	4	11.2	8.3	9
7382	41	1960	11	2.0	4	17	4	11.7	8.6	10
7383	41	1968	12	2.0	4	17	4	11.2	8.3	9
7384	41	1969	12	2.0	4	17	4	12.2	8.7	10

7385 rows × 11 columns

Regression

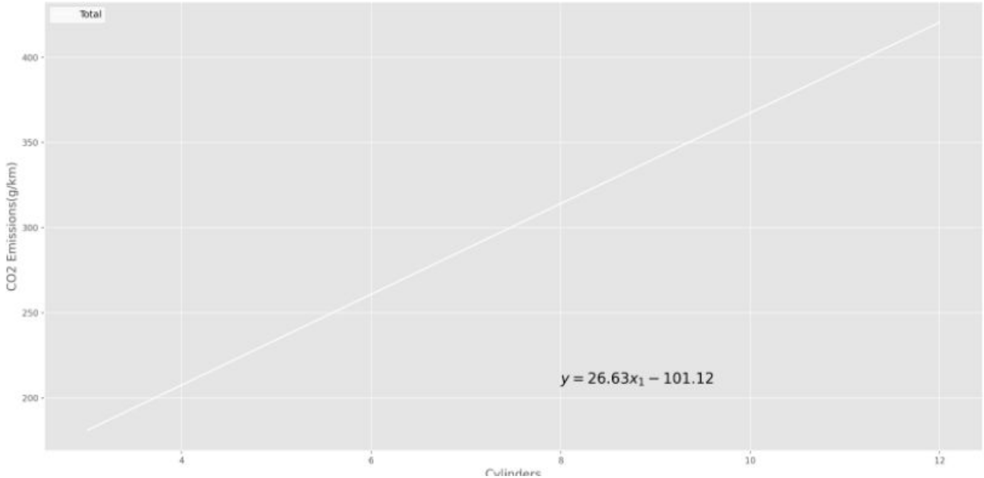
1. 브랜드 별 실린더 ~ co2 발생량

2. 전체 데이터 회기 분석

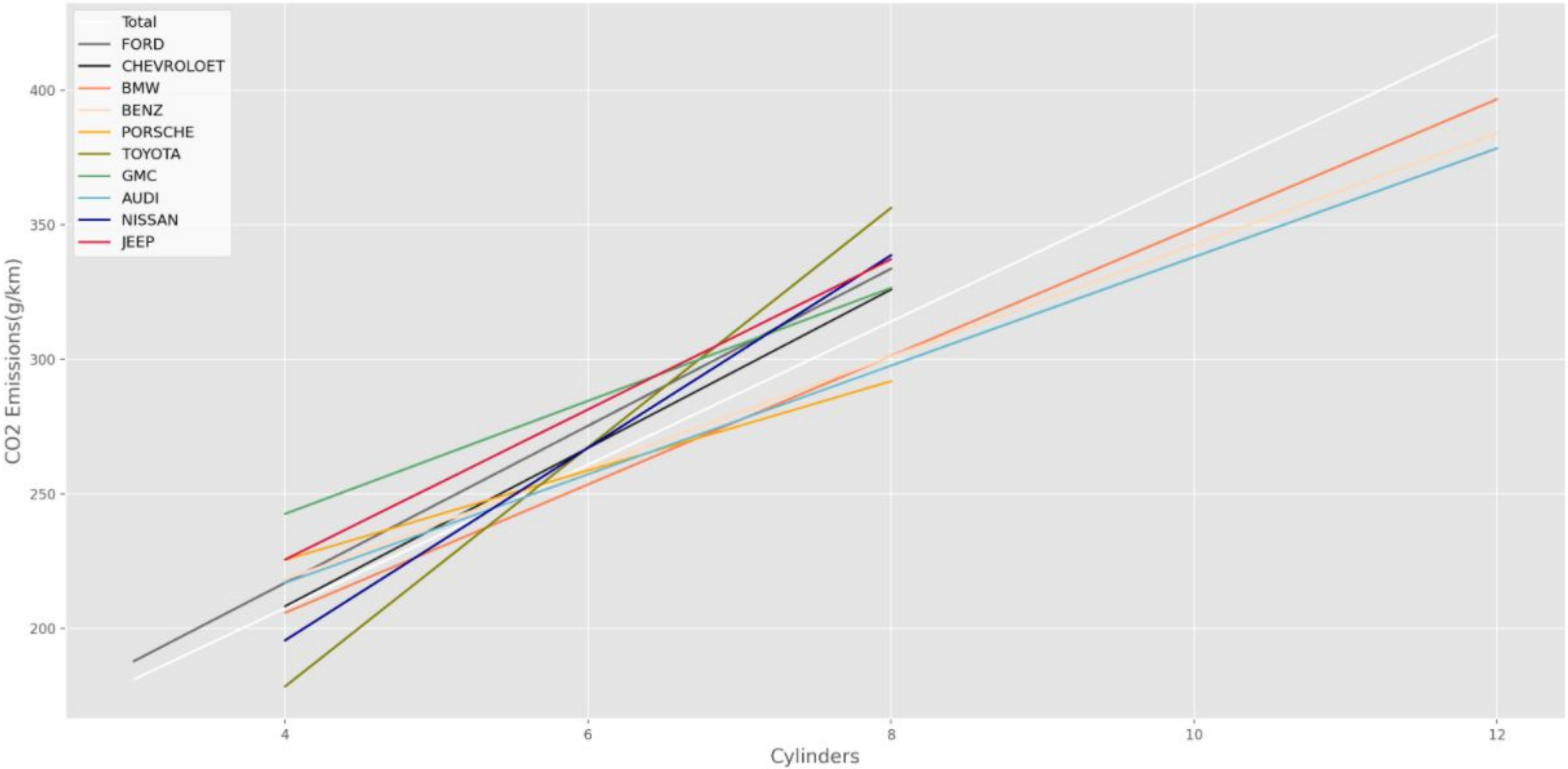
3. 변속기와 실린더 ~ co2 발생량

```
In [337]: from sklearn.model_selection import train_test_split

          X = df2['Cylinders'].values
          y= df2['CO2_emission'].values

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=13)

          X_train = X_train.reshape(-1, 1)
          X_test = X_test.reshape(-1, 1)

In [338]: from sklearn.linear_model import LinearRegression

          reg = LinearRegression()
          reg.fit(X_train, y_train)

Out[338]: LinearRegression()

In [339]: from sklearn.metrics import mean_squared_error

          pred_tr = reg.predict(X_train)
          pred_test = reg.predict(X_test)

          rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
          rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

          print('RMSE of Train Data : ', rmse_tr)
          print('RMSE of Test Data : ', rmse_test)

          RMSE of Train Data :  32.28411396985763
          RMSE of Test Data :  32.9041974300707

In [340]: reg.intercept_, reg.coef_

Out[340]: (101.11555359789514, array([26.62620671]))
```

< 전체 cylinder별 CO2 배출량 그래프 >



그래프



Total
FORD
CHEVROLOET
BMW
BENZ
PORSCHE
TOYOTA
GMC
AUDI
NISSAN
JEEP
CO2 Emissions(g/km)
400
350
300
250
200
4
6
8
10
12
Cylinders

브랜드 별 RMSE, Coef table

Name	RMSE_train	RMSE_test	Coef	Intercept
FORD	32.64	30.04	29.19	100.22
CHEVROLOET	36.56	36.70	29.43	90.55
BMW	22.81	22.71	23.87	110.35
BENZ	34.21	38.39	20.74	135.16
PORSCHE	24.53	24.77	16.61	159.08
TOYOTA	39.71	44.09	$\boxed{44.47}$	0.58
GMC	32.67	35.43	20.99	158.64
AUDI	23.90	25.56	20.20	136.14
NISSAN	28.33	34.41	35.79	52.45
JEEP	19.87	22.98	27.91	113.96

- Coef가 가장 높은 회사가 CO2 배출량이 많다고 볼 수 있음

브랜드 별 RMSE, Coef table

Name	RMSE_train	RMSE_test	Coef	Intercept	Cyl_2	Cyl_4	Cyl_6	Cyl_8
FORD	32.64	30.04	29.19	100.22	158.60	216.98	275.36	333.74
CHEVROLOET	36.56	36.70	29.43	90.55	149.41	208.27	267.13	325.99
BMW	22.81	22.71	23.87	110.35	158.09	205.82	253.55	301.29
BENZ	34.21	38.39	20.74	135.16	176.65	218.14	259.63	301.12
PORSCHE	24.53	24.77	16.61	159.08	192.29	225.50	258.71	291.92
TOYOTA	39.71	44.09	44.47	0.58	89.52	178.47	267.42	356.37
GMC	32.67	35.43	20.99	158.64	200.63	242.62	284.60	326.59
AUDI	23.90	25.56	20.20	136.14	176.53	216.93	257.32	297.71
NISSAN	28.33	34.41	35.79	52.45	124.02	195.59	267.17	338.74
JEEP	19.87	22.98	27.91	113.96	169.78	225.61	281.43	337.25

One-hot encoding

```
data_with_dummies = data_reg.copy()

col_to_1hot = ['Vehicle Class','Transmission','Fuel Type','Cylinders']
prfix_1hot = ['V-Cls', 'Trans', 'Fl-T','Cyl']

for col, pfx in zip(col_to_1hot, prfix_1hot):
    fuel_1hot = pd.get_dummies(data_reg[col], prefix=pfx, drop_first=True)
    data_with_dummies = data_with_dummies.join(fuel_1hot)
```

Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)	...	Fl-T_N	Fl-T_X	Fl-T_Z	Cyl_4	Cyl_5	Cyl_6	Cyl_8	Cyl_10	Cyl_12	Cyl_16
33	196	...	0	0	1	1	0	0	0	0	0	0
29	221	...	0	0	1	1	0	0	0	0	0	0
48	136	...	0	0	1	1	0	0	0	0	0	0
25	255	...	0	0	1	0	0	1	0	0	0	0
27	244	...	0	0	1	0	0	1	0	0	0	0

LinearRegression fit : 이상하다

```
X_train, X_test, y_train, y_test = train_test_split(final_data, labels, test_size=0.2, random_state=12)

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

pred_tr = lin_reg.predict(X_train)
pred_test = lin_reg.predict(X_test)
rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

print('RMSE of Train Data : ', rmse_tr)
print('RMSE of Test Data : ', rmse_test)
```

```
RMSE of Train Data :  4.737443359066909
RMSE of Test Data :  4.698992548166578
```

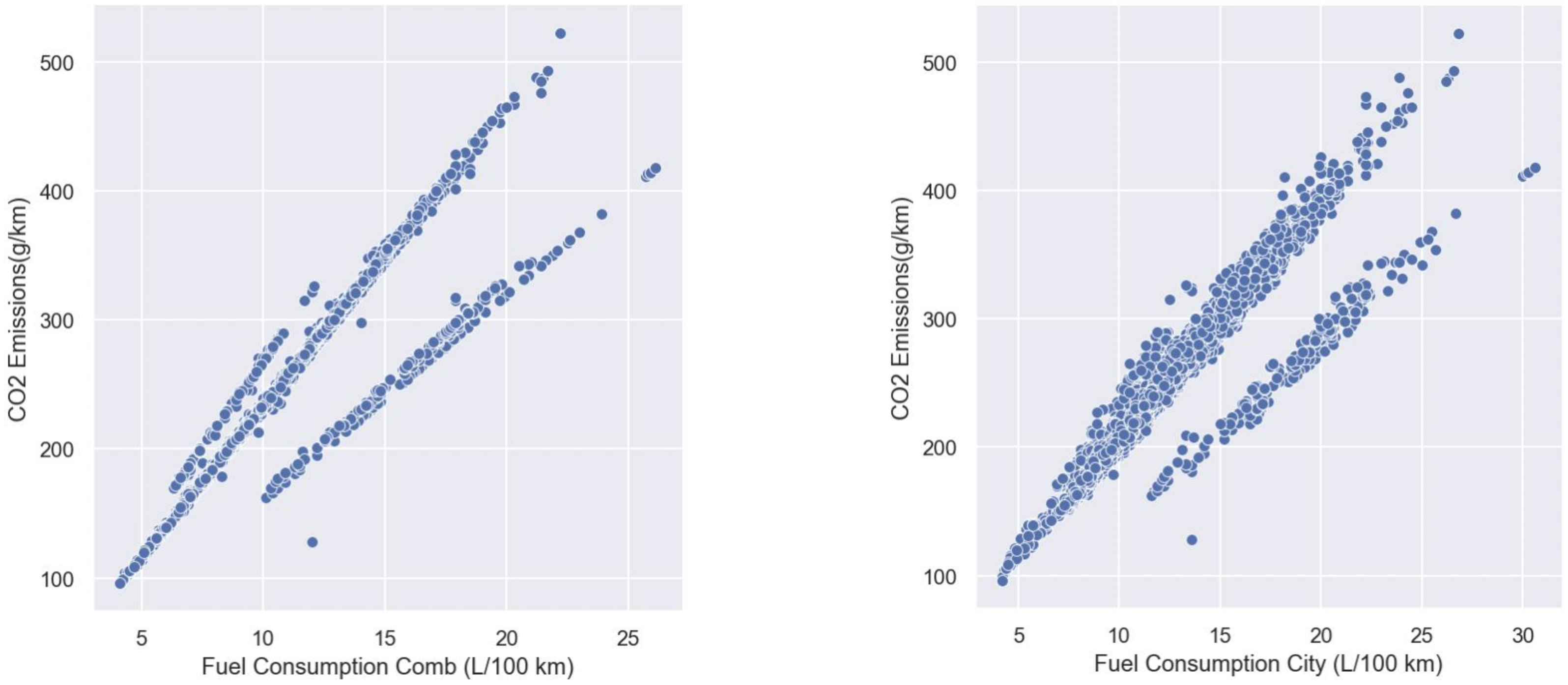
```
lin_reg.score(X_test,y_test)
```

```
0.9931729724310766
```

```
lin_reg.score(X_train,y_train)
```

```
0.993532796370605
```

다시 데이터 분석



일부 feature(‘**Fuel Consumption**’)이 label과 종속관계

LinearRegression 재시도

```
X_train, X_test, y_train, y_test = train_test_split(final_data2, labels, test_size=0.2, random_state=15)

lin_reg2 = LinearRegression()
lin_reg2.fit(X_train, y_train)

pred_tr = lin_reg2.predict(X_train)
pred_test = lin_reg2.predict(X_test)
rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

print('RMSE of Train Data : ', rmse_tr)
print('RMSE of Test Data : ', rmse_test)
```

```
RMSE of Train Data :  22.198630057191483
RMSE of Test Data :  22.857737632652853
```

```
lin_reg2.score(X_test,y_test)
```

```
0.8492322270770211
```

```
lin_reg2.score(X_train,y_train)
```

```
0.8555217498939712
```

OLS

OLS Regression Results

Dep. Variable:	CO2 Emissions(g/km)	R-squared:	0.855
Model:	OLS	Adj. R-squared:	0.854
Skew:	0.262	Prob(JB):	1.26e-264
Kurtosis:	4.917	Cond. No.	317.

Trans_AS8	30.2369	1.501	20.139	0.000	27.294
Trans_AS9	18.3839	2.908	6.321	0.000	12.683
Trans_AV10	-19.2377	6.889	-2.793	0.005	-32.741
Trans_AV6	-7.2297	2.461	-2.938	0.003	-12.053
Trans_AV7	9.7917	2.436	4.019	0.000	5.016
Trans_AV8	19.6999	3.822	5.154	0.000	12.207
Trans_M5	30.3822	2.055	14.787	0.000	26.354
Trans_M6	36.8068	1.485	24.782	0.000	33.895
Trans_M7	30.2628	3.023	10.011	0.000	24.337
Fl-T_E	-15.8458	1.297	-12.222	0.000	-18.387

변속기와 실린더로 co2 배출량 추정

1. 변속기 데이터 설명
2. 기어 수 추출 함수
3. 가장 많은 기어 수

1

A = Automatic
AM = Automated manual
AS = Automatic with select shift
AV = Continuously variable
M = Manual
3 - 10 = Number of gears

2

```
data['Transmission'].unique()
```

```
array(['AS5', 'M6', 'AV7', 'AS6', 'AM6', 'A6', 'AM7', 'AV8', 'AS8
       'A8', 'M7', 'A4', 'M5', 'AV', 'A5', 'AS7', 'A9', 'AS9', 'A
       'AS4', 'AM5', 'AM8', 'AM9', 'AS10', 'A10', 'AV10'], dtype=
```

```
def GetTransNum(s):
    try:
        int(s[-2:])
        return s[-2:]
    except ValueError:
        try:
            int(s[-1])
            return s[-1]
        except ValueError:
            return '6'
```

3

	Trans_val	Trans_type
count	7385	7385
unique	7	5
top	6	AS
freq	3259	3127

Trans-cylinder DataFrame

```
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Trans_val  7385 non-null   int64
 1   AS         7385 non-null   float64
 2   M          7385 non-null   float64
 3   AV         7385 non-null   float64
 4   AM         7385 non-null   float64
 5   A          7385 non-null   float64
 6   Cylinders  7385 non-null   int64
dtypes: float64(5), int64(2)
memory usage: 404.0 KB
```

LinearRegression

```
X_train, X_test, y_train, y_test = train_test_split(data_trans_final, labels, test_size=0.2, random_state=15)

lin_reg_trans2 = LinearRegression()
lin_reg_trans2.fit(X_train, y_train)

pred_tr = lin_reg_trans2.predict(X_train)
pred_test = lin_reg_trans2.predict(X_test)
rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

print('RMSE of Train Data : ', rmse_tr)
print('RMSE of Test Data : ', rmse_test)
```

```
RMSE of Train Data :  30.531334760826166
RMSE of Test Data :  30.19848049713175
```

```
lin_reg_trans2.score(X_train, y_train)
```

```
0.7266986935706093
```

```
lin_reg_trans2.score(X_test, y_test)
```

```
0.7368446432924975
```

LinearRegression

```
params = [{'reg':[LinearRegression()]},
          {'reg':[DecisionTreeRegressor()],
           'reg__max_depth': [4,6,8,10]
          },
          {'reg':[RandomForestRegressor()],
           'reg__n_estimators': [10, 100, 1000],
           'reg__max_depth' : [4, 6, 8, 10],
           'reg__min_samples_split': [2,3,4,5]
          }]

grid_pipeline = GridSearchCV(pipe, params, cv=5)
```

```
best_model.best_score_
```

```
0.7650266007806046
```

```
{'reg': RandomForestRegressor(max_depth=10, min_samples_split=4),
 'reg__max_depth': 10,
 'reg__min_samples_split': 4,
 'reg__n_estimators': 100}
```

감사합니다