In [5]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
import statsmodels.api as sm
from scipy.stats import shapiro, kstest,normaltest
import pickle
import json
import os
```

In [ ]:
```
Step 1: Problem statement: To predict the co2 emmission based on given data.

Understanding the Data
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 imperial gallon (mpg)
CO2 emissions   the tailpipe emissions of carbon dioxide (in grams per kilometre)
    for combined city and highway driving

```

In [ ]:
```
Step 2: Data Gathering.
    data gathering id a task data engineer,
    the current data set is downloaded from kaggle
```

In [10]:

```python
df=pd.read_csv('CO2 Emissions_Canada.csv')
df
```

Out[10]:

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Cor Co
0	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	
1	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	
3	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	
4	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	
...	...	...	...	...	...	...	...	...	...	
7380	VOLVO	XC40 T5 AWD	SUV - SMALL	2.0	4	AS8	Z	10.7	7.7	
7381	VOLVO	XC60 T5 AWD	SUV - SMALL	2.0	4	AS8	Z	11.2	8.3	
7382	VOLVO	XC60 T6 AWD	SUV - SMALL	2.0	4	AS8	Z	11.7	8.6	
7383	VOLVO	XC90 T5 AWD	SUV - STANDARD	2.0	4	AS8	Z	11.2	8.3	
7384	VOLVO	XC90 T6 AWD	SUV - STANDARD	2.0	4	AS8	Z	12.2	8.7	

7385 rows × 12 columns

In [3]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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  
dtypes: float64(4), int64(3), object(5)
memory usage: 692.5+ KB
```

In [11]:

```
x=df.drop(['Make','Model','CO2 Emissions(g/km)'], axis=1)
x
```

Out[11]:

	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)	Consur Comb
0	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	
1	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	
2	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	
3	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	
4	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	
...	...	...	...	...	...	...	...	...	
7380	SUV - SMALL	2.0	4	AS8	Z	10.7	7.7	9.4	
7381	SUV - SMALL	2.0	4	AS8	Z	11.2	8.3	9.9	
7382	SUV - SMALL	2.0	4	AS8	Z	11.7	8.6	10.3	
7383	SUV - STANDARD	2.0	4	AS8	Z	11.2	8.3	9.9	
7384	SUV - STANDARD	2.0	4	AS8	Z	12.2	8.7	10.7	

7385 rows × 9 columns

In [12]:

```
y=df[['CO2 Emissions(g/km)']]
y
```

Out[12]:

	CO2 Emissions(g/km)
0	196
1	221
2	136
3	255
4	244
...	...
7380	219
7381	232
7382	240
7383	232
7384	248

7385 rows × 1 columns

In [13]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 9 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Vehicle Class                      7385 non-null   object 
 1   Engine Size(L)                     7385 non-null   float64
 2   Cylinders                          7385 non-null   int64  
 3   Transmission                       7385 non-null   object 
 4   Fuel Type                          7385 non-null   object 
 5   Fuel Consumption City (L/100 km)   7385 non-null   float64
 6   Fuel Consumption Hwy (L/100 km)    7385 non-null   float64
 7   Fuel Consumption Comb (L/100 km)   7385 non-null   float64
 8   Fuel Consumption Comb (mpg)        7385 non-null   int64  
dtypes: float64(4), int64(2), object(3)
memory usage: 519.4+ KB
```

In [14]:
```
x= pd.get_dummies(x, columns=['Vehicle Class'])
```

In [15]:
```
x = pd.get_dummies(x, columns=['Transmission'])
```

In [16]:
```
x
```

Out[16]:

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	Vehicle Class_COMPACT	C
0	2.0	4	Z	9.9	6.7	8.5	33	True	
1	2.4	4	Z	11.2	7.7	9.6	29	True	
2	1.5	4	Z	6.0	5.8	5.9	48	True	
3	3.5	6	Z	12.7	9.1	11.1	25	False	
4	3.5	6	Z	12.1	8.7	10.6	27	False	
...	...	...	...	...	...	...	...	...	
7380	2.0	4	Z	10.7	7.7	9.4	30	False	
7381	2.0	4	Z	11.2	8.3	9.9	29	False	
7382	2.0	4	Z	11.7	8.6	10.3	27	False	
7383	2.0	4	Z	11.2	8.3	9.9	29	False	
7384	2.0	4	Z	12.2	8.7	10.7	26	False	

7385 rows × 50 columns

In [17]:
```python
x.replace({True:1,False:0},inplace=True)
x
```

Out[17]:

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	Vehicle Class_COMPACT	C
0	2.0	4	Z	9.9	6.7	8.5	33	1	
1	2.4	4	Z	11.2	7.7	9.6	29	1	
2	1.5	4	Z	6.0	5.8	5.9	48	1	
3	3.5	6	Z	12.7	9.1	11.1	25	0	
4	3.5	6	Z	12.1	8.7	10.6	27	0	
...	...	...	...	...	...	...	...	...	
7380	2.0	4	Z	10.7	7.7	9.4	30	0	
7381	2.0	4	Z	11.2	8.3	9.9	29	0	
7382	2.0	4	Z	11.7	8.6	10.3	27	0	
7383	2.0	4	Z	11.2	8.3	9.9	29	0	
7384	2.0	4	Z	12.2	8.7	10.7	26	0	

7385 rows × 50 columns

In [18]:
```python
x['Fuel Type'].unique()
```

Out[18]:
```
array(['Z', 'D', 'X', 'E', 'N'], dtype=object)
```

In [19]:
```python
x['Fuel Type'].replace({'Z':3, 'D':5, 'X':4, 'E':2, 'N':1},inplace=True)
```

In [20]:
```
1 x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 50 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Engine Size(L)                        7385 non-null   float64
 1   Cylinders                             7385 non-null   int64  
 2   Fuel Type                             7385 non-null   int64  
 3   Fuel Consumption City (L/100 km)      7385 non-null   float64
 4   Fuel Consumption Hwy (L/100 km)       7385 non-null   float64
 5   Fuel Consumption Comb (L/100 km)      7385 non-null   float64
 6   Fuel Consumption Comb (mpg)           7385 non-null   int64  
 7   Vehicle Class_COMPACT                 7385 non-null   int64  
 8   Vehicle Class_FULL-SIZE               7385 non-null   int64  
 9   Vehicle Class_MID-SIZE                7385 non-null   int64  
 10  Vehicle Class_MINICOMPACT             7385 non-null   int64  
 11  Vehicle Class_MINIVAN                 7385 non-null   int64  
 12  Vehicle Class_PICKUP TRUCK - SMALL    7385 non-null   int64  
 13  Vehicle Class_PICKUP TRUCK - STANDARD 7385 non-null   int64  
 14  Vehicle Class_SPECIAL PURPOSE VEHICLE 7385 non-null   int64  
 15  Vehicle Class_STATION WAGON - MID-SIZE 7385 non-null  int64  
 16  Vehicle Class_STATION WAGON - SMALL   7385 non-null   int64  
 17  Vehicle Class_SUBCOMPACT              7385 non-null   int64  
 18  Vehicle Class_SUV - SMALL             7385 non-null   int64  
 19  Vehicle Class_SUV - STANDARD          7385 non-null   int64  
 20  Vehicle Class_TWO-SEATER              7385 non-null   int64  
 21  Vehicle Class_VAN - CARGO             7385 non-null   int64  
 22  Vehicle Class_VAN - PASSENGER         7385 non-null   int64  
 23  Transmission_A10                      7385 non-null   int64  
 24  Transmission_A4                       7385 non-null   int64  
 25  Transmission_A5                       7385 non-null   int64  
 26  Transmission_A6                       7385 non-null   int64  
 27  Transmission_A7                       7385 non-null   int64  
 28  Transmission_A8                       7385 non-null   int64  
 29  Transmission_A9                       7385 non-null   int64  
 30  Transmission_AM5                      7385 non-null   int64  
 31  Transmission_AM6                      7385 non-null   int64  
 32  Transmission_AM7                      7385 non-null   int64  
 33  Transmission_AM8                      7385 non-null   int64  
 34  Transmission_AM9                      7385 non-null   int64  
 35  Transmission_AS10                     7385 non-null   int64  
 36  Transmission_AS4                      7385 non-null   int64  
 37  Transmission_AS5                      7385 non-null   int64  
 38  Transmission_AS6                      7385 non-null   int64  
 39  Transmission_AS7                      7385 non-null   int64  
 40  Transmission_AS8                      7385 non-null   int64  
 41  Transmission_AS9                      7385 non-null   int64  
 42  Transmission_AV                       7385 non-null   int64  
 43  Transmission_AV10                     7385 non-null   int64  
 44  Transmission_AV6                      7385 non-null   int64  
 45  Transmission_AV7                      7385 non-null   int64  
 46  Transmission_AV8                      7385 non-null   int64  
 47  Transmission_M5                       7385 non-null   int64  
 48  Transmission_M6                       7385 non-null   int64  
 49  Transmission_M7                       7385 non-null   int64  
dtypes: float64(4), int64(46)
memory usage: 2.8 MB
```

4. Feature Selection

In [21]:
```
df=pd.concat([x,y], axis=1)
df
```

Out[21]:

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	Vehicle Class_COMPACT	C
0	2.0	4	3	9.9	6.7	8.5	33	1	
1	2.4	4	3	11.2	7.7	9.6	29	1	
2	1.5	4	3	6.0	5.8	5.9	48	1	
3	3.5	6	3	12.7	9.1	11.1	25	0	
4	3.5	6	3	12.1	8.7	10.6	27	0	
...	...	...	...	...	...	...	...	...	
7380	2.0	4	3	10.7	7.7	9.4	30	0	
7381	2.0	4	3	11.2	8.3	9.9	29	0	
7382	2.0	4	3	11.7	8.6	10.3	27	0	
7383	2.0	4	3	11.2	8.3	9.9	29	0	
7384	2.0	4	3	12.2	8.7	10.7	26	0	

7385 rows × 51 columns

In [ ]:
```
Assumption 1 Linearity - to check the linearity between dependent and independent va
```

In [22]:
```
df.corr()
```

Out[22]:

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fue Consumptio Comb (mpg
Engine Size(L)	1.000000	0.927653	-0.276888	0.831379	0.761526	0.817060	-0.75785
Cylinders	0.927653	1.000000	-0.312808	0.800702	0.715252	0.780534	-0.71932
Fuel Type	-0.276888	-0.312808	1.000000	-0.464904	-0.428890	-0.457629	0.41428
Fuel Consumption City (L/100 km)	0.831379	0.800702	-0.464904	1.000000	0.948180	0.993810	-0.92705
Fuel Consumption Hwy (L/100 km)	0.761526	0.715252	-0.428890	0.948180	1.000000	0.977299	-0.89063
Fuel Consumption Comb (L/100 km)	0.817060	0.780534	-0.457629	0.993810	0.977299	1.000000	-0.92557
Fuel Consumption Comb (mpg)	-0.757854	-0.719321	0.414283	-0.927059	-0.890638	-0.925576	1.00000
Vehicle Class_COMPACT	-0.203355	-0.159493	-0.013841	-0.211595	-0.246223	-0.226017	0.23587
Vehicle Class_FULL-SIZE	0.122829	0.158072	-0.057081	0.096558	0.015905	0.070079	-0.07118
Vehicle Class_MID-SIZE	-0.085230	-0.096560	0.041311	-0.177091	-0.239639	-0.201011	0.22946
Vehicle Class_MINICOMPACT	-0.035092	0.001623	-0.120724	-0.060727	-0.056456	-0.059913	0.03400
Vehicle Class_MINIVAN	0.018929	0.010588	0.039049	0.035833	0.031265	0.034732	-0.04586
Vehicle Class_PICKUP TRUCK - SMALL	-0.006167	-0.059599	0.155465	0.022606	0.066215	0.038082	-0.06561
Vehicle Class_PICKUP TRUCK - STANDARD	0.274497	0.214065	0.047591	0.251740	0.328656	0.281617	-0.25312
Vehicle Class_SPECIAL PURPOSE VEHICLE	-0.064312	-0.081929	0.017549	-0.017083	0.012040	-0.007504	-0.01126
Vehicle Class_STATION WAGON - MID-SIZE	-0.002824	0.003864	-0.004941	-0.022640	-0.020634	-0.021841	0.02007
Vehicle Class_STATION WAGON - SMALL	-0.171660	-0.150949	0.089400	-0.162310	-0.143792	-0.158060	0.17190
Vehicle Class_SUBCOMPACT	-0.024449	0.007365	-0.121674	-0.025072	-0.070775	-0.041416	0.01208
Vehicle Class_SUV - SMALL	-0.212082	-0.238477	0.181754	-0.136099	-0.067386	-0.114150	0.04168
Vehicle Class_SUV - STANDARD	0.277005	0.240228	-0.042850	0.287485	0.335603	0.307492	-0.27659
Vehicle Class_TWO-SEATER	0.100990	0.141004	-0.183037	0.095279	0.059803	0.084318	-0.07682
Vehicle Class_VAN - CARGO	0.080514	0.065873	-0.034523	0.130020	0.172451	0.146337	-0.09125
Vehicle Class_VAN - PASSENGER	0.136153	0.098698	-0.009820	0.256174	0.286653	0.269795	-0.16211
Transmission_A10	0.115444	0.073244	-0.003898	0.047660	0.053677	0.050234	-0.05124
Transmission_A4	0.035005	0.021429	0.002715	0.092999	0.138233	0.109781	-0.05800
Transmission_A5	0.082283	0.046335	0.060460	0.082107	0.076236	0.080967	-0.08325
Transmission_A6	0.180285	0.103520	0.035965	0.240401	0.250183	0.247124	-0.20349
Transmission_A7	0.059610	0.093366	-0.065945	0.068980	0.073339	0.071227	-0.06932
Transmission_A8	0.222896	0.177551	0.009605	0.144139	0.115927	0.136018	-0.15362
Transmission_A9	-0.037827	-0.012209	-0.021471	-0.009958	-0.016534	-0.012002	-0.02427
Transmission_AM5	-0.037136	-0.033299	-0.018056	-0.037955	-0.032358	-0.036421	0.05156
Transmission_AM6	-0.116859	-0.108555	0.033012	-0.136236	-0.124887	-0.133960	0.19565
Transmission_AM7	0.043440	0.099700	-0.161241	0.048440	0.041557	0.047236	-0.05377

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fue Consumptio Comb (mpg
Transmission_AM8	0.005782	0.037235	-0.071368	-0.002122	-0.007263	-0.003825	-0.01084
Transmission_AM9	0.005061	0.004245	-0.015636	-0.008389	0.015936	0.000174	-0.00413
Transmission_AS10	0.058960	0.058953	0.048585	0.069644	0.092925	0.078809	-0.08413
Transmission_AS4	-0.009239	-0.014540	0.013313	-0.007790	-0.003638	-0.006402	0.00345
Transmission_AS5	0.014415	-0.004990	0.033590	0.010405	0.033420	0.018613	-0.02672
Transmission_AS6	-0.141620	-0.161483	0.161032	-0.066854	-0.049297	-0.062428	0.03926
Transmission_AS7	0.135143	0.135097	-0.132085	0.063726	0.073514	0.067832	-0.08083
Transmission_AS8	0.054816	0.127267	-0.178939	0.070785	0.007875	0.049413	-0.08543
Transmission_AS9	-0.014005	-0.020678	-0.028921	-0.018226	-0.017687	-0.018381	-0.00462
Transmission_AV	-0.165001	-0.163178	0.158423	-0.263754	-0.210674	-0.248187	0.35263
Transmission_AV10	-0.005862	-0.007232	-0.007703	-0.040751	-0.038925	-0.040942	0.04646
Transmission_AV6	-0.052374	-0.076331	0.085100	-0.133728	-0.083574	-0.117963	0.13448
Transmission_AV7	-0.085643	-0.078310	0.051741	-0.124949	-0.106043	-0.119910	0.13054
Transmission_AV8	-0.040485	-0.041885	-0.018030	-0.049058	-0.023289	-0.040648	0.03260
Transmission_M5	-0.160277	-0.149358	0.113679	-0.145531	-0.128692	-0.141211	0.14789
Transmission_M6	-0.155131	-0.154826	0.003904	-0.118235	-0.151254	-0.130877	0.11522
Transmission_M7	0.046261	0.045005	-0.086635	0.014994	-0.004302	0.008220	-0.03188
CO2 Emissions(g/km)	0.851145	0.832644	-0.255974	0.919592	0.883536	0.918052	-0.90742

51 rows × 51 columns

In [23]:
```
df.corr().tail(1)
```

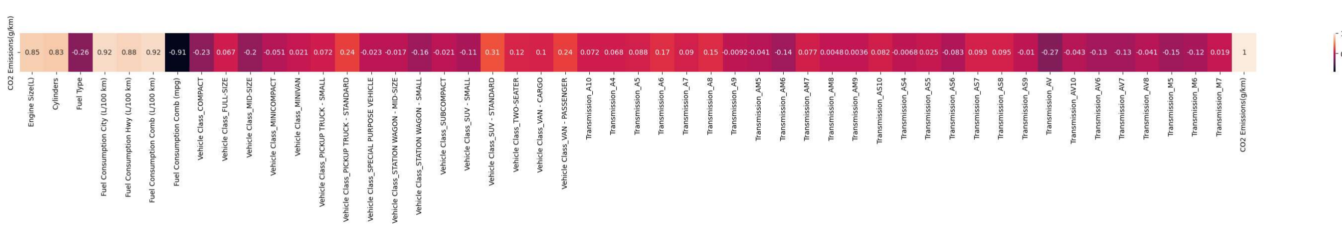
Out[23]:

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	Cl
CO2 Emissions(g/km)	0.851145	0.832644	-0.255974	0.919592	0.883536	0.918052	-0.907426	

1 rows × 51 columns

In [24]:
```
plt.figure(figsize=(40,1))
sns.heatmap(df.corr().tail(1),annot=True)
```

Out[24]: <Axes: >



In [ ]:
```
Assumption 2: No multicolinearity
there should not be any relation between independent variables
variance_inflation_factor is used to check the same
vif = 1/1-r2_Score
```

In [25]:
```
vif_index = variance_inflation_factor(x,0)
vif_index
```

Out[25]: 10.643168639918391

In [19]:
```
# vif_list= [variance_inflation_factor(x,i) for i in range(0,x.shape[1])]
# vif_list
```

step 5 Model Training

In [26]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=1)
x_train
```

Out[26]:

	Engine Size(L)	Cylinders	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	Vehicle Class_COMPACT	C
580	2.4	4	4	11.1	8.3	9.8	29	0	
3998	2.0	4	4	11.2	7.6	9.8	29	0	
2228	2.0	4	3	11.2	8.4	9.9	29	0	
2954	2.5	4	4	8.7	6.5	7.7	37	0	
4	3.5	6	3	12.1	8.7	10.6	27	0	
...	...	...	...	...	...	...	...	...	
905	3.4	6	3	11.9	8.6	10.4	27	0	
5192	2.0	4	3	9.3	7.3	8.4	34	0	
3980	2.0	4	3	10.7	8.5	9.7	29	0	
235	2.4	4	4	12.2	8.6	10.6	27	0	
5157	3.0	6	3	11.8	8.7	10.4	27	0	

5908 rows × 50 columns

In [27]:
```
lin_reg=LinearRegression()
lin_reg
```

Out[27]:
```
▾ LinearRegression
LinearRegression()
```

In [28]:
```
lin_reg.fit(x_train,y_train)
```

Out[28]:
```
▾ LinearRegression
LinearRegression()
```

In [29]:
```
y_pred=lin_reg.predict(x_test)
y_pred
```

Out[29]:
```
array([[220.0692432 ],
       [213.71540441],
       [165.18833474],
       ...,
       [173.39675343],
       [ 94.52234477],
       [192.09987737]])
```

In [30]:
```
y_pred_train = lin_reg.predict(x_train)
y_pred_train
```

Out[30]:
```
array([[220.99693219],
       [225.34882152],
       [218.24451026],
       ...,
       [204.64528846],
       [241.73841597],
       [235.94050767]])
```

Step 6 Model Evaluation: model is being evaluated in this step

In [31]:
```
residual = y_test - y_pred
residual
```

Out[31]:

	CO2 Emissions(g/km)
2196	-18.069243
5688	0.284596
7198	8.811665
6476	-7.893289
4909	9.949096
...	...
6773	-5.612443
2984	9.429215
4396	3.603247
5911	19.477655
3554	1.900123

1477 rows × 1 columns

In [32]:
```
residual_train = y_train - y_pred_train
```

In [33]:
```
mse_test = mean_squared_error(y_test,y_pred)
mse_test
```

Out[33]: 190.58098602127004

In [34]:
```
mae_test = mean_absolute_error(y_test,y_pred)
mae_test
```

Out[34]: 10.392285274136775

In [29]:
```
np.sqrt(mse_test)
```

Out[29]: 13.805107244106075

In [37]:
```
mse_train = mean_squared_error(y_train,y_pred_train)
mse_train
```

Out[37]: 183.98298277284985

In [36]:
```
mae_train = mean_absolute_error(y_train,y_pred_train)
mae_train
```

Out[36]: 10.270380939705657

In [38]:
```
np.sqrt(mse_train)
```

Out[38]: 13.564032688431928

In [39]:
```
r2score_test=r2_score(y_test,y_pred)
r2score_test
```

Out[39]: 0.9440266939018702

In [40]:
```
r2score_train=r2_score(y_train,y_pred_train)
r2score_train
```

Out[40]: 0.9463243086961876

In [41]:
```
adjusted_r2_test = 1 - ((x_test.shape[0]-1)*(1-r2score_test))/(x_test.shape[0]-x_tes
adjusted_r2_test
```

Out[41]: 0.9420640955113327

In [42]:
```
adjusted_r2_train = 1 - ((x_train.shape[0]-1)*(1-r2score_train))/(x_train.shape[0]-x
adjusted_r2_train
```

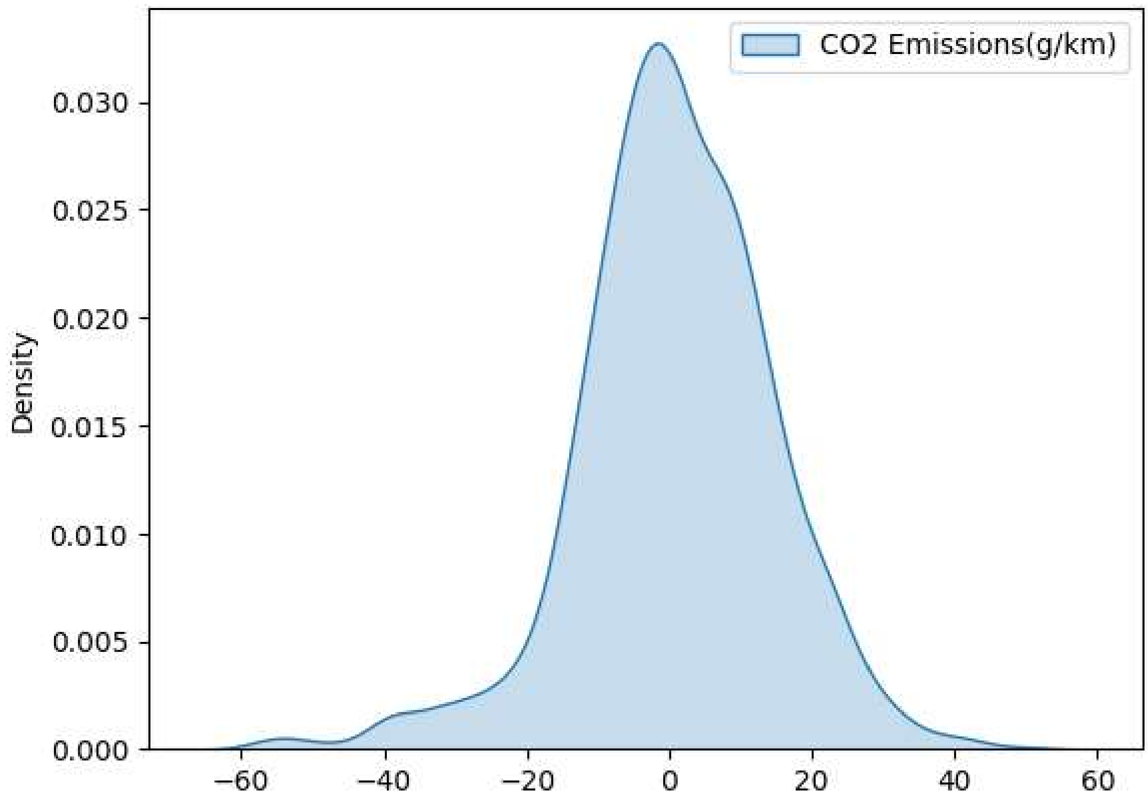
Out[42]: 0.9458660903992453

Assumption 3 Normality of residual

Visualization Techniques

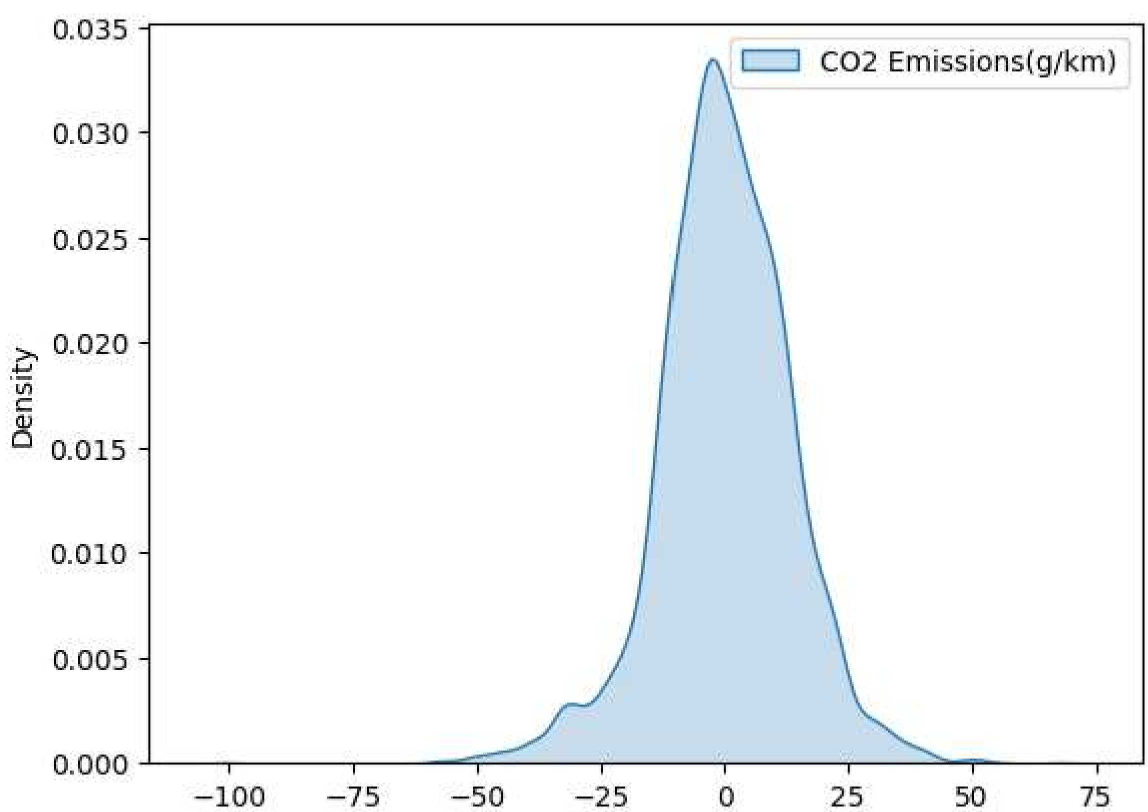
In [43]:
```
sns.kdeplot(residual,fill=True)
```

Out[43]: <Axes: ylabel='Density'>

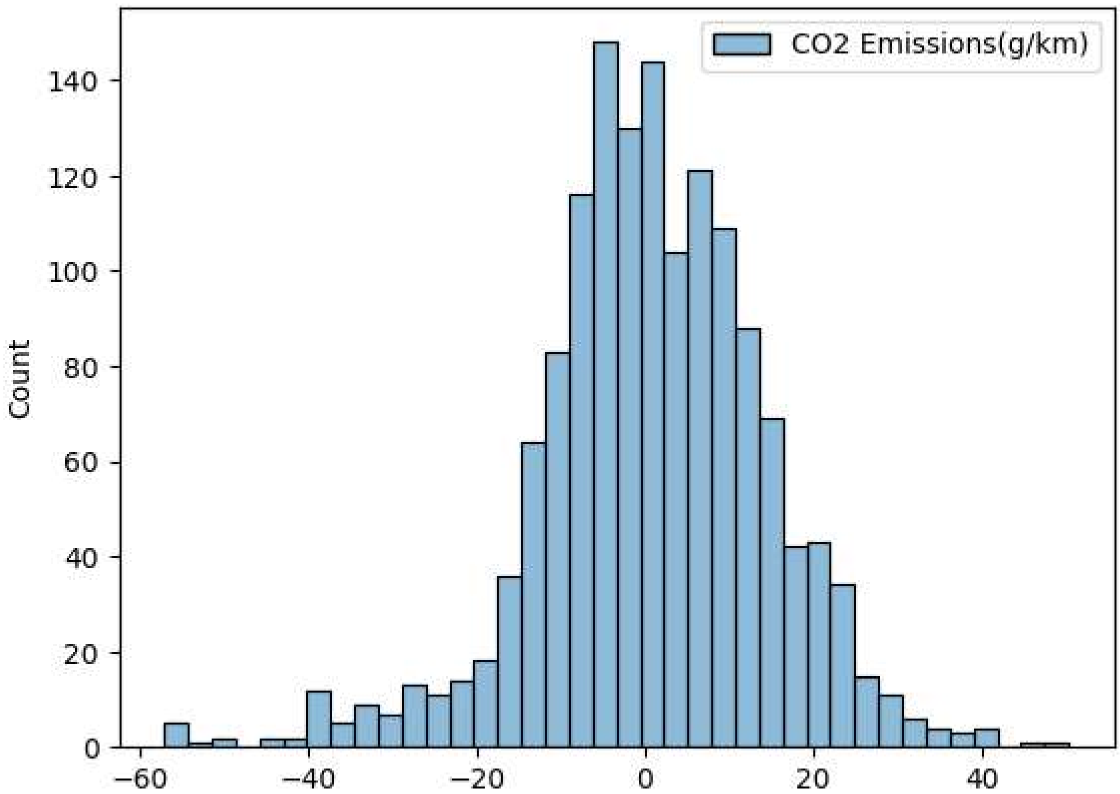


In [44]:
```
sns.kdeplot(residual_train,fill=True)
```

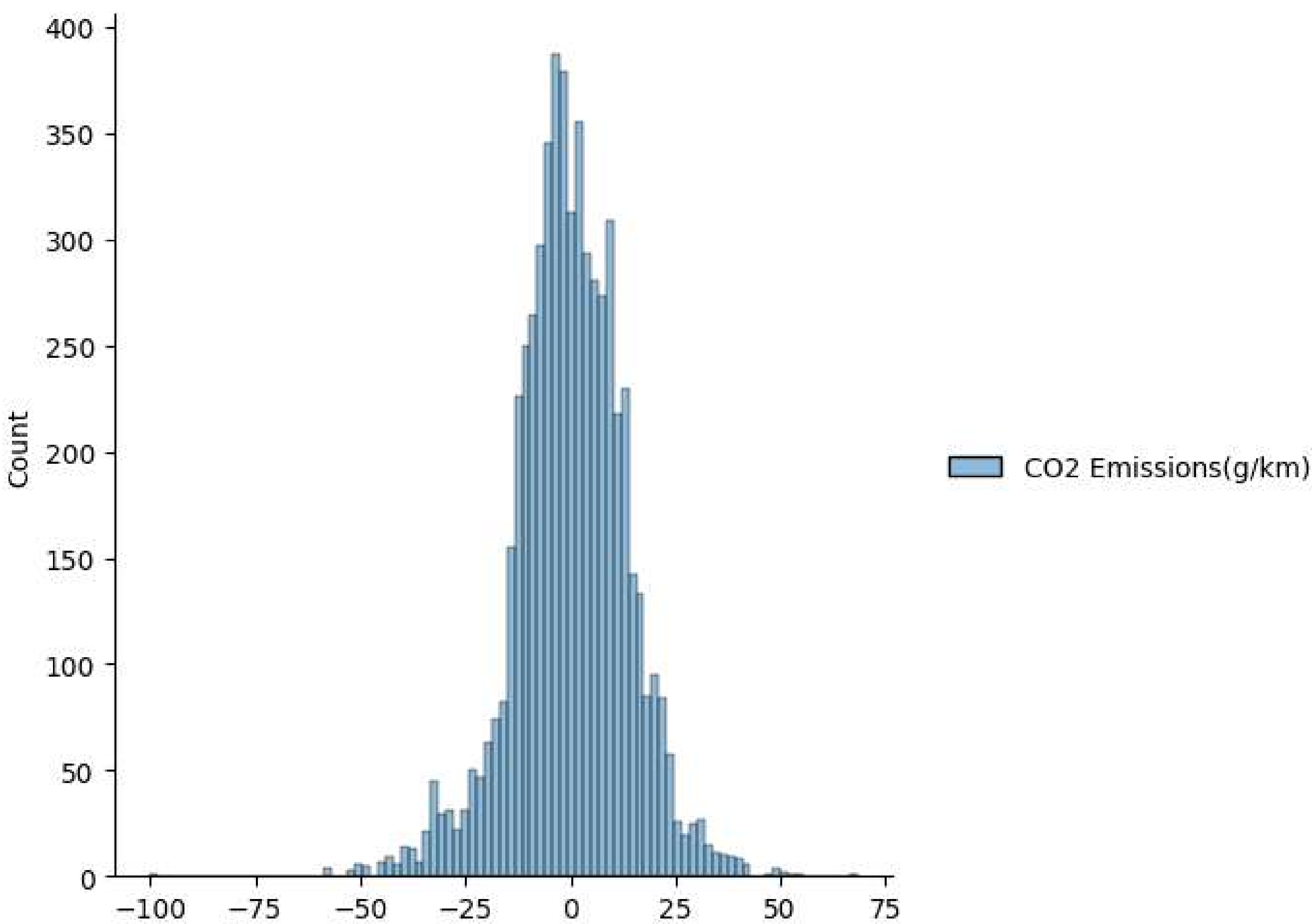
Out[44]: <Axes: ylabel='Density'>



In [45]:
```
1 sns.histplot(residual)
```
Out[45]: <Axes: ylabel='Count'>



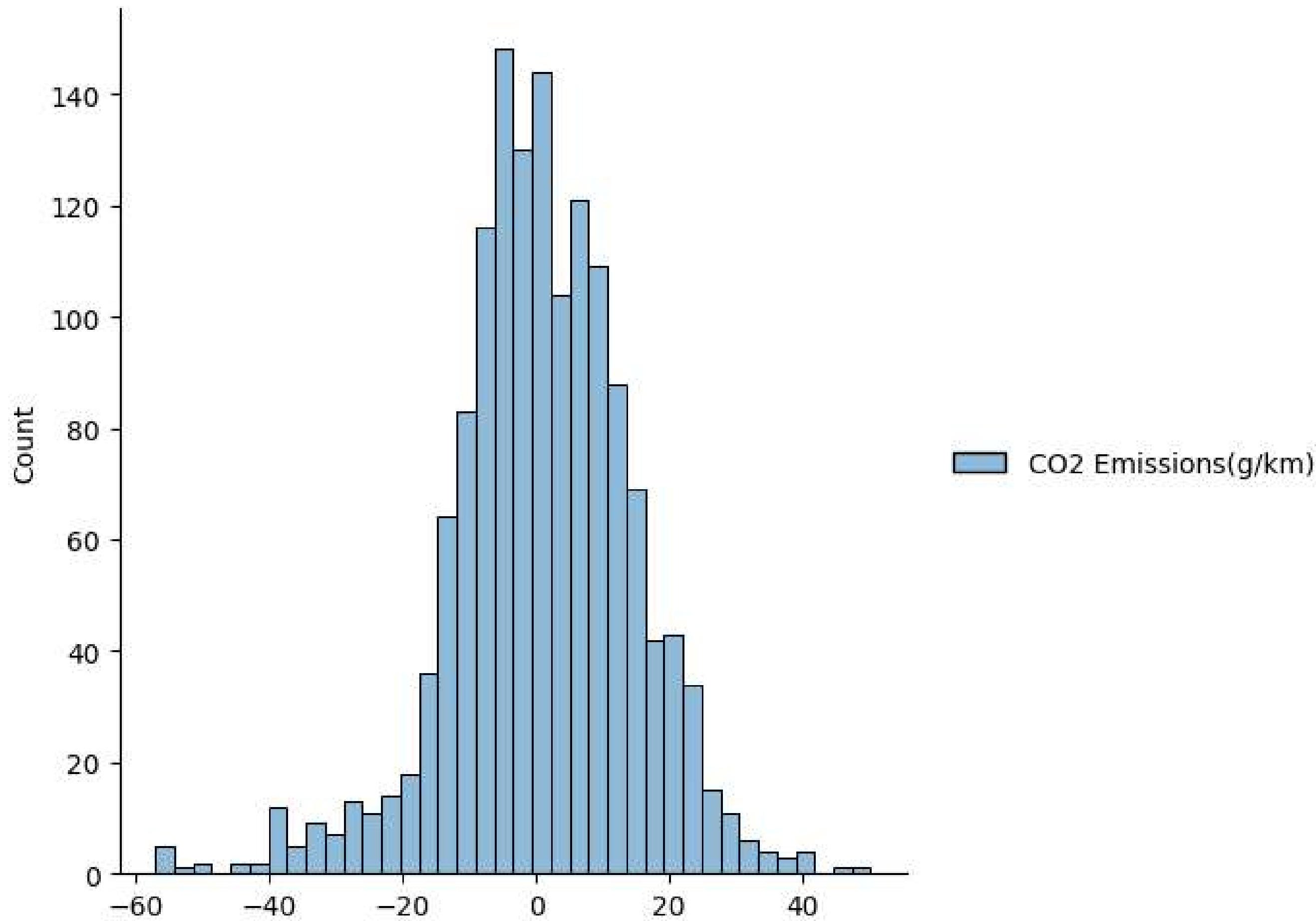
In [46]:
```
1 sns.displot(residual_train)
```
Out[46]: <seaborn.axisgrid.FacetGrid at 0x1c2265e9d90>



In [47]:

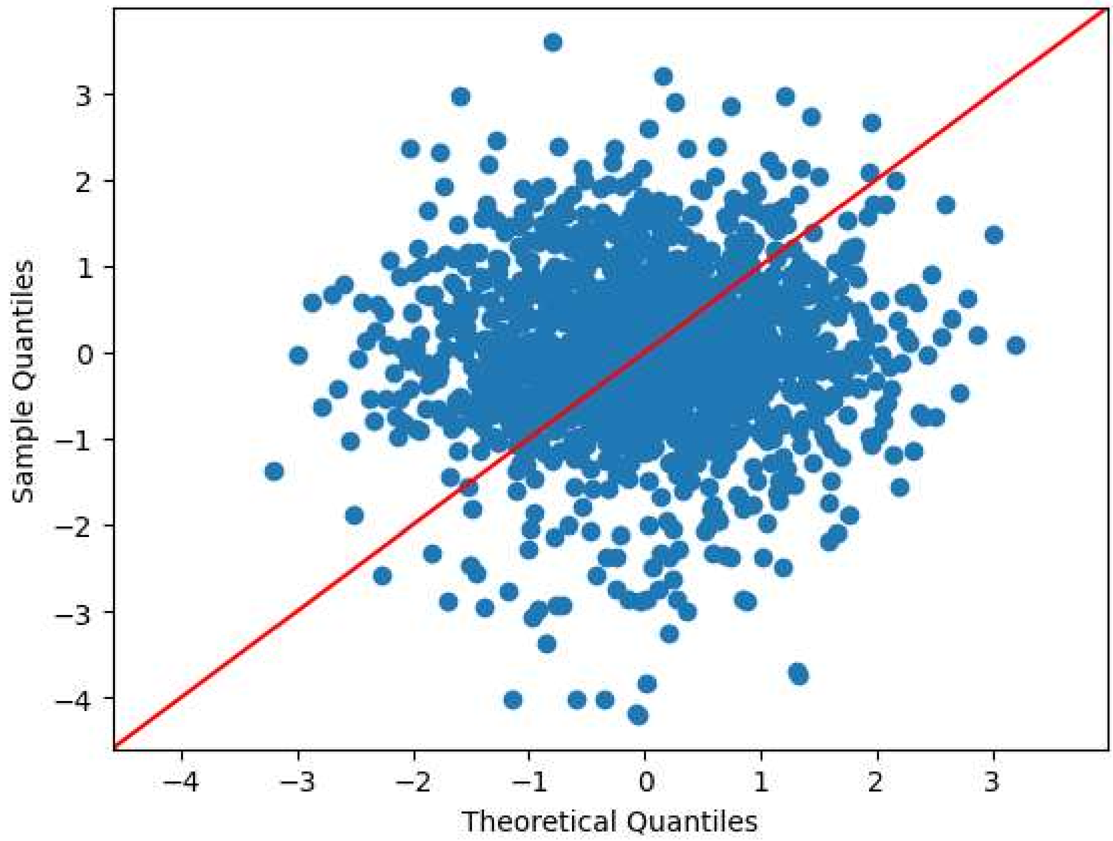
```
sns.displot(residual)
```

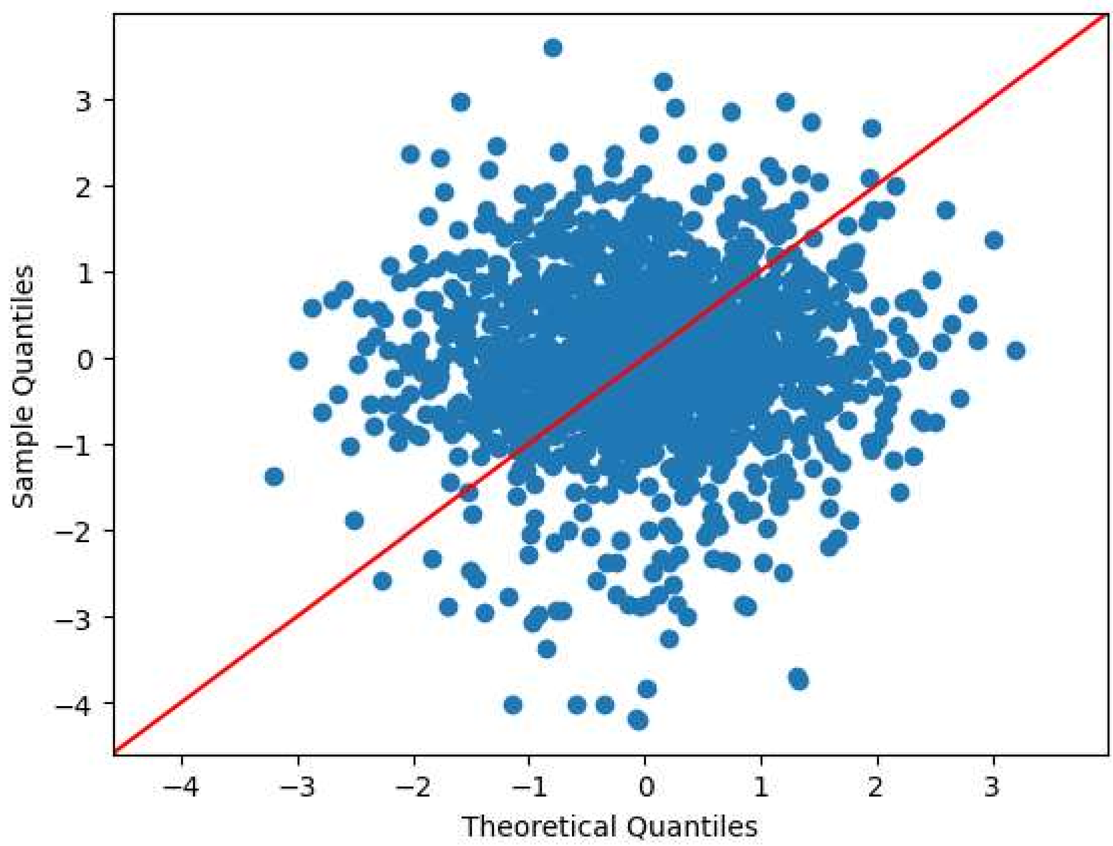
Out[47]: <seaborn.axisgrid.FacetGrid at 0x1c2294bf5d0>



In [49]:
```python
sm.qqplot(residual,line='45',fit=True)
```

Out[49]:





Hypothesis Testing

In [ ]:
```
Significance:
    if p_value is greater then 0.05 then residual is normally distributed
    else not normally distributed
```

shapiro test

In [50]:
```python
stat,p_value=shapiro(residual)
```

In [51]:
```python
if p_value>0.05:
    print('residual >> normally distributed')
else:
    print('residual >> not normally distributed')
```

```
residual >> not normally distributed
```

In [ ]:
```
kstest
```

In [42]:
```
_,p_value=normaltest(residual_train)
```

In [52]:
```
normaltest(residual_train)
```

Out[52]:
```
NormaltestResult(statistic=array([297.04677434]), pvalue=array([3.14132246e-65]))
```

In [53]:
```
if p_value>0.05:
    print('residual >> normally distributed')
else:
    print('residual >> not normally distributed')
```

```
residual >> not normally distributed
```

skewness

In [45]:
```
residual.skew()
```

Out[45]:
```
CO2 Emissions(g/km)   -0.437417
dtype: float64
```

In [46]:
```
from scipy.stats import skew
skew(residual)
```

Out[46]:
```
array([-0.43697275])
```

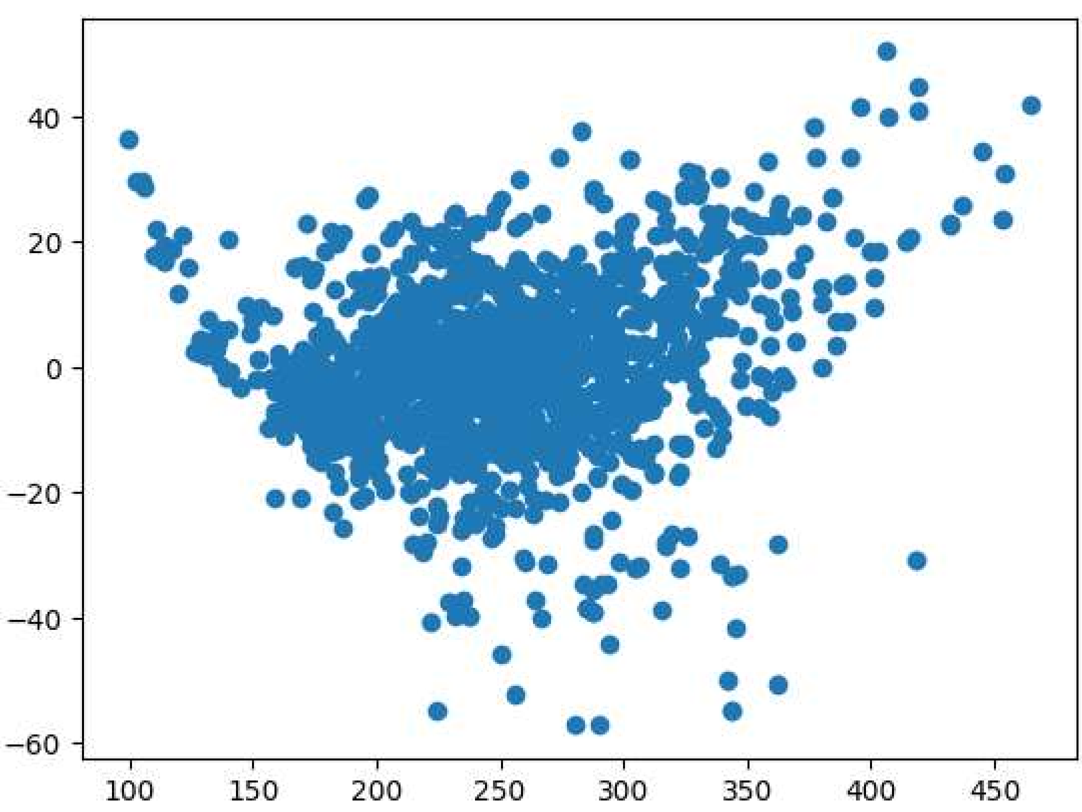
In [ ]:
```
-0.5 to 0.5 >> symmetrically distributed
-1 to-0.5 >> negetively skewed
0.5 to 1 >> positively skewed
1> highly positively skewed
-1< highily negetively skewed
```

In [ ]:
```
Assumption 4: Homoskedesticity
    if the value of residual goes on incresing as y increses, it is called as hetros
    Residual should be homoskedastic

```

In [54]:
```
plt.scatter(x= y_test,y= residual)
```

Out[54]:
```
<matplotlib.collections.PathCollection at 0x1c229663e10>
```




```
conclusion: model is performing well as
    1. we have low bias and low varience
```

```
    2. R2_score and r2_adjusted score is > 0.9
```

In [37]:

```
def get_input_row(make,model,Vehicle_Class, Engine_Size, Cylinders,Transmission, Fue
                  ,Fuel_Consumption_City1, Fuel_Consumption_Hwy1
                  ,Fuel_Consumption_Comb2, Fuel_Consumption_Comb3):
    df1=pd.DataFrame(np.zeros(shape=(50)))
    df1.index=x.columns
    df2=df1.T
    df2['Engine Size(L)']=Engine_Size
    df2['Cylinders']=Cylinders
    df2['Fuel Consumption City (L/100 km)']=Fuel_Consumption_City1
    df2['Fuel Consumption Hwy (L/100 km)']=Fuel_Consumption_Hwy1
    df2['Fuel Consumption Comb (L/100 km)']=Fuel_Consumption_Comb2
    df2['Fuel Consumption Comb (mpg)']=Fuel_Consumption_Comb3
    df2['Fuel Type']=Fuel_Type
    col_name='Vehicle Class_'+ Vehicle_Class
    df2[col_name]= 1
    col_name1='Transmission_'+ Transmission
    df2[col_name1]=1
    df2['Fuel Type'].replace({'Z':3, 'D':5, 'X':4, 'E':2, 'N':1},inplace=True)
    return df2
```

In [42]:

```
input_df=get_input_row('Suraj', 'SUV', 'COMPACT', 2.0, 4, 'AS5', 'Z', 9.9, 6.7, 8.6,
y_predicted = lin_reg.predict(input_df)
predicted_co2_emmission = y_predicted[0][0]
```

In [18]:

```
with open('linear_regression.pkl','wb') as f:
    pickle.dump(lin_reg,f)
```

In [25]:

```
dict1 = {'columns_x' : x.columns.to_list() }
with open('project_data.json','w') as f:
    json.dump(dict1,f)
```