SC1015
Mini Project

FDAD | Group 4

Davin See U2321193A

Cleon Chua U2322435L

Goh Yi Ting U2321172C

Content

GREEN PLAN

The Problem

34%

of Singaporeans

own at least 1 car

Every year, Singapore’s vehicles release about 6.4 million tonnes of carbon emissions, which makes up about 15% of the city-state’s total emissions

Goal

To find out which variables from our dataset contribute more significantly to the CO2 emissions of a vehicle

Predict the most significant variable using our chosen best regression model

motorist

Oct 2023 1st COE Bidding Results

CAT A

CAR UP TO 1600CC & 97KW

$104,000

▼ $1,000

CAT B

CAR ABOVE 1600CC OR 97KW

$146,002

▲ $5,113

CAT C

GOODS VEHICLE & BUS

$85,900

▲ $2,099

CAT D

MOTORCYCLE

$10,856

▲ $156

CAT E

OPEN-ALL EXCEPT MOTORCYCLE

$152,000

▲ $7,360

CHOICE OF DATASET

Size of dataset: 7385

```
In [7]:
1 cd_data = pd.read_csv("CO2Emissions_Canada.csv")
2 cd_data.head(20)
```

Out[7]:

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
0	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	33	196
1	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	29	221
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	48	136
3	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	25	255
4	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	27	244
5	ACURA	RLX	MID-SIZE	3.5	6	AS6	Z	11.9	7.7	10.0	28	230
6	ACURA	TL	MID-SIZE	3.5	6	AS6	Z	11.8	8.1	10.1	28	232
7	ACURA	TL AWD	MID-SIZE	3.7	6	AS6	Z	12.8	9.0	11.1	25	255
8	ACURA	TL AWD	MID-SIZE	3.7	6	M6	Z	13.4	9.5	11.6	24	267
9	ACURA	TSX	COMPACT	2.4	4	AS5	Z	10.6	7.5	9.2	31	212
10	ACURA	TSX	COMPACT	2.4	4	M6	Z	11.2	8.1	9.8	29	225

CO2 Emission by Vehicles

Make

The company that makes the car.

Audi, Ford, BMW etc.

Model

4WD/4X4 = Four-wheel drive
AWD = All-wheel drive
FFV = Flexible-fuel vehicle
SWB = Short wheelbase
LWB = Long wheelbase
EWB = Extended wheelbase

Transmission

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

Fuel type

X = Regular gasoline
Z = Premium gasoline
D = Diesel
E = Ethanol (E85)
N = Natural gas

Fuel Consumption

City and highway fuel consumption ratings are shown in litres per 100 kilometres (L/100 km) - the combined rating (55% city, 45% hwy) is shown in L/100 km and in miles per gallon (mpg)

CO2 Emissions

The tailpipe emissions of carbon dioxide (in grams per kilometre) for combined city and highway driving

Data Cleaning

Changed the fuel type to the actual names of the fuel type

Fuel Type
Z
Z
Z
Z
Z
Z
Z
Z
Z
Z
Z
Z

Fuel Type
Premium Gasoline
Premium Gasoline
Premium Gasoline
Premium Gasoline
Premium Gasoline
...
Premium Gasoline
Premium Gasoline

Data Cleaning

Separated the original Transmission variable into 2 columns to analyse them separately

Transmission
AS5
M6
AV7
AS6
AS6
AS6
AS6
AS6
M6
AS5
M6

Gears	Transmission Type
5	AS
6	M
7	AV
6	AS
6	AS
...	...
8	AS
8	AS
8	AS
8	AS
8	AS

Gears

CVT or continuously variable transmission cars don't have multiple gears. They essentially have one gear with infinite variability.

```
cd_data["Gears"].fillna(value=1, inplace=True)
```

```
print(cd_data["Gears"].dropna().value_counts())
```

```
Gears
6     3259
8     1802
7     1026
9      419
5      307
1      295
10     210
4       67
Name: count, dtype: int64
```

Predictor Variables to be used

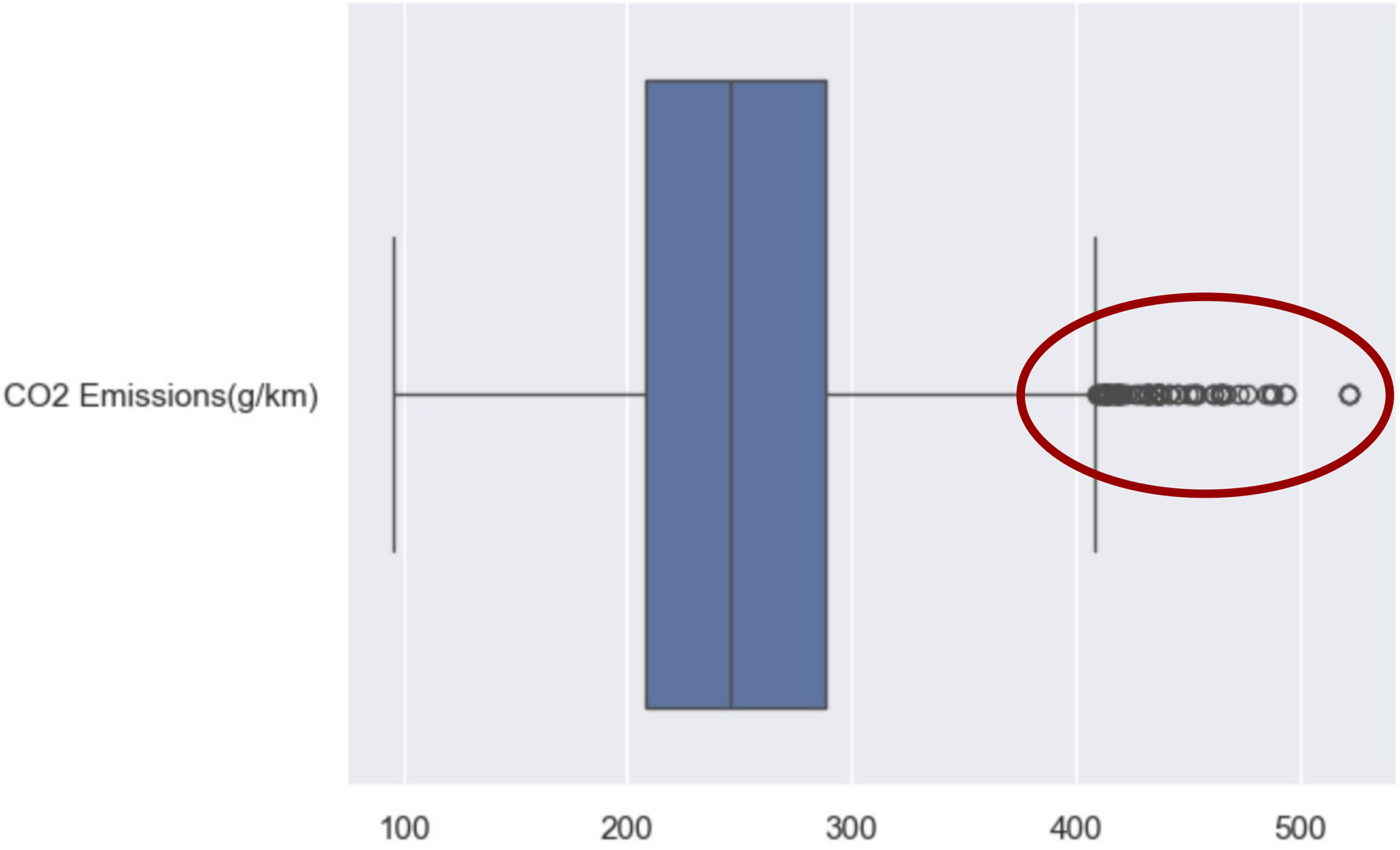
Numerical

- Engine Size (L)
- Fuel Consumption Comb (L/100km)

Categorical

- Vehicle class
- Cylinders
- Fuel Type
- Gears
- Transmission Type

Finding Outliers



Finding Outliers

```
Q1 = cd_data['CO2 Emissions(g/km)'].quantile(0.25)
Q3 = cd_data['CO2 Emissions(g/km)'].quantile(0.75)
IQR = Q3 - Q1
Q1, Q3, IQR
```

```
(208.0, 288.0, 80.0)
```

```
upper_limit = Q3 + (1.5 * IQR)
lower_limit = Q1 - (1.5 * IQR)
lower_limit, upper_limit
```

```
(88.0, 408.0)
```

```
# find the outliers
outliers = cd_data.loc[(cd_data['CO2 Emissions(g/km)'] > upper_limit) | (cd_data['CO2 Emissions(g/km)'] < lower_limit)]
```

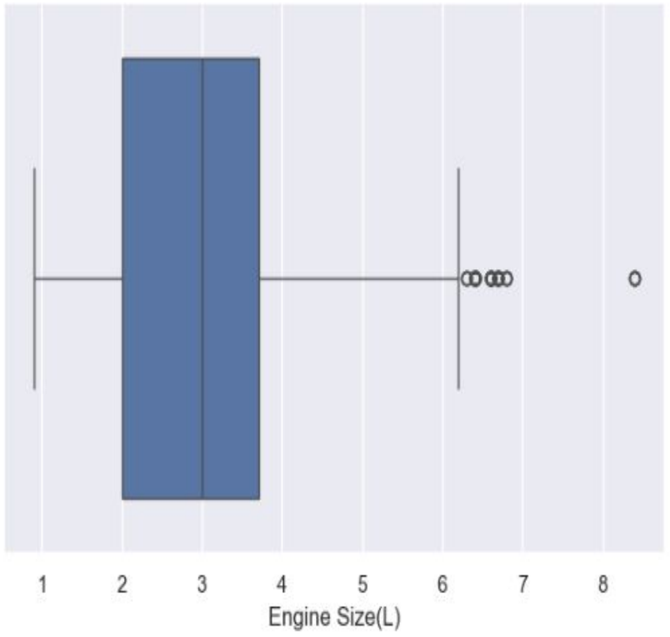
Number of cars for each vehicle class

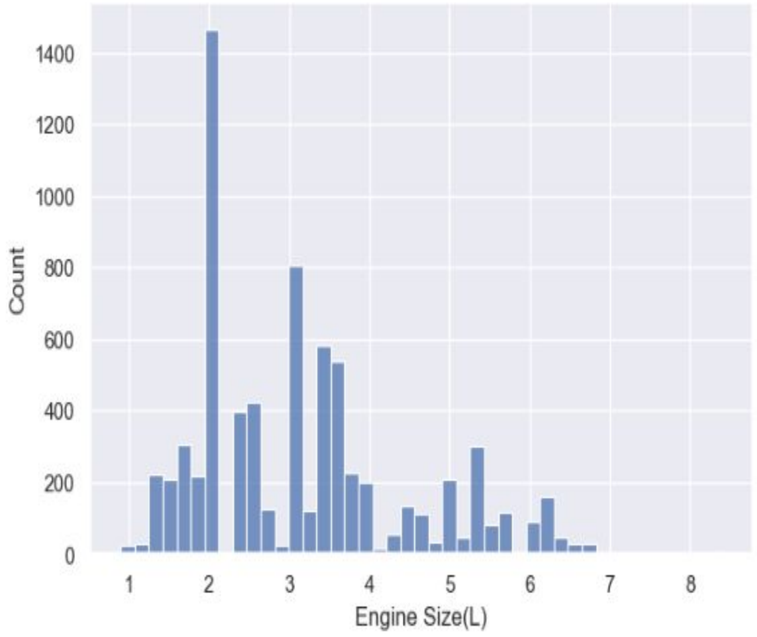
```
Vehicle Class
SUV - SMALL                     1217
MID-SIZE                        1133
COMPACT                         1022
SUV - STANDARD                   735
FULL-SIZE                        639
SUBCOMPACT                       606
PICKUP TRUCK - STANDARD          538
TWO-SEATER                       460
MINICOMPACT                      326
STATION WAGON - SMALL            252
PICKUP TRUCK - SMALL             159
MINIVAN                           80
SPECIAL PURPOSE VEHICLE           77
VAN - PASSENGER                   66
STATION WAGON - MID-SIZE          53
VAN - CARGO                       22
Name: count, dtype: int64
```

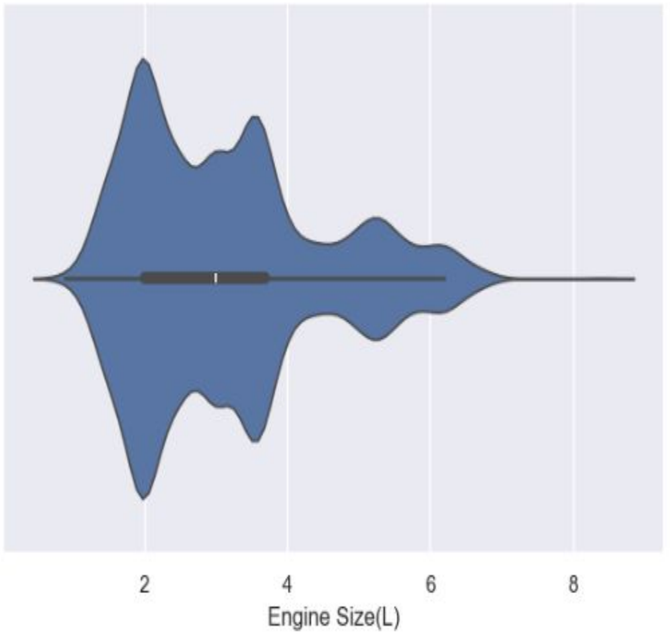
Outliers

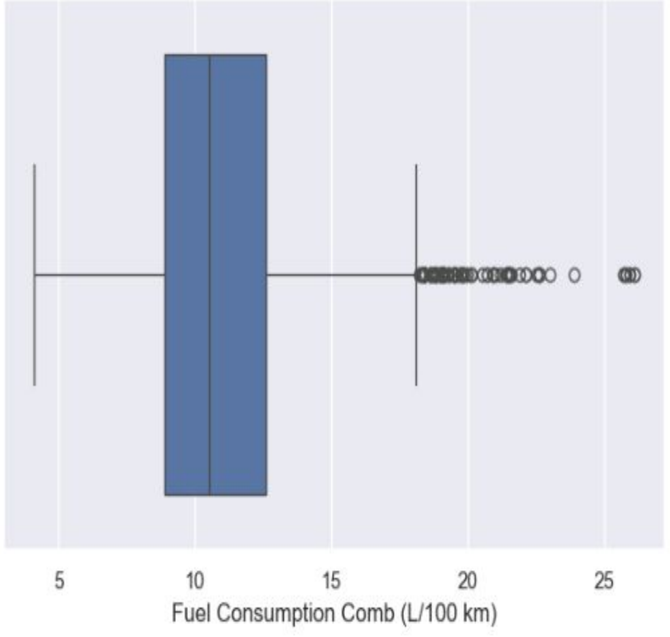
```
Vehicle Class
VAN - PASSENGER                 38
TWO-SEATER                      24
SUV - STANDARD                  13
MID-SIZE                         4
PICKUP TRUCK - STANDARD          1
Name: count, dtype: int64
```

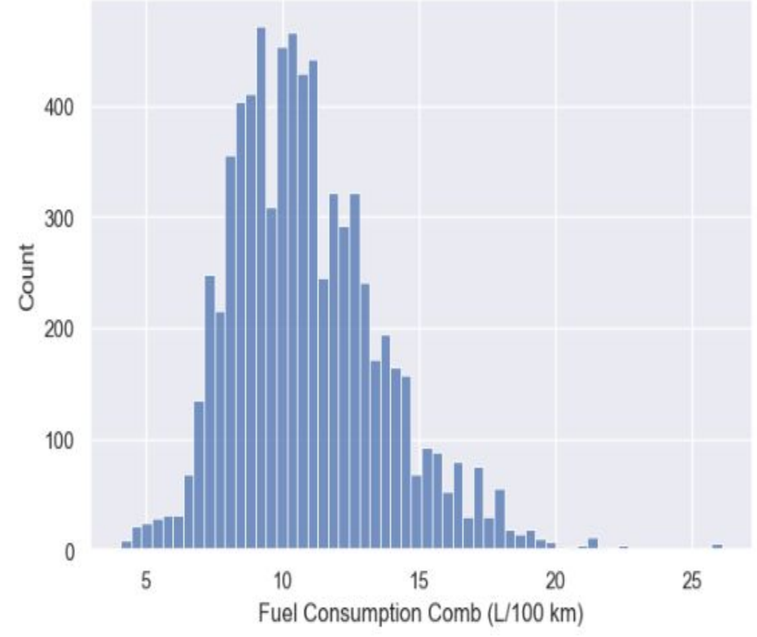
Data Exploration:
Uni-variate exploration on numeric variables

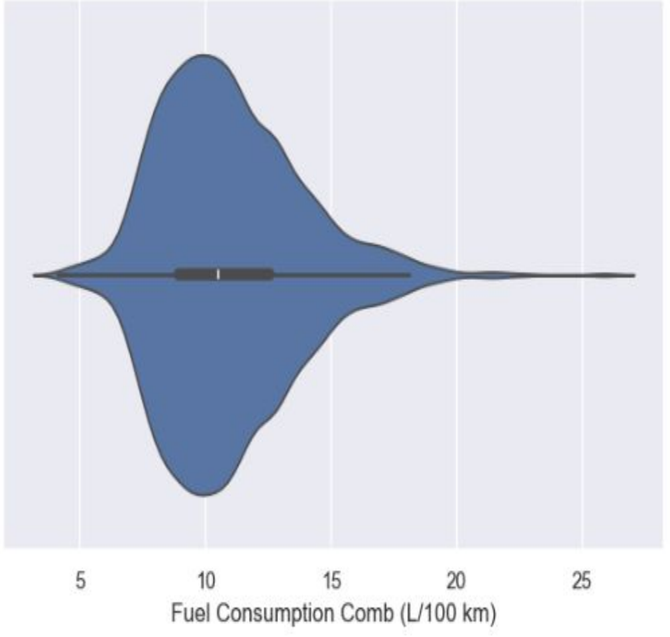

1
2
3
4
5
6
7
8
Engine Size(L)


1400
1200
1000
800
600
400
200
0
Count
1
2
3
4
5
6
7
8
Engine Size(L)


2
4
6
8
Engine Size(L)


5
10
15
20
25
Fuel Consumption Comb (L/100 km)


400
300
200
100
0
Count
5
10
15
20
25
Fuel Consumption Comb (L/100 km)


5
10
15
20
25
Fuel Consumption Comb (L/100 km)

Accounting for skew of each numeric variable:

```
print("The skew for Engine Size: ", cd_data["Engine Size(L)"].skew())
print("The skew for Fuel Consumption Comb: ", cd_data["Fuel Consumption Comb (L/100 km)"].skew())
```

4] Python

```
The skew for Engine Size:  0.802075961336007
The skew for Fuel Consumption Comb:  0.8520547445765442
```

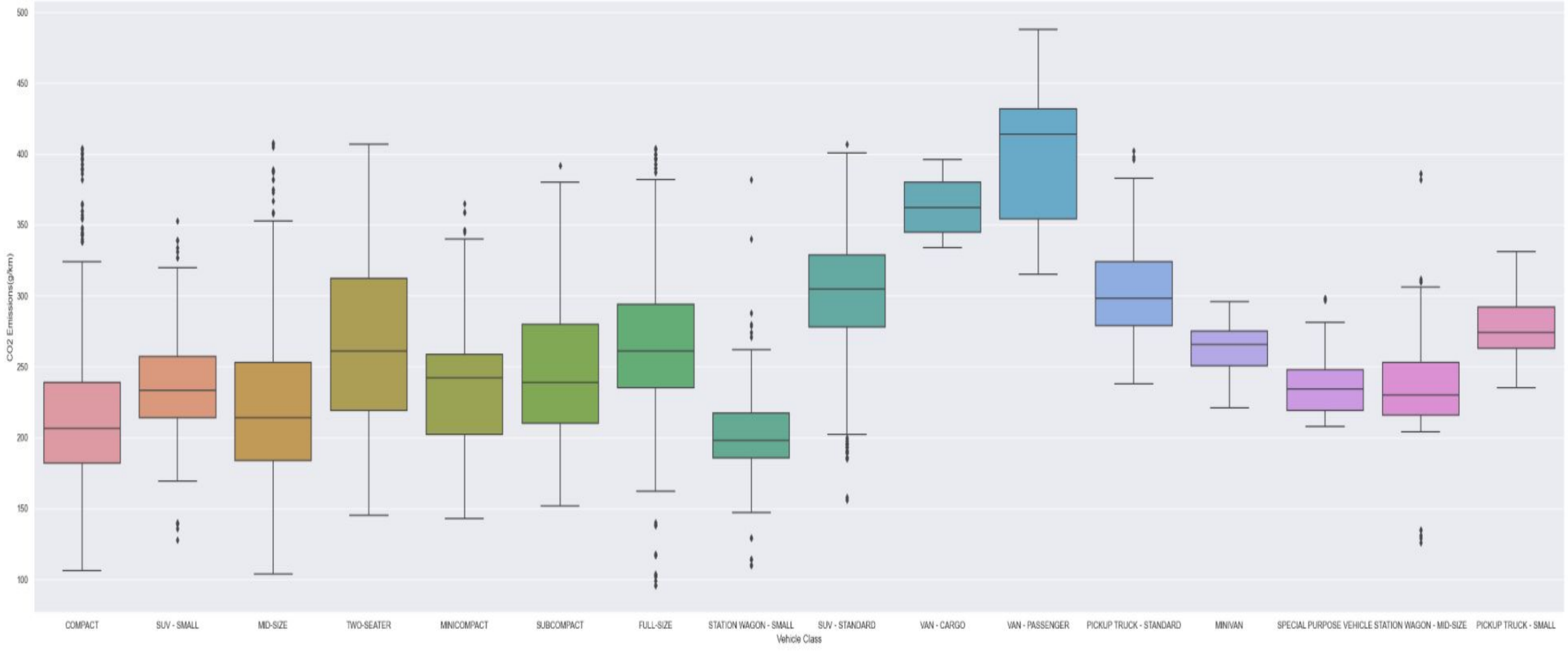
```
print(f'Engine Size (L) kurtosis = {cd_data["Engine Size(L)"].kurt()}')
print(f'Fuel Consumption Comb (L/100km) kurtosis = {cd_data["Fuel Consumption Comb (L/100 km)"].kurt()}')
```

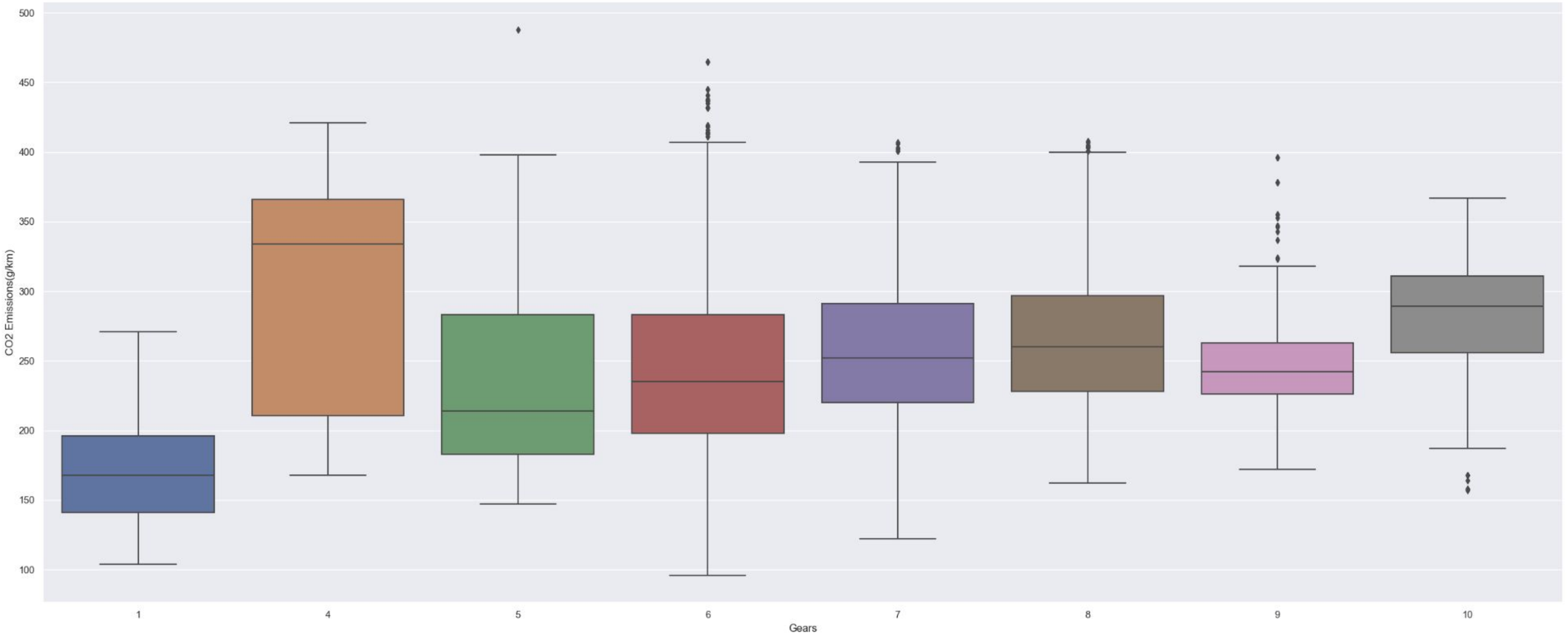
5] Python

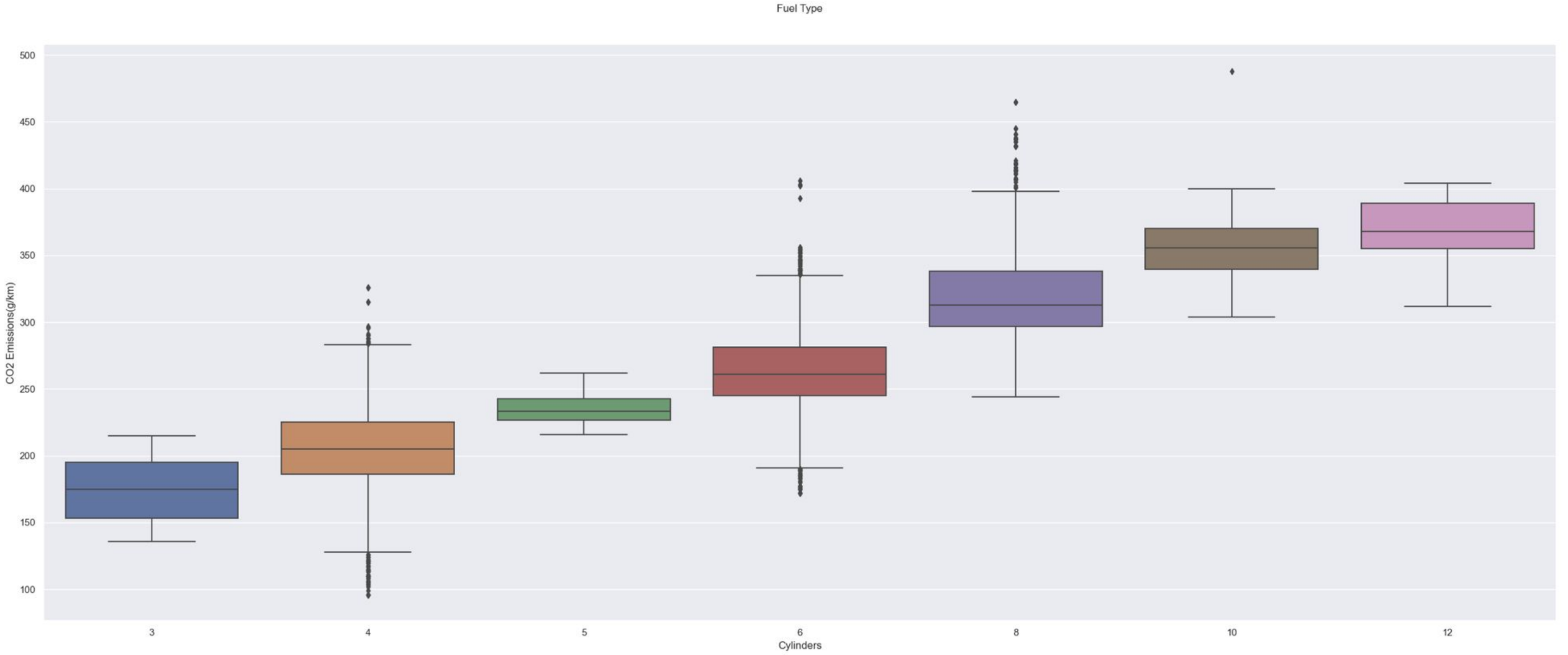
```
Engine Size (L) kurtosis = -0.1438836043778
Fuel Consumption Comb (L/100km) kurtosis = 1.3729090137574715
```

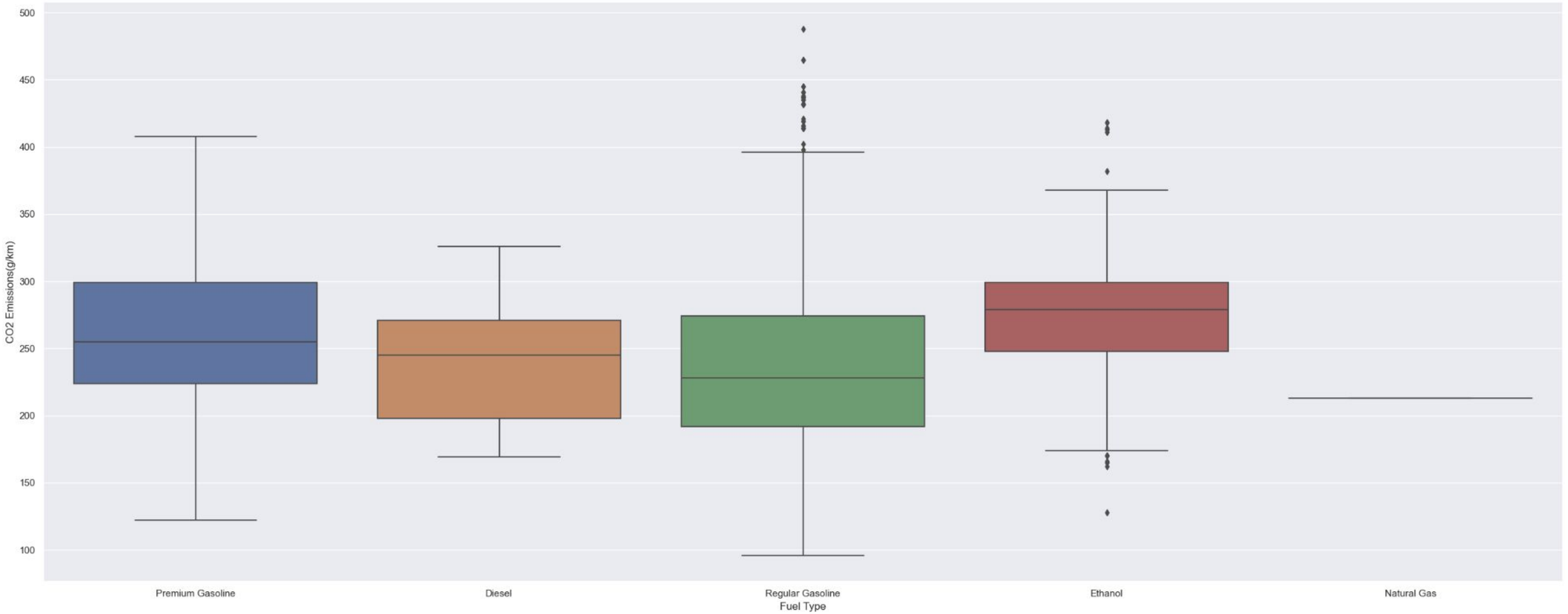
Since the kurtosis of Engine Size is < 0, Engine Size (L) is said to be platykurtic while Fuel Consumption Comb (L/100km) is said to be leptokurtic since its kurtosis is > 0. The magnitude of the kurtosis of both variables are deemed to be acceptable and close to a normal value (which is 0). Similarly both variables are positively skewed, but since their values are less than 1, the magnitude of their skewness are also deemed to be acceptable. Next, we will do a univariate exploration of the categorical variables (Vehicle Class, Cylinders, Fuel Type, Gears, Transmission Type)

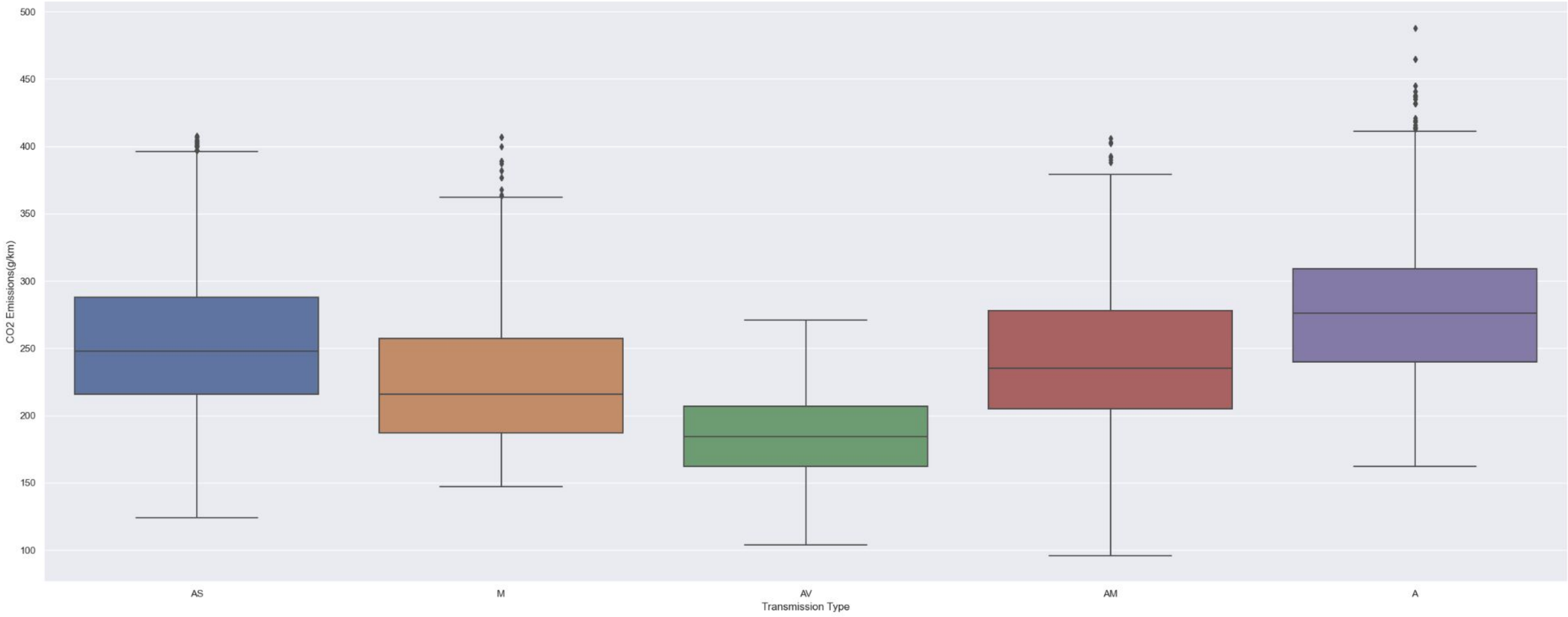
Data Exploration:
Bi-variate exploration on categorical variables


CO2 Emissions(g/km)
500
450
400
350
300
250
200
150
100
COMPACT
SUV - SMALL
MID-SIZE
TWO-SEATER
MINICOMPACT
SUBCOMPACT
FULL-SIZE
STATION WAGON - SMALL
SUV - STANDARD
VAN - CARGO
VAN - PASSENGER
PICKUP TRUCK - STANDARD
MINIVAN
SPECIAL PURPOSE VEHICLE
STATION WAGON - MID-SIZE
PICKUP TRUCK - SMALL
Vehicle Class


500
450
400
350
300
250
200
150
100
CO2 Emissions(g/km)
1
4
5
6
7
8
9
10
Gears


Fuel Type
CO2 Emissions(g/km)
500
450
400
350
300
250
200
150
100
3
4
5
6
8
10
12
Cylinders


500
450
400
350
300
250
200
150
100
CO2 Emissions(g/km)
Premium Gasoline
Diesel
Regular Gasoline
Ethanol
Natural Gas
Fuel Type


500
450
400
350
300
250
200
150
100
CO2 Emissions(g/km)
AS
M
AV
AM
A
Transmission Type

One-hot Encoding

VClass_COMPACT	...	VClass_SUV - SMALL	VClass_SUV - STANDARD	VClass_TWO-SEATER	VClass_VAN - CARGO	VClass_VAN - PASSENGER
1	...	0	0	0	0	0
1	...	0	0	0	0	0
1	...	0	0	0	0	0
0	...	1	0	0	0	0
0	...	1	0	0	0	0
0	...	0	0	0	0	0
0	...	0	0	0	0	0
0	...	0	0	0	0	0
0	...	0	0	0	0	0
1	...	0	0	0	0	0
1	...	0	0	0	0	0
1	...	0	0	0	0	0
0	...	0	0	1	0	0

FT_Diesel	FT_Ethanol	FT_Natural Gas	FT_Premium Gasoline	FT_Regular Gasoline
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0
0	0	0	1	0

Investigating the multicollinearity of the 5 categorical variables

In [43]:
```python
from scipy.stats import chi2_contingency
import itertools

def generate_pairs(arr):
    return list(itertools.combinations(arr, 2))

def apply_function_to_pairs(arr):
    pairs = generate_pairs(arr)

    for x, y in pairs:
        chi2, p_value, dof, ex = chi2_contingency(pd.crosstab(cd_data[x], cd_data[y]))
        print(f"p value for chi square test for {x} and {y} is: {p_value}")
    return

apply_function_to_pairs(["Cylinders", "Fuel Type", "Transmission Type", "Vehicle Class", "Gears"])
```

```
p value for chi square test for Cylinders and Fuel Type is: 1.3300421409111848e-192
p value for chi square test for Cylinders and Transmission Type is: 5.789382830017478e-262
p value for chi square test for Cylinders and Vehicle Class is: 0.0
p value for chi square test for Cylinders and Gears is: 2.9225385121989606e-254
p value for chi square test for Fuel Type and Transmission Type is: 1.0137188060338775e-249
p value for chi square test for Fuel Type and Vehicle Class is: 0.0
p value for chi square test for Fuel Type and Gears is: 0.0
p value for chi square test for Transmission Type and Vehicle Class is: 0.0
p value for chi square test for Transmission Type and Gears is: 0.0
p value for chi square test for Vehicle Class and Gears is: 0.0
```

Our null hypothesis is that each categorical variable is independent from every other categorical variable. For all (5C2 == 10) cases, our chi squared test returned a p-value that is very close to 0, or 0. Using a 95% confidence test, we found that p-value for all cases was much smaller than the alpha level of 0.05, so we do not reject the null hypothesis. We can say that all the categorical variables are independent from each other- there is no multicollinearity.

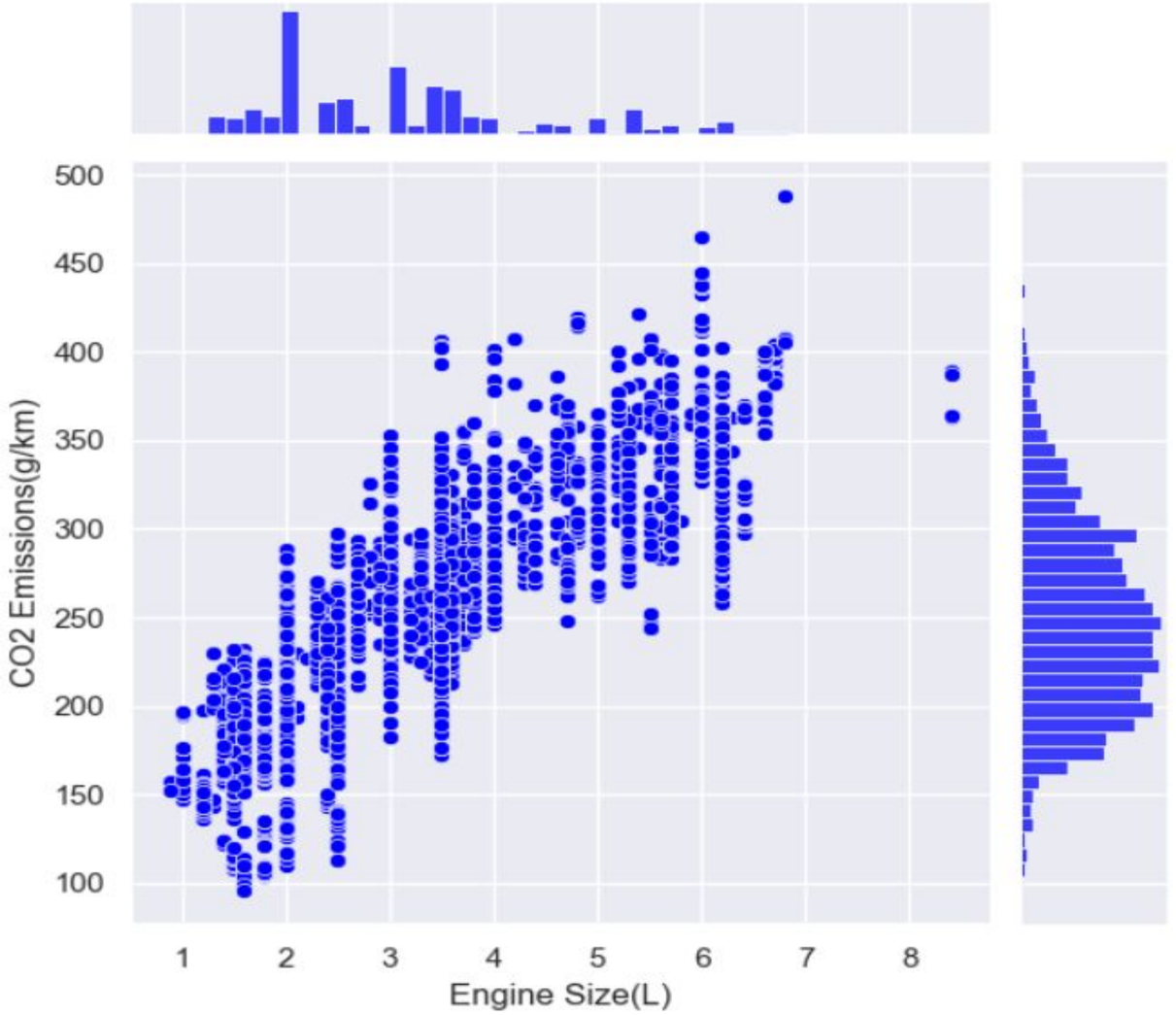
Data Exploration:

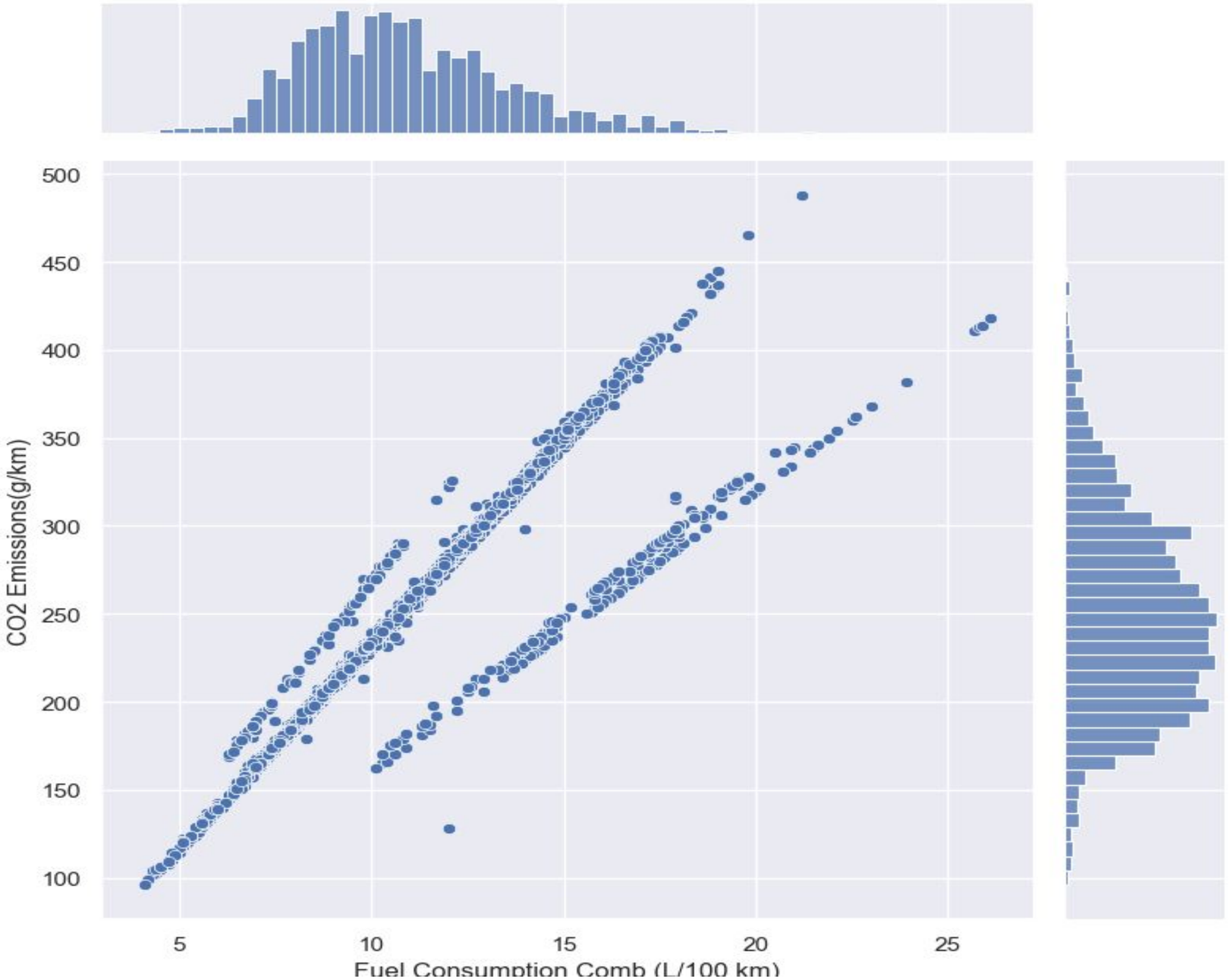
Bivariate analysis of numeric variables

Engine Size vs CO2 Emissions

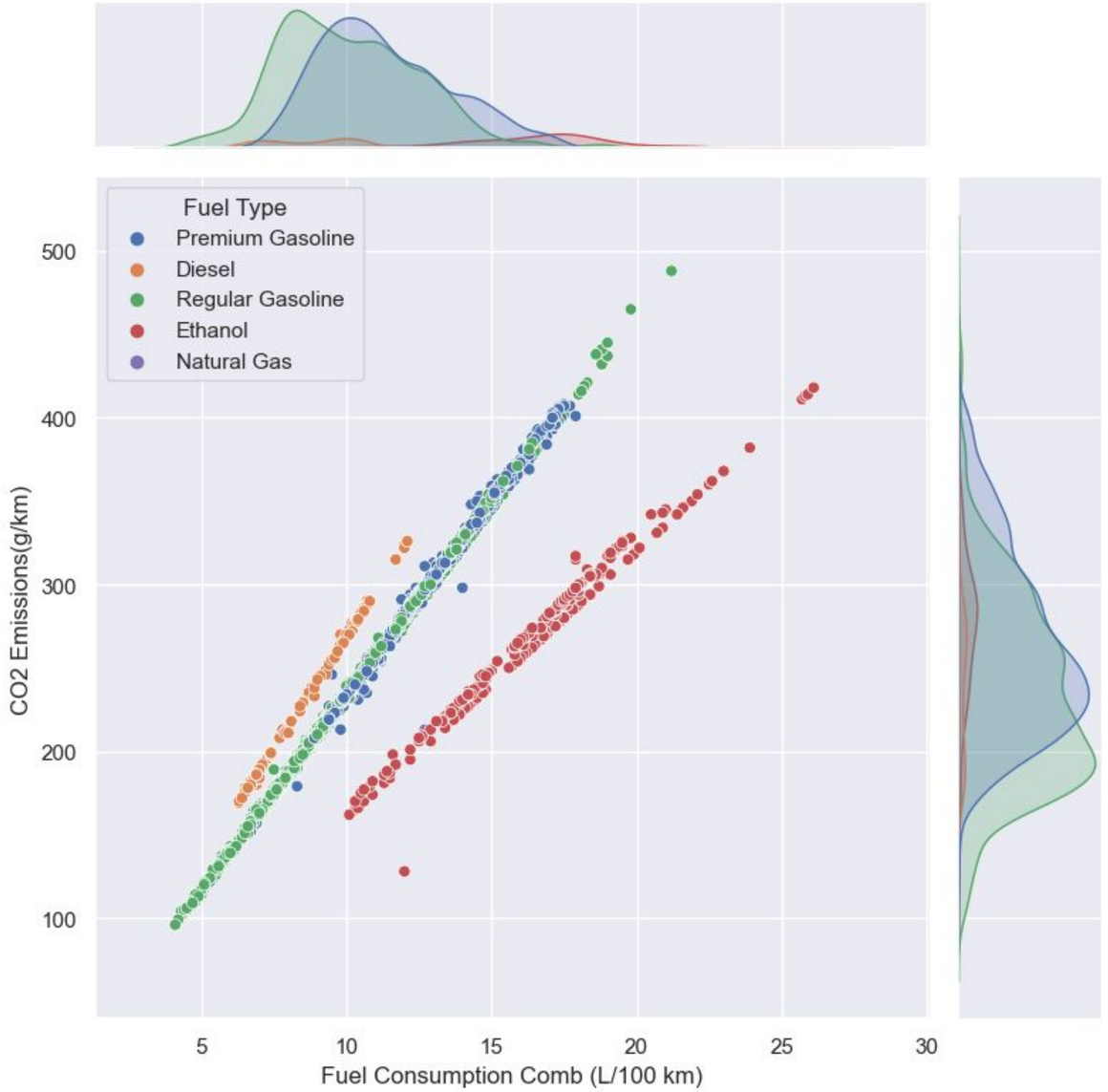
In [37]:
```
sb.jointplot(data = cd_data, x = "Engine Size(L)", y = "CO2 Emissions(g/km)", height = 6, color = "blue")
```

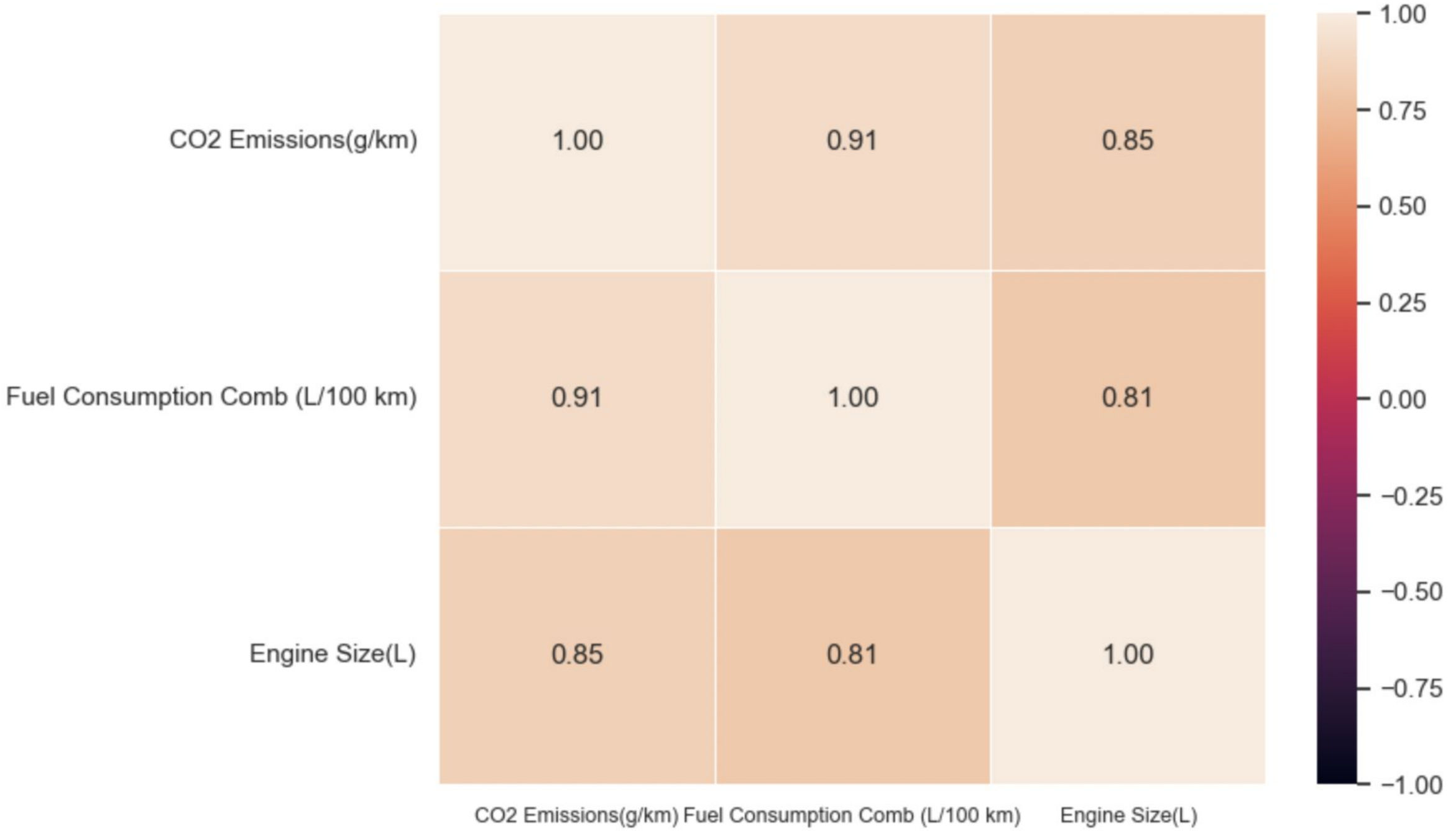
Out[37]: <seaborn.axisgrid.JointGrid at 0x2a1efcc90>




500
450
400
350
300
250
200
150
100
CO2 Emissions(g/km)
5
10
15
20
25
Fuel Consumption Comb (L/100 km)

```
sb.jointplot(data = cd_data, x = "Fuel Consumption Comb (L/100 km)", y = "CO2 Emissions(g/km)", height = 8, hue = "Fuel Type
```




CO2 Emissions(g/km)
Fuel Consumption Comb (L/100 km)
Engine Size(L)
1.00
0.91
0.85
0.91
1.00
0.81
0.85
0.81
1.00
CO2 Emissions(g/km)
Fuel Consumption Comb (L/100 km)
Engine Size(L)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Model 1:
Multi Linear Regression

Model 2:
Ridge Regression

Model 3:
Lasso Regression

Model 1:
Multi Linear Regression

```
1 linreg = LinearRegression()
2
3 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20, random_state = 69)
4 print("Train Set :", y_train.shape, X_train.shape)
5 print("Test Set  :", y_test.shape, X_test.shape)
```

```
Train Set : (5874, 1) (5874, 30)
Test Set  : (1469, 1) (1469, 30)
```

```
1 X_train.describe()
```

	Engine Size(L)	Cylinders	Fuel Consumption Comb (L/100 km)	Gears	FT_Diesel	FT_Ethanol	FT_Natural Gas	FT_Premium Gasoline	FT_Regular Gasoline	VClass_COMPACT	...	VClass - SI
count	5874.000000	5874.000000	5874.000000	5874.000000	5874.000000	5874.000000	5874.000000	5874.000000	5874.000000	5874.000000	...	5874.00
mean	3.155107	5.604528	10.931716	6.659517	0.024855	0.048689	0.000170	0.435478	0.490807	0.135853	...	0.16
std	1.351017	1.797908	2.832862	1.640987	0.155697	0.215236	0.013048	0.495862	0.499958	0.342661	...	0.37
min	0.900000	3.000000	4.100000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00
25%	2.000000	4.000000	8.900000	6.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00
50%	3.000000	6.000000	10.500000	6.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00
75%	3.700000	6.000000	12.600000	8.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	...	0.00
max	8.400000	12.000000	26.100000	10.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.00

8 rows × 30 columns

```
1 y_train.describe()
```

	CO2 Emissions(g/km)
count	5874.000000
mean	249.783793
std	57.015024
min	96.000000
25%	208.000000
50%	245.000000
75%	288.000000
max	465.000000

Model 1:
Multi Linear Regression

Model 1: Multilinear Regression Model

```
1 #we fit our train data onto our multilinear regression model
2 linreg.fit(X_train, y_train)
```

▾ LinearRegression

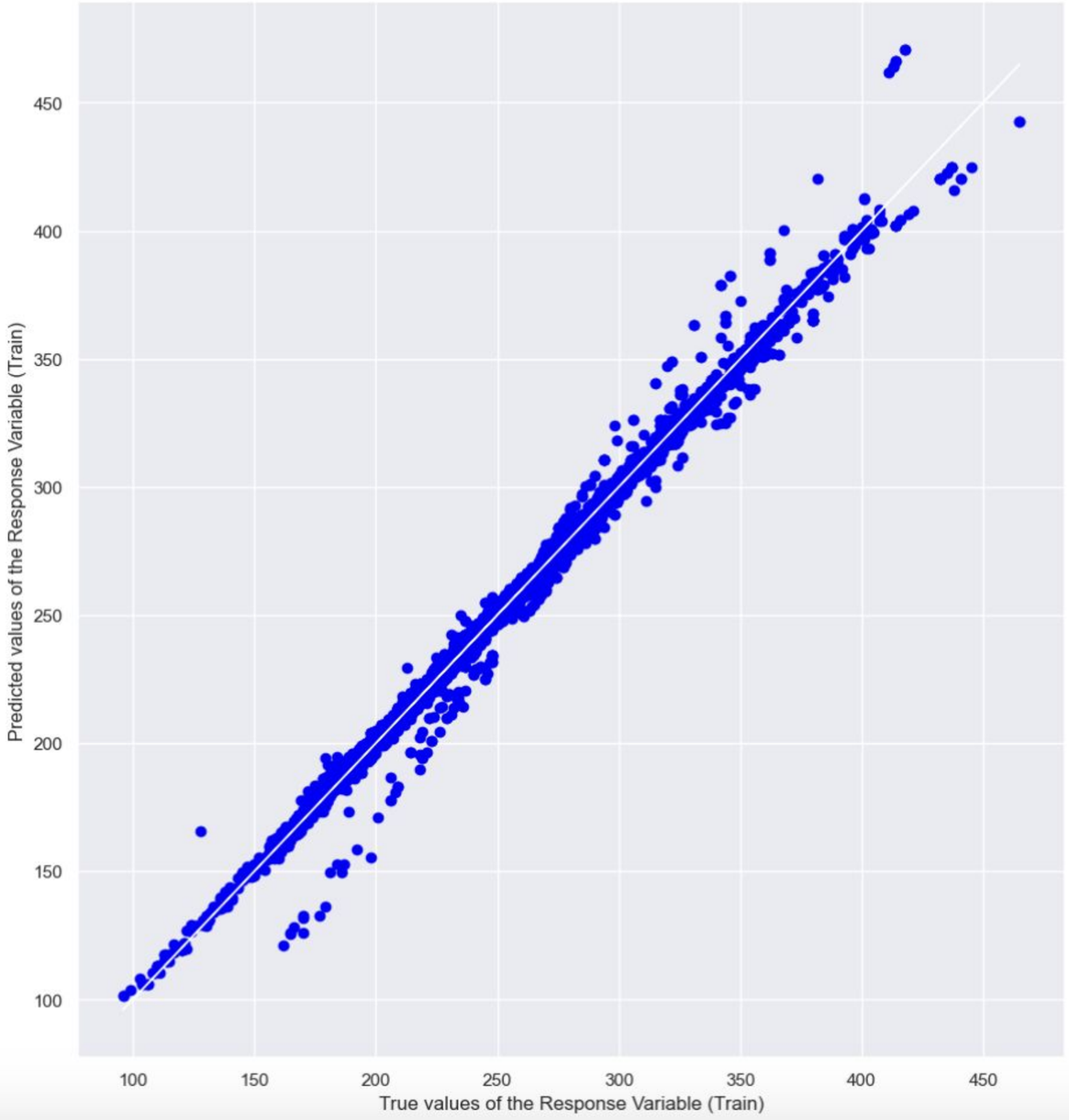
LinearRegression()

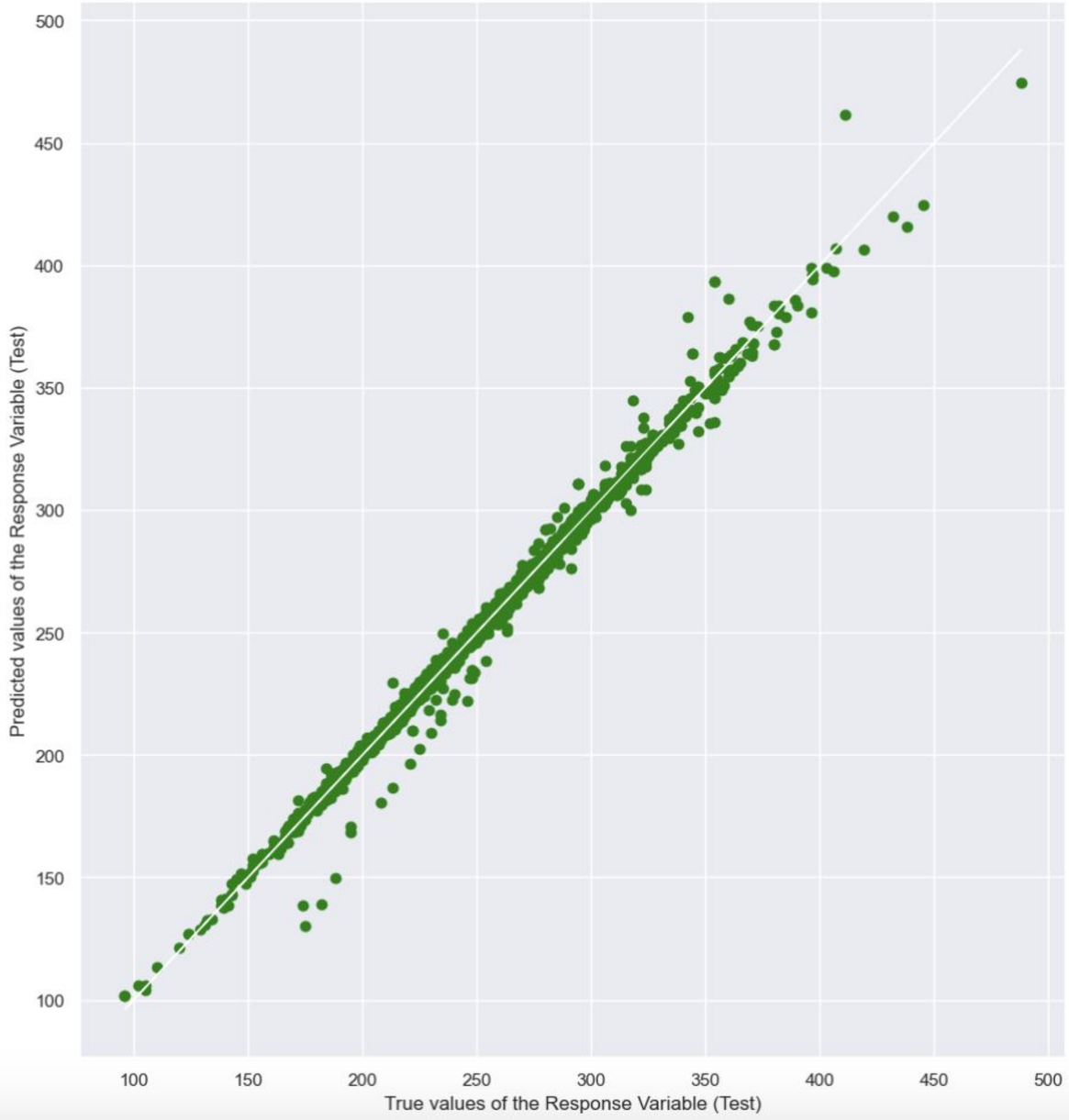
```
1 print('Intercept \t: b = ', linreg.intercept_)
2 print('Coefficients \t: a = ', linreg.coef_)
```

```
Intercept       : b =  [-8.79870338e+12]
Coefficients    : a =  [[ 1.71381927e-01  6.00511846e-01  2.23369924e+01  3.86860281e-01
   9.10939369e+12  9.10939369e+12  9.10939369e+12  9.10939369e+12
   9.10939369e+12  1.12085573e+11  1.12085573e+11  1.12085573e+11
   1.12085573e+11  1.12085573e+11  1.12085573e+11  1.12085573e+11
   1.12085573e+11  1.12085573e+11  1.12085573e+11  1.12085573e+11
   1.12085573e+11  1.12085573e+11  1.12085573e+11  1.12085573e+11
   1.12085573e+11 -4.22775884e+11 -4.22775884e+11 -4.22775884e+11
  -4.22775884e+11 -4.22775884e+11]]
```

```
1 y_train_pred = linreg.predict(X_train)
2 y_test_pred = linreg.predict(X_test)
```

Model 1:
Multi Linear Regression





Ridge and Lasso Regressions

Model Interpretability: Finding the most significant factors affecting CO_2 Emissions

High multicollinearity between our numeric variables, which both models can account for.

Model 2:
Ridge Regression

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=69)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (5874, 30)
The dimension of X_test is (1469, 30)
```

```
#We scale our data so that we remove any incorrect penalisation of our loss function. This allows for a more
#optimal model after regularisation is done.
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)
```

Model 2:
Ridge Regression

```
from sklearn.linear_model import Ridge
ridge = Ridge()
```

```
#we try to identify the alpha value that gives the best results
grid = {'alpha': [(0.00001 * 10**x) for x in range(1, 8)] }
from sklearn.model_selection import GridSearchCV
ridge_cv = GridSearchCV(ridge, grid, cv=3, n_jobs=1)
ridge_cv.fit(x_train, y_train)
y_pred = ridge_cv.predict(x_test)
```

Using GridSearchCV to test various alpha values from 0.0001 to 100.
Optimal alpha value = 0.1

Model 2:
Ridge Regression

```
1 ridge_cv.best_estimator_.intercept_
```

```
249.78379298604014
```

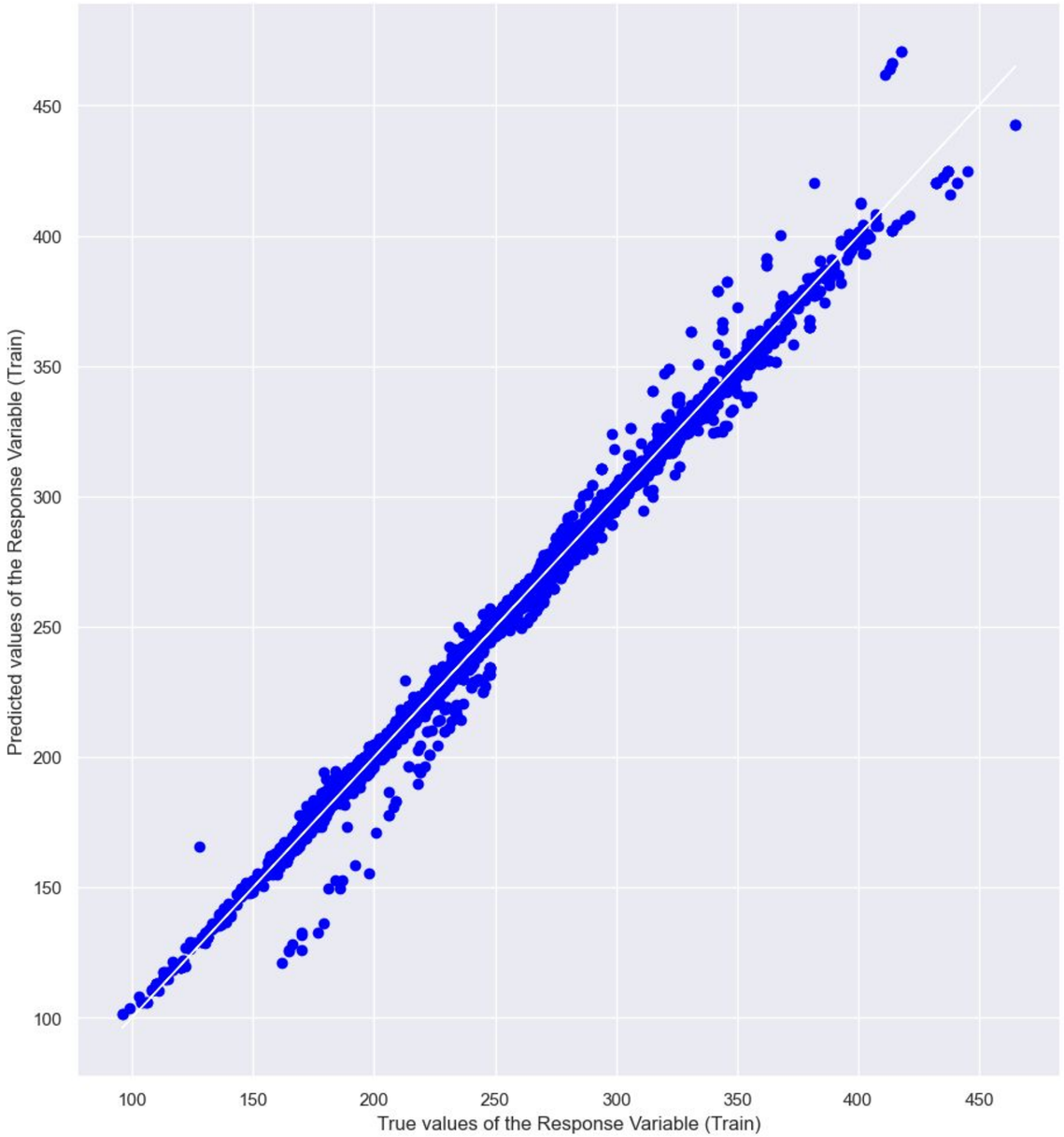
```
1 ridge_cv.best_estimator_.coef_
```

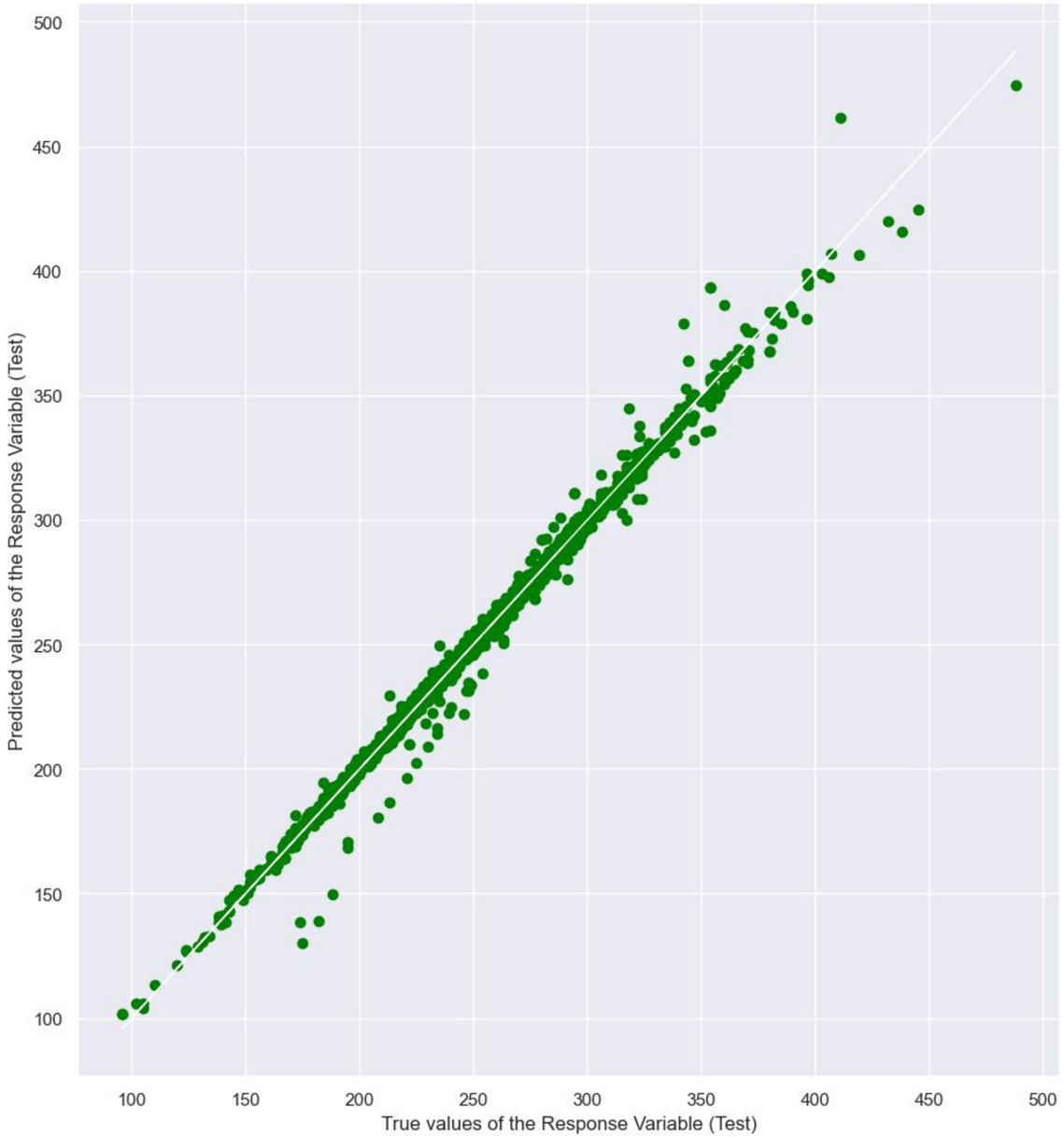
```
array([ 2.35012294e-01,  1.03789344e+00,  6.33529542e+01,  7.10935248e-01,
        5.77996642e+00, -2.25310602e+01, -9.50519088e-01,  3.90077538e+00,
        4.05578009e+00, -2.32552652e-01,  2.33151493e-01, -1.40474527e-01,
       -1.57775666e-01,  6.02204330e-02,  4.54719117e-01,  2.39050623e-01,
        4.37681329e-01, -8.65502478e-03, -1.48569133e-01, -7.62200757e-02,
        2.50634321e-01,  8.38092469e-02, -6.55521904e-03, -7.97161554e-01,
       -1.12973936e+00, -4.80597534e-03,  7.15979506e-02,  6.89947145e-02,
       -1.88141713e-01, -3.72582132e-03])
```

LEGEND: FT - Fuel Type, VClass - Vehicle Class, TType - Transmission Type

[Engine Size(L), Cylinders, Fuel Consumption Comb (L/100 km), Gears,
FT_Diesel, FT_Ethanol, FT_Natural Gas, FT_Premium Gasoline,
FT_Regular Gasoline, VClass_COMPACT, VClass_FULL-SIZE, VClass_MID-SIZE,
VClass_MINICOMPACT, VClass_MINIVAN, VClass_PICKUP TRUCK - SMALL, VClass_PICKUPTRUCK - STANDARD,
VClass_SPECIAL PURPOSE VEHICLE, VClass_STATION WAGON - MID-SIZE, VClass_STATION WAGON - SMALL, VClass_SUBCOMPACT,
VClass_SUV - SMALL, VClass_SUV - STANDARD, VClass_TWO-SEATER, VClass_VAN - CARGO,
VClass_VAN - PASSENGER, TType_A, TType_AM, TType_AS,
TType_AV, TType_M]

Model 2:
Ridge Regression





Model 3:
Lasso Regression

```
X = final_data
y = cd_data["CO2 Emissions(g/km)"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=69)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
```

```
he dimension of X_train is (5874, 30)
he dimension of X_test is (1469, 30)
```

```
#We scale our data so that we remove any incorrect penalisation of our loss function. This allows for a more
#optimal model after regularisation is done.
scale = StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

Model 3:
Lasso Regression

```
from sklearn.linear_model import Lasso
lasso = Lasso()
#we try to identify the alpha value that gives the best results
grid = {'alpha': [(0.00001 * 10**x) for x in range(1, 8)] }
lasso_cv = GridSearchCV(lasso, grid, cv=3, n_jobs=-1)
lasso_cv.fit(X_train, y_train)
```

Using GridSearchCV to test various alpha values from 0.0001 to 100.
Optimal alpha value = 0.001

Model 3:
Lasso Regression

```
1 lasso_cv.best_estimator_.intercept_
```

```
249.78379298604014
```

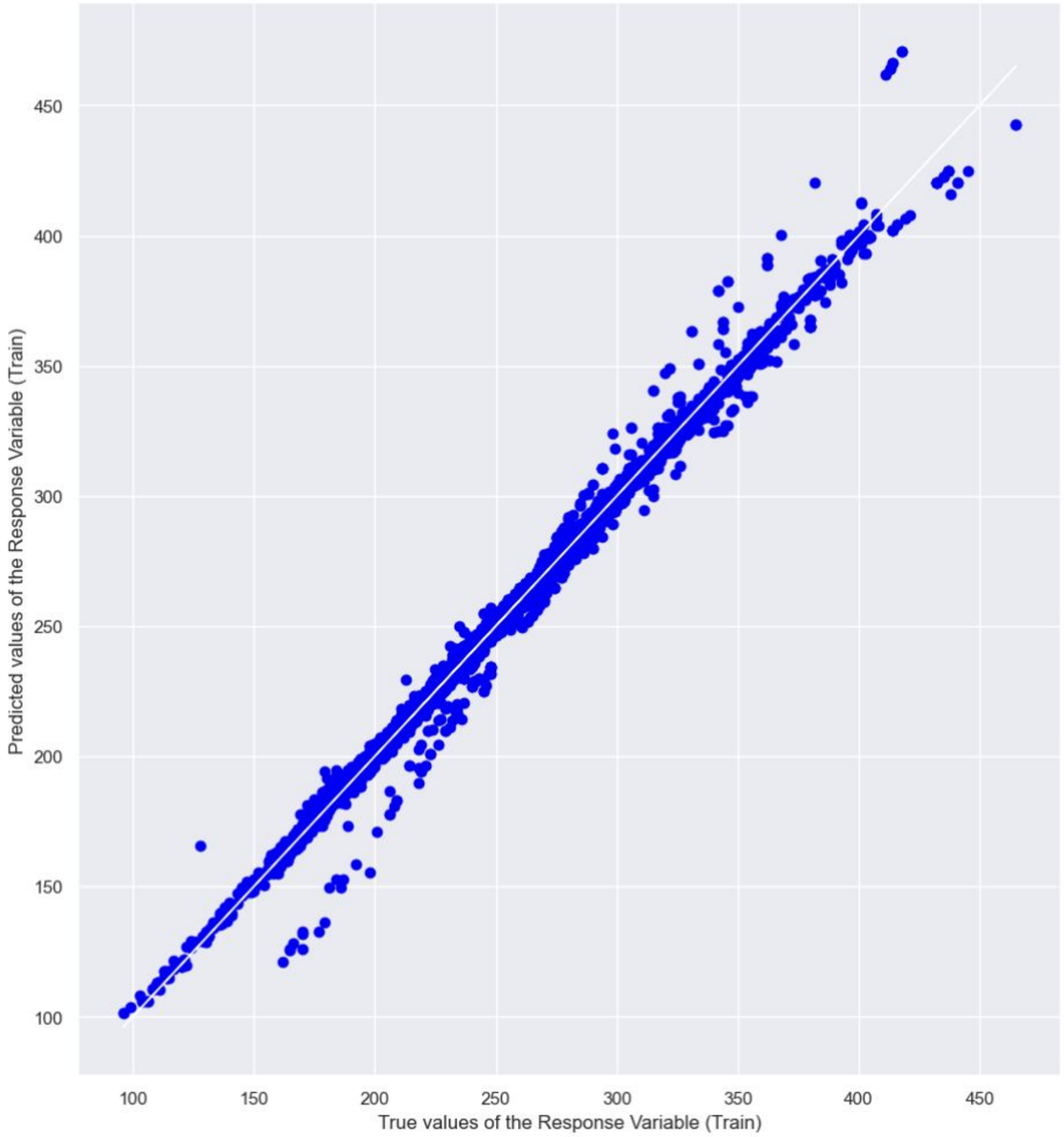
```
1 lasso_cv.best_estimator_.coef_
```

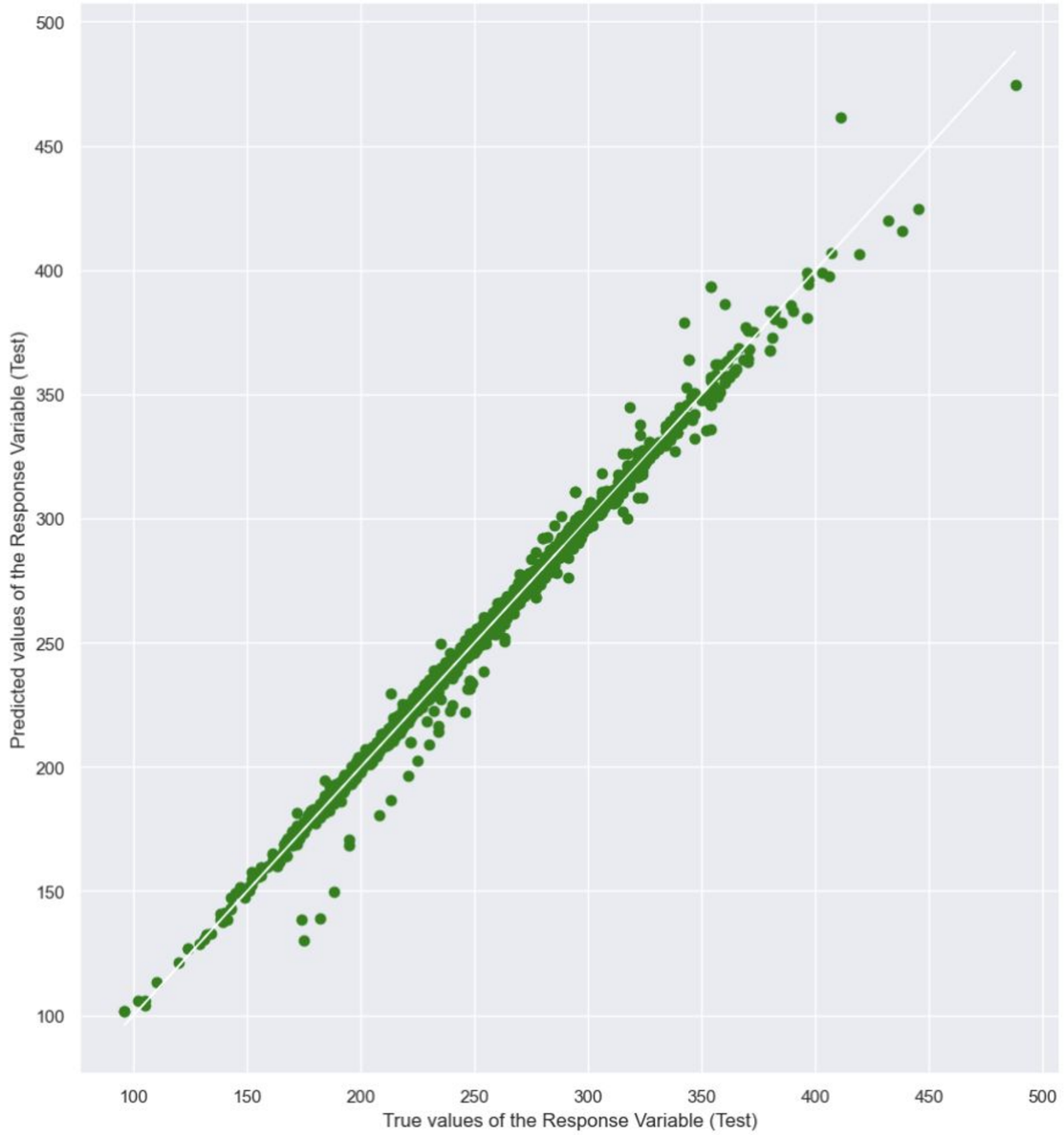
```
array([ 2.32746781e-01,  1.03493865e+00,  6.33593620e+01,  7.10680204e-01,
        4.55498780e+00, -2.42255008e+01, -1.05225664e+00, -0.00000000e+00,
        1.22225236e-01, -2.13763146e-01,  2.46745190e-01, -1.20651631e-01,
       -1.45804081e-01,  6.42525641e-02,  4.60342067e-01,  2.50783722e-01,
        4.41603284e-01, -3.05755826e-03, -1.38307218e-01, -6.09492868e-02,
        2.67806645e-01,  9.72316320e-02,  4.97972019e-03, -7.94087593e-01,
       -1.12529621e+00, -3.63775695e-03,  6.96158965e-02,  6.79317045e-02,
       -1.87007402e-01, -4.15954699e-03])
```

LEGEND: FT - Fuel Type, VClass - Vehicle Class, TType - Transmission Type

[Engine Size(L), Cylinders, Fuel Consumption Comb (L/100 km), Gears,
FT_Diesel, FT_Ethanol, FT_Natural Gas, FT_Premium Gasoline,
FT_Regular Gasoline, VClass_COMPACT, VClass_FULL-SIZE, VClass_MID-SIZE,
VClass_MINICOMPACT, VClass_MINIVAN, VClass_PICKUP TRUCK - SMALL, VClass_PICKUPTRUCK - STANDARD,
VClass_SPECIAL PURPOSE VEHICLE, VClass_STATION WAGON - MID-SIZE, VClass_STATION WAGON - SMALL, VClass_SUBCOMPACT,
VClass_SUV - SMALL, VClass_SUV - STANDARD, VClass_TWO-SEATER, VClass_VAN - CARGO,
VClass_VAN - PASSENGER, TType_A, TType_AM, TType_AS,
TType_AV, TType_M]

Model 3:
Lasso Regression





Comparing the 3 Models

Multi-linear Regression:

Explained Variance (R^2):
0.9907044602598732
Mean Squared Error (MSE):
28.628733087673407
Root Mean Squared Error (RMSE):
5.350582499847414
Standard Error of Prediction:
5.070121609360958

Ridge Regression:

Explained Variance (R^2):
0.9907984867550578
Mean Squared Error (MSE):
28.339146951842242
Mean Absolute Error:
3.030406582873876

Lasso Regression:

Explained Variance (R^2):
0.9907984867550578
Mean Squared Error (MSE):
28.339146951842242
Mean Absolute Error:
3.030406582873876

Most significant variable:

Fuel Consumption

Inrix 2022 Global Traffic Scorecard

Model 3:
Lasso Regression

```
1 lasso_cv.best_estimator_.intercept_
```

```
249.78379298604014
```

```
1 lasso_cv.best_estimator_.coef_
```

```
array([ 2.32746781e-01,  1.03493865e+00,  6.33593620e+01,  7.10680204e-01,
        4.55498780e+00, -2.42255008e+01, -1.05225664e+00, -0.00000000e+00,
        1.22225236e-01, -2.13763146e-01,  2.46745190e-01, -1.20651631e-01,
       -1.45804081e-01,  6.42525641e-02,  4.60342067e-01,  2.50783722e-01,
        4.41603284e-01, -3.05755826e-03, -1.38307218e-01, -6.09492868e-02,
        2.67806645e-01,  9.72316320e-02,  4.97972019e-03, -7.94087593e-01,
       -1.12529621e+00, -3.63775695e-03,  6.96158965e-02,  6.79317045e-02,
       -1.87007402e-01, -4.15954699e-03])
```

LEGEND: FT - Fuel Type, VClass - Vehicle Class, TType - Transmission Type

[Engine Size(L), Cylinders, Fuel Consumption Comb (L/100 km), Gears,
FT_Diesel, FT_Ethanol, FT_Natural Gas, FT_Premium Gasoline,
FT_Regular Gasoline, VClass_COMPACT, VClass_FULL-SIZE, VClass_MID-SIZE,
VClass_MINICOMPACT, VClass_MINIVAN, VClass_PICKUP TRUCK - SMALL, VClass_PICKUPTRUCK - STANDARD,
VClass_SPECIAL PURPOSE VEHICLE, VClass_STATION WAGON - MID-SIZE, VClass_STATION WAGON - SMALL, VClass_SUBCOMPACT,
VClass_SUV - SMALL, VClass_SUV - STANDARD, VClass_TWO-SEATER, VClass_VAN - CARGO,
VClass_VAN - PASSENGER, TType_A, TType_AM, TType_AS,
TType_AV, TType_M]

Recommendations

1. Plan your time and rush hour route
2. Take the public transport
3. Go for regular car servicing


blueSG
SLT 6810 S
9870000155
blueSG

Thank you!