In [50]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,GridSearchCV,RandomizedSearchCV
from sklearn.tree import DecisionTreeRegressor
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
```

### Step 1: Problem statement:

```
     To predict the organisational level co2 emmission based on the fule consumption vs co2 emission data, collected for different vehicles at different traffic conditions.

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

```

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

## Step 2: Data Gathering

In [2]:
```
df=pd.read_csv('CO2 Emissions_Canada.csv')
df
```

Out[2]:

| | 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

## Step 3: EDA & Feature Engineering

In [3]:
```
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 [4]:
```
x=df.drop(['Make','Model','CO2 Emissions(g/km)'], axis=1)
x
```

Out[4]:

| | 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 [5]:
```
y=df[['CO2 Emissions(g/km)']]
y
```

Out[5]:

| | 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 [6]:
```python
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 [7]:
```python
x= pd.get_dummies(x, columns=['Vehicle Class'])
```

In [8]:
```python
x = pd.get_dummies(x, columns=['Transmission'])
```

In [9]:
```python
x
```

Out[9]:

| | 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 [10]:
```
x.replace({True:1,False:0},inplace=True)
x
```

Out[10]:

| | 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 [11]:
```
x['Fuel Type'].unique()
```

Out[11]:
```
array(['Z', 'D', 'X', 'E', 'N'], dtype=object)
```

In [12]:
```
x['Fuel Type'].replace({'Z':3, 'D':5, 'X':4, 'E':2, 'N':1},inplace=True)
```

In [13]:
```
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
```

***There are no any null/missing values so missing values handling is not required***

***Here we are going to use Decision Tree Model, so Outlier Handling / Scaling is not required***

## Step 4: Feature Selection

***there are no any assumptions over data so we will be performing this step after model training and eveluation***

## step 5 Model Training

In [15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=1)
x_train
```

Out[15]:

| | 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 [18]:
```
dt_reg=DecisionTreeRegressor()
dt_reg
```

Out[18]: DecisionTreeRegressor()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [19]:
```
dt_reg.fit(x_train,y_train)
```

Out[19]: DecisionTreeRegressor()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [21]:
```
y_pred=dt_reg.predict(x_test)
y_pred
```

Out[21]: array([202., 214., 174., ..., 177., 115., 194.])

In [22]:
```
y_pred_train = dt_reg.predict(x_train)
y_pred_train
```

Out[22]: array([225., 230., 232., ..., 226., 244., 246.])

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

In [27]:
```
mse_test = mean_squared_error(y_test,y_pred)
mse_test
```

Out[27]: 7.408273443834266

In [28]:
```
mae_test = mean_absolute_error(y_test,y_pred)
mae_test
```

Out[28]: 1.5294978022804697

In [29]:
```
np.sqrt(mse_test)
```
Out[29]: 2.721814366159872

In [30]:
```
mse_train = mean_squared_error(y_train,y_pred_train)
mse_train
```
Out[30]: 1.2366056651728623

In [31]:
```
mae_train = mean_absolute_error(y_train,y_pred_train)
mae_train
```
Out[31]: 0.42930610632878735

In [32]:
```
np.sqrt(mse_train)
```
Out[32]: 1.1120277268003989

In [33]:
```
r2score_test=r2_score(y_test,y_pred)
r2score_test
```
Out[33]: 0.9978242028977429

In [34]:
```
r2score_train=r2_score(y_train,y_pred_train)
r2score_train
```
Out[34]: 0.9996392293300826

```
conclusion: model is performing well as
    1. we have low bias and low varience
    2. R2_score is > 0.99
    3. MSE and MAE values are low for both training and testing data.
```

## Hyper parameter tuning

In [51]:
```
df_reg_h = DecisionTreeRegressor()
param_grid = {'criterion' : ['squared_error', 'absolute_error'],
              'max_depth' : np.arange(5,15),
              'min_samples_split' : np.arange(3,8),
              'min_samples_leaf' : np.arange(2,5)}
gscv = RandomizedSearchCV(df_reg_h,param_grid,cv=5,n_jobs=-1)
```

In [52]:
```
gscv.fit(x_train,y_train)
```
Out[52]:
```
RandomizedSearchCV
RandomizedSearchCV(cv=5, estimator=DecisionTreeRegressor(), n_jobs=-1,
                   param_distributions={'criterion': ['squared_error',
                                                      'absolute_error'],
                                        'max_depth': array([ 5,  6,  7,  8,  9, 10,
11, 12, 13, 14]),
                                        'min_samples_leaf': array([2, 3, 4]),
                                        'min_samples_split': array([3, 4, 5, 6,
7])})
estimator: DecisionTreeRegressor
DecisionTreeRegressor()
DecisionTreeRegressor
DecisionTreeRegressor()
```

In [53]:
```
gscv.best_estimator_
```
Out[53]:
```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=9, min_samples_leaf=2, min_samples_split=3)
```

In [54]:
```
dt_reg_hyp = DecisionTreeRegressor(max_depth=9, min_samples_leaf=2, min_samples_spli
dt_reg_hyp.fit(x_train,y_train)
```

Out[54]:
```
DecisionTreeRegressor

DecisionTreeRegressor(max_depth=9, min_samples_leaf=2, min_samples_split=3)
```

In [55]:
```
y_pred=dt_reg_hyp.predict(x_test)
y_pred_train = dt_reg_hyp.predict(x_train)
mse_test = mean_squared_error(y_test,y_pred)
print('mse_test', mse_test)
mae_test = mean_absolute_error(y_test,y_pred)
print('mae_test', mae_test)
print('RMSE Test', np.sqrt(mse_test))
r2score_test=r2_score(y_test,y_pred)
print('r2score_test',r2score_test)
mse_train = mean_squared_error(y_train,y_pred_train)
print('mse_train', mse_train)
mae_train = mean_absolute_error(y_train,y_pred_train)
print('mae_train', mae_train)
print('RMSE Train', np.sqrt(mse_train))
r2score_train=r2_score(y_train,y_pred_train)
print('r2score_train',r2score_train)
```

```
mse_test 7.369223276074442
mae_test 2.064009165325812
RMSE Test 2.71463133336268
r2score_test 0.9978356718645
mse_train 6.131035844703403
mae_train 1.8566362075012992
RMSE Train 2.476092858659263
r2score_train 0.9982113150770079
```

```
conclusion: By using best parameters from hyper parameter tuning,
    model is performing well as
    1. we have low bias and low varience
    2. R2_score is > 0.99
    3. MSE and MAE values are low for both training and testing data.
```

### Creating get data function, pickle and json data file

In [35]:
```
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 [56]:
```
input_df=get_input_row('Suraj', 'SUV', 'COMPACT', 2.0, 4, 'AS5', 'Z', 9.9, 6.7, 8.6,
y_predicted = dt_reg_hyp.predict(input_df)
predicted_co2_emmission = y_predicted[0]
```

In [57]:
```
with open('DT_regression.pkl','wb') as f:
    pickle.dump(dt_reg_hyp,f)
```

In [39]:

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