NO.29

Decision Support System

FUEL CONSUMPTION PREDICTED MODEL

Instructor: Dr. Le Hai Ha

Studane Name: Le Khanh Ngoc

Student ID: 20213082

DATA PREPARATION

The opacity of fuel consumption data for individual vehicles, particularly those from various remote manufacturers, creates a challenge for both consumers and manufacturers. To address this issue, this project focuses on developing a model to predict fuel efficiency for light-duty vehicles sold in Canada. We will utilize a dataset containing model-specific fuel consumption ratings to achieve this goal.

Your paragraph text

Understanding fuel efficiency is critical for both parties involved. Consumers benefit from accurate estimates that empower informed purchasing decisions, while manufacturers gain valuable insight for optimizing fuel efficiency, leading to environmental advantages and potentially a stronger market position. Through analysis of this dataset, we aim to build a model that predicts fuel consumption for various vehicle models by using **REGRESSION MODELS,** ultimately serving the needs of both consumers and manufacturers.

DATA DESCRIPTION

Source dataset: https://www.kaggle.com/datasets/ahmettyilmazz/fuel-consumption/data

Datasets provide model-specific fuel consumption ratings and estimated carbon dioxide emissions for new light-duty vehicles for retail sale in Canada. The fuel consumption ratings for 2000 to 2022 vehicles have been adjusted to reflect the improved testing that is more representative of everyday driving. Note that these are approximate values that were generated from the original ratings, not from vehicle testing.

Some details of this dataset:

Model:

4WD/4X4 = Four-wheel drive
CNG = Compressed natural gas
NGV = Natural gas vehicle
AWD = All-wheel drive
FFV = Flexible-fuel vehicle
# = High output engine that provides more power than the standard engine of the same size

Transmission:

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

Fuel Type:

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

Fuel Consumption:

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

CO2 Emissions (g/km):

- Are estimated tailpipe carbon dioxide emissions (in grams per kilometre) based on fuel type and the combined fuel consumption rating.

DATA PREPARATION

DATA SELECTION

As you may already know, the dataset consists of 12 columns: 'YEAR', 'MAKE', 'MODEL', 'VEHICLE CLASS', 'ENGINE SIZE', 'CYLINDERS', 'TRANSMISSION', 'FUEL', 'FUEL CONSUMPTION', 'HWY (L/100 km)', 'COMB (L/100 km)', 'COMB (mpg)', and 'EMISSIONS'. However, in this project, we have excluded and will not utilize the 'EMISSIONS' variable as an independent variable for predicting fuel consumption. It is based on the fact that 'EMISSIONS' is predicted from 'FUEL CONSUMPTION'.

Input		Output
YEAR	MAKE	FUEL CONSUMPTION
MODEL	VEHICLE CLASS	
ENGINE SIZE	CYLINDERS	
TRANSMISSION	FUEL	
HWY (L/100km)	COMB (L/100km)	
COMB (mpg)		

OBJECTIVES

- Data cleaning: Clean and preprocess the data, label the data, and remove outliers.
- Build predictive models for fuel consumption: Develop models using various algorithms such as Linear Regression, XGBoost, GBM, Decision Tree, CatBoost.
- Model evaluation: Evaluate the models based on their predictive performance.
- Predict with new dataset.

DATA PRE-PROCESSING

Importing libraries

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

Reading dataset

```
#Load the csv into a dataframe
df = pd.read_csv('Fuel_Consumption_2000-2022.csv')
df.head()
```

	YEAR	MAKE	MODEL	VEHICLE CLASS	ENGINE SIZE	CYLINDERS	TRANSMISSION	FUEL	FUEL CONSUMPTION	HWY (L/100 km)	COMB (L/100 km)	COMB (mpg)	EMISSIONS
0	2000	ACURA	1.6EL	COMPACT	1.6	4	A4	X	9.2	6.7	8.1	35	186
1	2000	ACURA	1.6EL	COMPACT	1.6	4	M5	X	8.5	6.5	7.6	37	175
2	2000	ACURA	3.2TL	MID-SIZE	3.2	6	AS5	Z	12.2	7.4	10.0	28	230
3	2000	ACURA	3.5RL	MID-SIZE	3.5	6	A4	Z	13.4	9.2	11.5	25	264
4	2000	ACURA	INTEGRA	SUBCOMPACT	1.8	4	A4	X	10.0	7.0	8.6	33	198

```
df.columns
#12 colums: 'YEAR', 'MAKE', 'MODEL', 'VEHICLE CLASS', 'ENGINE SIZE', 'CYLINDERS', 'TRANSMISSION', 'FUEL', 'FUEL CONSUMPTION', 'HWY (L/100 km)','COMB (L/100 km)', 'COMB (mpg)', 'EMISSIONS'

#Check data types and the information of this data frame
df.dtypes
df.info()
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   YEAR              22555 non-null  int64
 1   MAKE              22555 non-null  int32
 2   MODEL             22555 non-null  int32
 3   VEHICLE CLASS     22555 non-null  int32
 4   ENGINE SIZE       22555 non-null  float64
 5   CYLINDERS         22555 non-null  int64
 6   TRANSMISSION      22555 non-null  int32
 7   FUEL              22555 non-null  int32
 8   FUEL CONSUMPTION  22555 non-null  float64
 9   HWY (L/100 km)    22555 non-null  float64
 10  COMB (L/100 km)   22555 non-null  float64
 11  COMB (mpg)        22555 non-null  int64
dtypes: float64(4), int32(5), int64(3)
```

Checking duplicated value

```
#Check duplicated value
df.duplicated().sum() #Output: 1 duplicated row
df.drop_duplicates(inplace=True) #Remove duplicated row
```

DATA PRE-PROCESSING

Checking missing value

```
df.isnull().sum()
```

```
YEAR                0
MAKE                0
MODEL               0
VEHICLE CLASS       0
ENGINE SIZE         0
CYLINDERS           0
TRANSMISSION        0
FUEL                0
FUEL CONSUMPTION    0
HWY (L/100 km)      0
COMB (L/100 km)     0
COMB (mpg)          0
EMISSIONS           0
dtype: int64
```

```
#So, no missing value in this dataset
```

Labelling

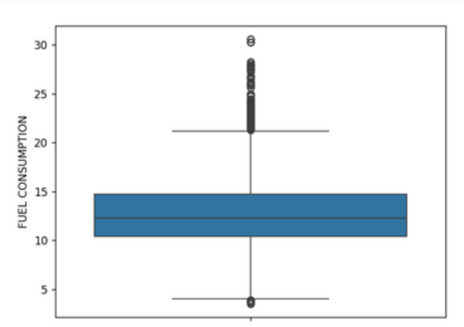
```
#We have object(5), so let's convert categorical columns into numerical ones.
#If we remove all categorical columns, the model will be constructed maybe not meaningful or we'll miss some important informations.
from sklearn.feature_selection import mutual_info_regression
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

#Because the 'EMISSIONS' is estimated by 'FUEL CONSUMPTION', so we cannot use 'EMISSION' to predict it.
df = df.drop("EMISSIONS", axis= 1)
df_encoded = df

df_encoded["MAKE"] = encoder.fit_transform(df["MAKE"])
df_encoded["MODEL"] = encoder.fit_transform(df["MODEL"])
df_encoded["VEHICLE CLASS"] = encoder.fit_transform(df["VEHICLE CLASS"])
df_encoded["TRANSMISSION"] = encoder.fit_transform(df["TRANSMISSION"])
df_encoded["FUEL"] = encoder.fit_transform(df["FUEL"])

#Extract the feature columns and target columns before modelling.
features = df_encoded.drop("FUEL CONSUMPTION", axis=1)
target = df_encoded["FUEL CONSUMPTION"]
print(df)
```

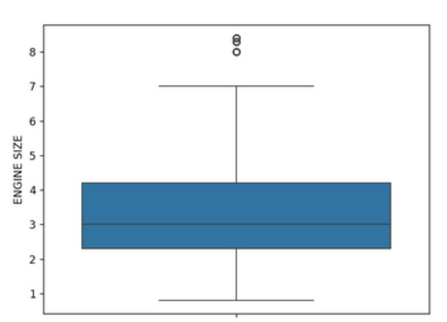
DATA PRE-PROCESSING

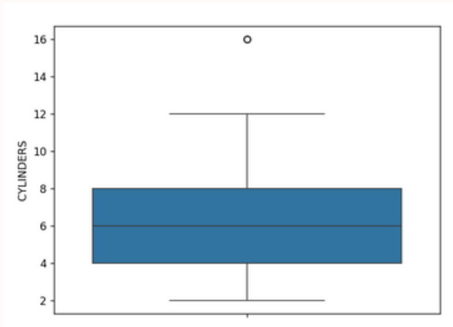
	YEAR	MAKE	MODEL	VEHICLE CLASS	ENGINE SIZE	CYLINDERS	TRANSMISSION	FUEL	FUEL CONSUMPTION	HWY (L/100 km)	COMB (L/100 km)	COMB (mpg)
0	2000	0	1	0	1.6	4	2	3	9.2	6.7	8.1	35
1	2000	0	1	0	1.6	4	27	3	8.5	6.5	7.6	37
2	2000	0	61	4	3.2	6	15	4	12.2	7.4	10.0	28
3	2000	0	62	4	3.5	6	2	4	13.4	9.2	11.5	25
4	2000	0	2173	17	1.8	4	2	3	10.0	7.0	8.6	33
...	...	...	...	...	...	...	...	...	...	...	...	...
22551	2022	86	4061	21	2.0	4	18	4	10.7	7.7	9.4	30
22552	2022	86	4067	21	2.0	4	18	4	10.5	8.1	9.4	30
22553	2022	86	4068	21	2.0	4	18	4	11.0	8.7	9.9	29
22554	2022	86	4088	22	2.0	4	18	4	11.5	8.4	10.1	28
22555	2022	86	4089	22	2.0	4	18	4	12.4	8.9	10.8	26

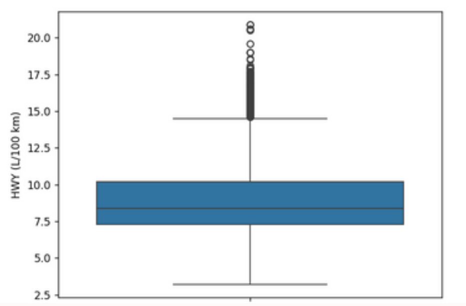
Checking outlier

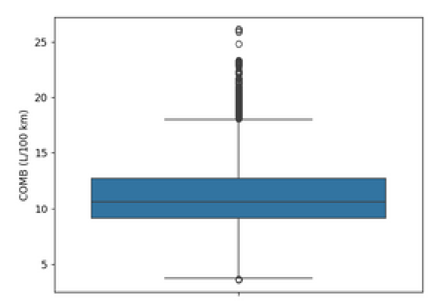
```
sns.boxplot(df['ENGINE SIZE'])
plt.show()
```











#We can see that the outliers appear consistently across plots of different variables, which may indicate that these values are a natural part of the data and not noise or errors. Therefore, we do not need to remove these outliers.

Descriptive Statistics

```
df.describe() #Summary of this dataframe
```

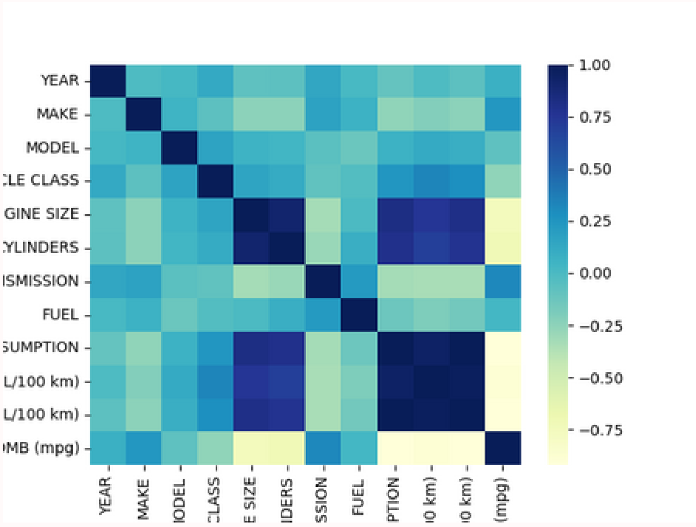
	YEAR	MAKE	MODEL	VEHICLE CLASS	ENGINE SIZE	...	FUEL	FUEL CONSUMPTION	HWY (L/100 km)	COMB (L/100 km)	COMB (mpg)
count	22556.000000	22556.000000	22556.000000	22556.000000	22556.000000	...	22556.000000	22556.000000	22556.000000	22556.000000	22556.000000
mean	2011.554442	40.392002	2101.669445	12.375067	3.356646	...	3.274827	12.763513	8.919126	11.034341	27.374534
std	6.298269	24.787685	1209.219227	8.892687	1.335425	...	0.808823	3.500999	2.274764	2.910920	7.376982
min	2000.000000	0.000000	0.000000	0.000000	0.800000	...	0.000000	3.500000	3.200000	3.600000	11.000000
25%	2006.000000	18.000000	1081.000000	4.000000	2.300000	...	3.000000	10.400000	7.300000	9.100000	22.000000
50%	2012.000000	34.000000	2056.000000	13.000000	3.000000	...	3.000000	12.300000	8.400000	10.600000	27.000000
75%	2017.000000	62.000000	3174.250000	18.000000	4.200000	...	4.000000	14.725000	10.200000	12.700000	31.000000
max	2022.000000	86.000000	4241.000000	31.000000	8.400000	...	4.000000	30.600000	20.900000	26.100000	78.000000

```
corr_matrix = df.corr(numeric_only=True) #Check correlation between all variables
print(corr_matrix)
```

DATA PRE-PROCESSING

	YEAR	MAKE	MODEL	VEHICLE CLASS	ENGINE SIZE	CYLINDERS	TRANSMISSION	FUEL	FUEL CONSUMPTION	HWY (L/100 km)	COMB (L/100 km)	COMB (mpg)
YEAR	1.000000	-0.010979	0.019102	0.122764	-0.077782	-0.072607	0.137136	0.014978	-0.098631	-0.007471	-0.068020	0.079989
MAKE	-0.010979	1.000000	0.050688	-0.068057	-0.242228	-0.243246	0.169319	0.066048	-0.256671	-0.211232	-0.244132	0.237249
MODEL	0.019102	0.050688	1.000000	0.163550	0.057520	0.035476	-0.064564	-0.121172	0.069073	0.117852	0.087085	-0.075378
VEHICLE CLASS	0.122764	-0.068057	0.163550	1.000000	0.156541	0.114055	-0.086159	-0.032294	0.247151	0.333855	0.280690	-0.260137
ENGINE SIZE	-0.077782	-0.242228	0.057520	0.156541	1.000000	0.913377	-0.324956	-0.005149	0.821605	0.749394	0.807316	-0.755002
CYLINDERS	-0.072607	-0.243246	0.035476	0.114055	0.913377	1.000000	-0.283964	0.086585	0.794943	0.698344	0.771587	-0.714215
TRANSMISSION	0.137136	0.169319	-0.064564	-0.086159	-0.324956	-0.283964	1.000000	0.226152	-0.327587	-0.342998	-0.337355	0.324409
FUEL	0.014978	0.066048	-0.121172	-0.032294	-0.005149	0.086585	0.226152	1.000000	-0.126156	-0.195315	-0.152111	0.030066
FUEL CONSUMPTION	-0.098631	-0.256671	0.069073	0.247151	0.821605	0.794943	-0.327587	-0.126156	1.000000	0.942351	0.992960	-0.921361
HWY (L/100 km)	-0.007471	-0.211232	0.117852	0.333855	0.749394	0.698344	-0.342998	-0.195315	0.942351	1.000000	0.975014	-0.884744
COMB (L/100 km)	-0.068020	-0.244132	0.087085	0.280690	0.807316	0.771587	-0.337355	-0.152111	0.992960	0.975014	1.000000	-0.920915
COMB (mpg)	0.079989	0.237249	-0.075378	-0.260137	-0.755002	-0.714215	0.324409	0.030066	-0.921361	-0.884744	-0.920915	1.000000

```
sns.heatmap(corr_matrix, cmap='YlGnBu')
plt.show()
```



The correlation analysis between 'FUEL CONSUMPTION' and other variables reveals several significant relationships. 'ENGINE SIZE' (0.821605) and 'CYLINDERS' (0.794943) show strong positive correlations with 'FUEL CONSUMPTION', indicating that larger engine sizes and more cylinders are associated with higher fuel consumption. 'VEHICLE CLASS' also has a positive correlation (0.247151), suggesting that the type of vehicle impacts fuel consumption.

Conversely, 'TRANSMISSION' (-0.327587) and 'MAKE' (-0.256671) have moderate negative correlations with 'FUEL CONSUMPTION', indicating that the type of transmission and the vehicle's make influence fuel efficiency. 'FUEL' (-0.126156) and 'YEAR' (-0.098631) show weaker negative correlations, suggesting minimal impact on fuel consumption.

Additionally, 'FUEL CONSUMPTION' is highly positively correlated with 'HWY (L/100 km)' (0.942351) and 'COMB (L/100 km)' (0.992960), highlighting consistent consumption patterns across different driving conditions. In contrast, 'COMB (mpg)' (-0.921361) has a strong negative correlation, reflecting the inverse relationship between fuel consumption measured in liters per 100 km and miles per gallon.

DATA PRE-PROCESSING

Splitting dataset

```
#Split data into training set and testing set (80:20)
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score

x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error

#Data Normalization before modelling
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
```

DATA MODELLING

Linear Regression

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
lr_pred = lr.predict(x_test)

print('Linear Regression:')
print('MSE:', mean_squared_error(y_test, lr_pred))
print('MAE:', mean_absolute_error(y_test, lr_pred))
print('R-squared:', r2_score(y_test, lr_pred))

#Output:
MSE: 0.006184353699414737
MAE: 0.055195326670586804
R-squared: 0.9994884320056183
```

XGBoost

```
import xgboost as xgb
dtrain = xgb.DMatrix(x_train, label=y_train)
dtest = xgb.DMatrix(x_test, label=y_test)

# Define the hyperparameters for the XGBoost model
params = {
   "objective": "reg:squarederror", "booster": "gbtree", "eta": 0.1, "max_depth": 5, "subsample": 0.8,
   "colsample_bytree": 0.8, "seed": 0
}
model = xgb.train(params, dtrain, num_boost_round=100)  #Train the XGBoost model

xgb_pred = model.predict(dtest)  #Predict the target values for the testing set

# Calculate the mean squared error between the actual and predicted target values
print('MSE:', mean_squared_error(y_test, xgb_pred))
print('MAE:', mean_absolute_error(y_test, xgb_pred))
print('R-squared:', r2_score(y_test, xgb_pred))

#Output:
MSE: 0.014515802042705549
MAE: 0.08995168886497514
R-squared: 0.9987992569476531
```

DATA MODELLING

GBM

```
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor().fit(X_train,y_train)
gbm_pred = gbr.predict(X_test)

print('MSE:', mean_squared_error(y_test, gbm_pred))
print('MAE:', mean_absolute_error(y_test, gbm_pred))
print('R-squared:', r2_score(y_test, gbm_pred))
#Output:
MSE: 0.04801514726617903
MAE: 0.16440473136625944
R-squared: 0.9960282005556662
```

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor().fit(X_train,y_train)
dt_pred = dt.predict(X_test)

print('MSE:', mean_squared_error(y_test, dt_pred))
print('MAE:', mean_absolute_error(y_test, dt_pred))
print('R-squared:', r2_score(y_test, dt_pred))
#Output:
MSE: 0.008211436170212772
MAE: 0.035660460992908254
R-squared: 0.9993207523151553
```

CatBoost

```
from catboost import CatBoostRegressor
cat = CatBoostRegressor(verbose=False).fit(X_train,y_train)
cat_pred = cat.predict(X_test)

print('MSE:', mean_squared_error(y_test, cat_pred))
print('MAE:', mean_absolute_error(y_test, cat_pred))
print('R-squared:', r2_score(y_test, cat_pred))
#Output:
MSE: 0.008873737484271676
MAE: 0.06535248330071486
R-squared: 0.9992964841919492
```

DATA MODELLING

Stopping condition

- Linear Regression: No specific stopping condition is defined, other than the model completing the training process.
- Gradient Boosting, Decision Tree, CatBoost: These models use default parameters and have no specific stopping condition other than the model completing the training process.
- XGBoost: Uses num_boost_round=100 to define the maximum number of iterations for the training process.

DATA EVALUATION

Evaluating model

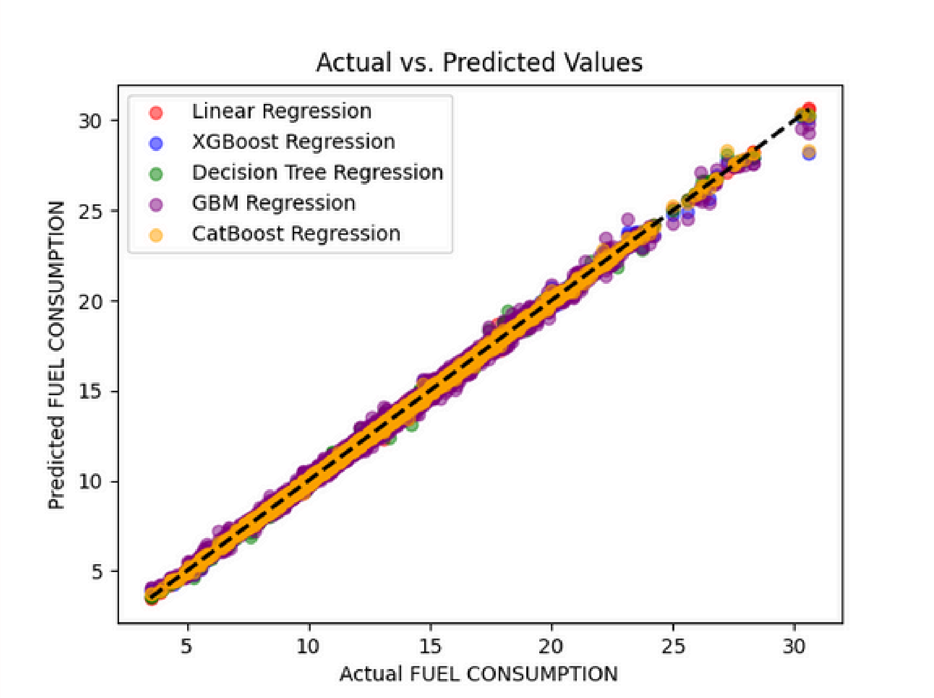
	MSE	MAE	R-squared
Linear Regression	0.00515	0.05431	0.99959
XGBoost	0.01554	0.08986	0.99877
GBM	0.04902	0.16630	0.99611
Decision Tree	0.00839	0.03651	0.99933
CatBoost	0.00887	0.06535	0.99930

Linear Regression and Decision Tree tend to produce better results with R-squared values close to 1 and lower MSE/MAE compared to other models. XGBoost and CatBoost also achieve promising results with high accuracy and low error rates, while GBM performs less favorably compared to the other models.

Predicting

```
# Plot Actual vs Predicted
plt.scatter(y_test, lr_pred, color='red', alpha=0.5, label='Linear Regression')
plt.scatter(y_test, xgb_pred, color='blue', alpha=0.5, label='XGBoost Regression')
plt.scatter(y_test, dt_pred, color='green', alpha=0.5, label='Decision Tree Regression')
plt.scatter(y_test, gbm_pred, color='purple', alpha=0.5, label='GBM Regression')
plt.scatter(y_test, cat_pred, color='orange', alpha=0.5, label='CatBoost Regression')
plt.plot([target.min(), target.max()], [target.min(), target.max()], 'k--', lw=2)

plt.xlabel('Actual FUEL CONSUMPTION')
plt.ylabel('Predicted FUEL CONSUMPTION')
plt.title('Actual vs. Predicted Values')

plt.legend()
plt.show()
```

DATA EVALUATION



Based on the model evaluation with test data, as shown in the chart, there are no signs of overfitting or underfitting. All four models—Linear Regression, Decision Tree, XGBoost, and CatBoost—perform well. Therefore, these models accurately predict Fuel Consumption based on vehicle characteristics.

APPLICATION

Therefore, these models can be used to predict the fuel consumption of new vehicle models based on parameters such as engine size, number of cylinders, fuel type, etc. Additionally, Unsupervised Learning Models or clustering algorithms can be used to group vehicles based on characteristics like engine size, number of cylinders, fuel type, etc. Vehicles within similar groups may exhibit similar fuel consumption patterns.

Consequently, these models can assist vehicle manufacturers and related organizations in predicting and optimizing fuel consumption, thereby reducing costs and environmental impact; supporting consumers in making purchasing decisions; and analyzing and forecasting fuel consumption trends and related factors in the future, thereby informing appropriate business and management strategies.

Applying Linear Regression model to predict 'FUEL CONSUMPTION' in 2023

Source dataset: https://www.kaggle.com/datasets/imtkaggleteam/fuel-concumption-ratings-2023/data

```
#Predict new dataset
new_df = pd.read_csv('Fuel_Consumption_Ratings_2023.csv', encoding='latin1')
new_df = new_df.dropna()
new_df = new_df.rename(columns={
   'Year': 'YEAR',
   'Make': 'MAKE',
   'Model': 'MODEL',
   'Vehicle Class': 'VEHICLE CLASS',
   'Engine Size (L)': 'ENGINE SIZE',
   'Cylinders': 'CYLINDERS',
   'Transmission': 'TRANSMISSION',
   'Fuel Type': 'FUEL',
   'Fuel Consumption (L/100Km)': 'FUEL CONSUMPTION',
   'Hwy (L/100 km)':'HWY (L/100 km)',
   'Comb (L/100 km)':'COMB (L/100 km)',
   'Comb (mpg)':'COMB (mpg)'
})
print(new_df)
```

APPLICATION

```
#Labelling
new_df_encoded = new_df

new_df_encoded["MAKE"] = encoder.fit_transform(new_df["MAKE"])
new_df_encoded["MODEL"] = encoder.fit_transform(new_df["MODEL"])
new_df_encoded["VEHICLE CLASS"] = encoder.fit_transform(new_df["VEHICLE CLASS"])
new_df_encoded["TRANSMISSION"] = encoder.fit_transform(new_df["TRANSMISSION"])
new_df_encoded["FUEL"] = encoder.fit_transform(new_df["FUEL"])

new_features = new_df[['YEAR', 'MAKE', 'MODEL', 'VEHICLE CLASS', 'ENGINE SIZE', 'CYLINDERS',
'TRANSMISSION', 'FUEL','HWY (L/100 km)','COMB (L/100 km)', 'COMB (mpg)']]
new_predict = lr.predict(new_features)
new_target = new_df['FUEL CONSUMPTION']
print('MSE:', mean_squared_error(new_target, new_predict))
print('MAE:', mean_absolute_error(new_target, new_predict))
print('R-squared:', r2_score(new_target, new_predict))
```

Result

```
>>> print('MSE:', mean_squared_error(new_target, new_predict))
MSE: 0.004548007545428 53
>>> print('MAE:', mean_absolute_error(new_target, new_predict))
MAE: 0.05231772096637514 5
>>> print('R-squared:', r2_score(new_target, new_predict))
R-squared: 0.9996188049424067
```

It can be said that the Linear Regression model has predicted 'FUEL CONSUMPTION' in 2023 very well (R-squared ~ 1). Therefore, we can prove that the applicability of this model is very high, and it can support consumers and manufacturers in the future.

CONCLUSION

The models have proven to be highly effective in accurately predicting vehicle fuel consumption for the year 2023, as evidenced by our evaluation metrics. Its remarkable accuracy and reliability make it an invaluable tool for various real-world applications.

The applicability of this model in practical scenarios cannot be overstated. It can greatly assist consumers in making well-informed decisions regarding vehicle purchases based on projected fuel consumption. Furthermore, manufacturers can utilize this model to optimize their vehicle designs for enhanced fuel efficiency, resulting in cost reduction and a minimized environmental impact. Additionally, the model can be employed to forecast fuel consumption trends, enabling businesses and policymakers to develop effective strategies for the future.

There are several advantages associated with the models. Firstly, its simplicity makes it easily implementable and interpretable, rendering it accessible to a wide range of users. Secondly, the model exhibits computational efficiency, allowing for swift predictions even when dealing with large datasets. Lastly, the transparency of it is noteworthy, as the coefficients provide clear insights into the relationship between vehicle characteristics and fuel consumption, aiding in the understanding of their impact.

However, it is important to acknowledge the disadvantages of this models. One notable limitation is its vulnerability to outliers, which have the potential to distort predictions and affect the overall accuracy of the model.