```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
fuel_data = pd.read_csv('fuel_consumption_dataset (2).csv')
fuel_data.head()
```

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUM
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	

```
fuel_data.isnull()
```

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMP
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
1062	False	False	False	False	False	False	False	False	False	
1063	False	False	False	False	False	False	False	False	False	
1064	False	False	False	False	False	False	False	False	False	
1065	False	False	False	False	False	False	False	False	False	
1066	False	False	False	False	False	False	False	False	False	

1067 rows × 13 columns

```
print("\n=== UNIQUE VALUES PER COLUMN (FUEL) ===")
for col in fuel_data.columns:
    print(f"{col}: {fuel_data[col].nunique()} unique values")
```

```
=== UNIQUE VALUES PER COLUMN (FUEL) ===
MODELYEAR: 1 unique values
MAKE: 39 unique values
MODEL: 663 unique values
VEHICLECLASS: 16 unique values
ENGINESIZE: 45 unique values
CYLINDERS: 7 unique values
TRANSMISSION: 22 unique values
FUELTYPE: 4 unique values
FUELCONSUMPTION_CITY: 167 unique values
FUELCONSUMPTION_HWY: 118 unique values
FUELCONSUMPTION_COMB: 148 unique values
FUELCONSUMPTION_COMB_MPG: 43 unique values
CO2EMISSIONS: 159 unique values
```

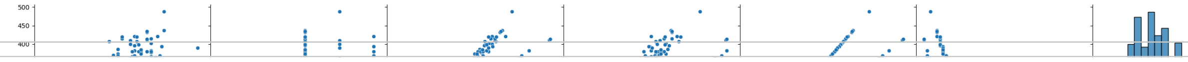
```
print("\n=== FUEL DATASET INFO ===")
print(fuel_data.info())

print("\n=== FUEL DATASET DESCRIPTION ===")
print(fuel_data.describe())
```

```
print("\n=== MISSING VALUES (FUEL) ===")
print(fuel_data.isnull().sum())

print("\n=== UNIQUE VALUES PER COLUMN (FUEL) ===")
for col in fuel_data.columns:
    print(f"{col}: {fuel_data[col].nunique()} unique values")
```

```
mean      2014.0     3.346298     5.794752             13.296532
std          0.0     1.415895     1.797447              4.101253
min       2014.0     1.000000     3.000000              4.600000
25%       2014.0     2.000000     4.000000             10.250000
50%       2014.0     3.400000     6.000000             12.600000
75%       2014.0     4.300000     8.000000             15.550000
max       2014.0     8.400000    12.000000             30.200000

       FUELCONSUMPTION_HWY  FUELCONSUMPTION_COMB  FUELCONSUMPTION_COMB_MPG  \
count          1067.000000           1067.000000               1067.000000
mean              9.474602             11.580881                 26.441425
std               2.794510              3.485595                  7.468702
min               4.900000              4.700000                 11.000000
25%               7.500000              9.000000                 21.000000
50%               8.800000             10.900000                 26.000000
75%              10.850000             13.350000                 31.000000
max              20.500000             25.800000                 60.000000

       CO2EMISSIONS
count   1067.000000
mean     256.228679
std       63.372304
min      108.000000
25%      207.000000
50%      251.000000
75%      294.000000
max      488.000000

=== MISSING VALUES (FUEL) ===
MODELYEAR                   0
MAKE                        0
MODEL                       0
VEHICLECLASS                0
ENGINESIZE                  0
CYLINDERS                   0
TRANSMISSION                0
FUELTYPE                    0
FUELCONSUMPTION_CITY        0
FUELCONSUMPTION_HWY         0
FUELCONSUMPTION_COMB        0
FUELCONSUMPTION_COMB_MPG    0
CO2EMISSIONS                0
dtype: int64

=== UNIQUE VALUES PER COLUMN (FUEL) ===
MODELYEAR: 1 unique values
MAKE: 39 unique values
MODEL: 663 unique values
VEHICLECLASS: 16 unique values
ENGINESIZE: 45 unique values
CYLINDERS: 7 unique values
TRANSMISSION: 22 unique values
FUELTYPE: 4 unique values
FUELCONSUMPTION_CITY: 167 unique values
FUELCONSUMPTION_HWY: 118 unique values
FUELCONSUMPTION_COMB: 148 unique values
FUELCONSUMPTION_COMB_MPG: 43 unique values
CO2EMISSIONS: 159 unique values
```

```
sns.pairplot(
    data=fuel_data,
    x_vars=['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY', 'FUELCONSUMPTION_HWY',
            'FUELCONSUMPTION_COMB', 'FUELCONSUMPTION_COMB_MPG','CO2EMISSIONS'],
    y_vars=['CO2EMISSIONS'],
    height=4,
    aspect=1
)
```

```
<seaborn.axisgrid.PairGrid at 0x7df96dbb7d90>
```



```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

numeric_cols = ['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY',
                'FUELCONSUMPTION_HWY', 'FUELCONSUMPTION_COMB', 'FUELCONSUMPTION_COMB_MPG']
categorical_cols = ['MAKE', 'MODEL', 'VEHICLECLASS', 'FUELTYPE', 'TRANSMISSION']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ])

X = preprocessor.fit_transform(fuel_data)
y = fuel_data['CO2EMISSIONS']
```

```python
X.shape
```

```
(1065, 750)
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
from sklearn.tree import DecisionTreeRegressor


model_fuel = DecisionTreeRegressor(random_state=0)


model_fuel.fit(X_train, y_train)
```

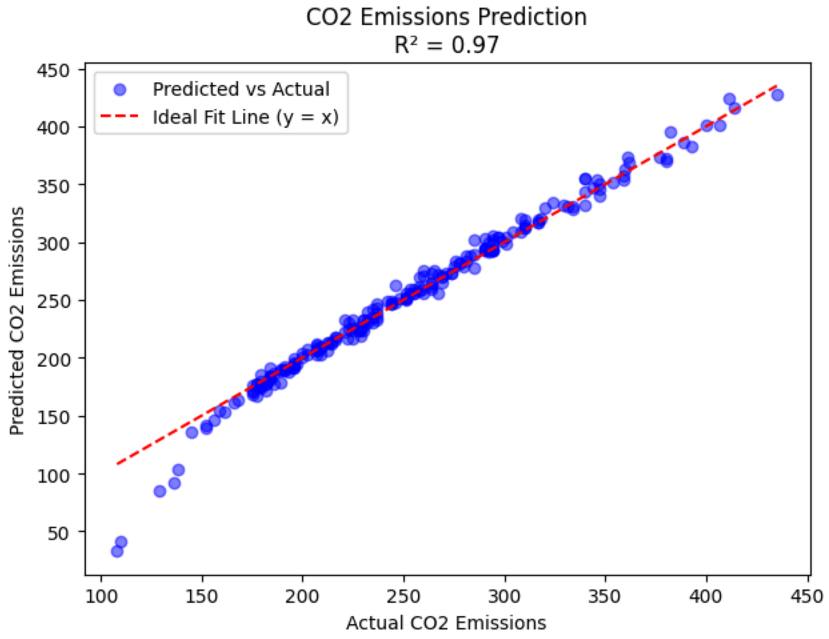
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

```python
y_pred = model_fuel.predict(X_test)
```

```python
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"📊 Regression Evaluation Metrics:")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"R² Score: {r2:.4f}")
```

```
📊 Regression Evaluation Metrics:
Mean Absolute Error (MAE): 1.1127
Mean Squared Error (MSE): 23.2817
Root Mean Squared Error (RMSE): 4.8251
R² Score: 0.9942
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred, alpha=0.5, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label='Ideal Fit Line (y = x)')
plt.xlabel('Actual CO2 Emissions')
plt.ylabel('Predicted CO2 Emissions')
plt.title(f'CO2 Emissions Prediction\nR² = {r2_fuel:.2f}')
plt.legend()

plt.tight_layout()
plt.show()
```



Start coding or generate with AI.