Petrol Consumption

For the given dataset, predict the co-relation between petrol consumption and different features affecting it.

Criteria: Low RMSE to pass.

Importing the Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Data Preprocessing

```
df = pd.read_csv('petrol_consumption.csv')
```

```
df.head()
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petro
0	9.0	3571	1976	0.525	
1	9.0	4092	1250	0.572	
2	9.0	3865	1586	0.580	
3	7.5	4870	2351	0.529	
4	8.0	4399	431	0.544	

```
df.describe()
```

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	P
count	48.000000	48.000000	48.000000	48.000000	
mean	7.668333	4241.833333	5565.416667	0.570333	
std	0.950770	573.623768	3491.507166	0.055470	
min	5.000000	3063.000000	431.000000	0.451000	
25%	7.000000	3739.000000	3110.250000	0.529750	
50%	7.500000	4298.000000	4735.500000	0.564500	
75%	8.125000	4578.750000	7156.000000	0.595250	
max	10.000000	5342.000000	17782.000000	0.724000	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Petrol_tax                    48 non-null     float64
 1   Average_income                48 non-null     int64
 2   Paved_Highways                48 non-null     int64
 3   Population_Driver_licence(%)  48 non-null     float64
 4   Petrol_Consumption            48 non-null     int64
dtypes: float64(2), int64(3)
memory usage: 2.0 KB
```

```
X = df.iloc[: , : -1].values
y = df.iloc[: ,  -1].values
```

```
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 = 0)
```

Model of choice

The model of choice for this challenge will be a decision tree Regressor. The reason behind this is simple. The decision tree model is well adapted to higher dimensional datasets, and additionally no preprocessing is needed.

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

Calculating The Loss after training

The loss after training is calculated through Root mean squared error function, which is a cost function. Basically:

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

The RMSE of a model determines the absolute fit of the model to the data. In other words, it indicates how close the actual data points are to the model's predicted values. A low value of RMSE indicates a better fit and is a good measure for determining the accuracy of the model's predictions.

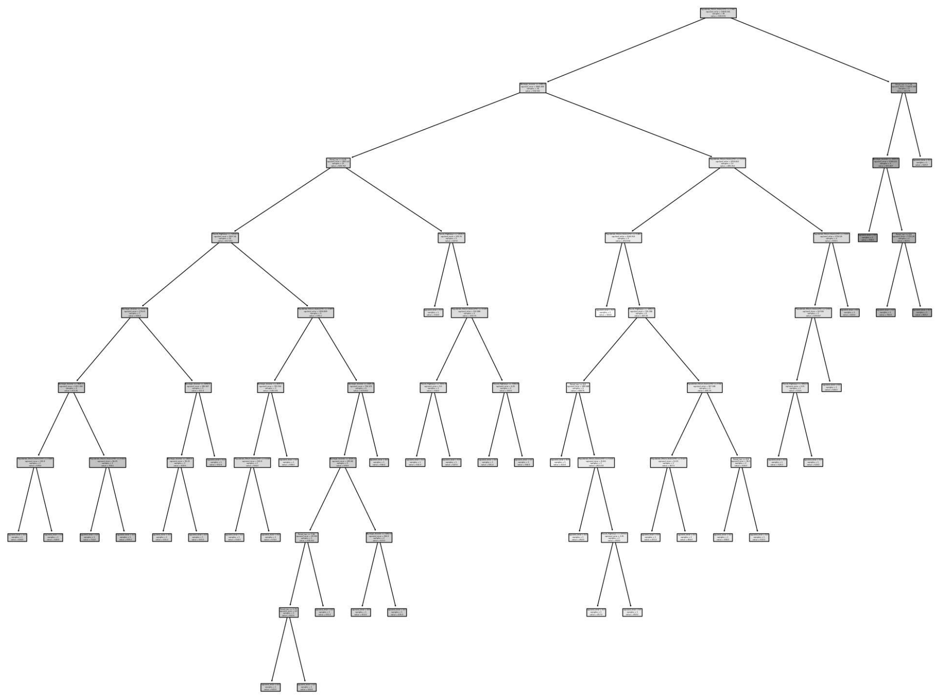
```
from sklearn.metrics import mean_squared_error

y_pred = regressor.predict(X_test)

rmse = float(format(np.sqrt(mean_squared_error(y_test, y_pred)), '.3f'))


print(f'RMSE: ', rmse)
```

```
RMSE:  67.345
```

Visualising the decision tree

Decision trees in higher dimensions are generally not visualizable. However, there is a function in the sklearn.tree module which does that. We will use that function to visualize the entire decision tree which our model built and used.

```
from sklearn.tree import plot_tree
fig = plt.figure(figsize=(25,20))
_ = plot_tree(regressor,
                   feature_names=['Petrol_tax', 'Average_income', 'Paved_Highways', 'Population_Driver_licence(%)'],
                   filled=True)
fig.savefig("decision_tree.pdf")
```



✓ 4s completed at 12:43 AM