2024

Fuel Efficiency Prediction

Capstone Project - Regression

DIBOSH BARUAH

1. Introduction

The goal of this project is to develop a machine learning model that predicts the fuel consumption of vehicles based on various features, such as the vehicle's mass, engine capacity, fuel type, emissions, energy consumption, and other relevant characteristics. This model can help manufacturers and researchers understand the factors influencing fuel efficiency and optimize vehicle designs for better performance. By leveraging historical data from a large dataset, this project aims to predict fuel consumption, which can be a valuable tool in improving vehicle fuel efficiency, reducing emissions, and enhancing overall environmental sustainability.

2. Data Collection

The dataset used for this project consists of **1 million records**, which includes various features related to automobiles. Each record represents a vehicle with different specifications, including its mass, fuel type, energy consumption, and fuel consumption values. The data includes the following attributes:

- **Mass (m)**: The weight of the vehicle (in kilograms).
- **Engine capacity (ec)**: The engine displacement of the vehicle (in cubic centimeters).
- **Fuel type (Ft)**: The type of fuel used by the vehicle (e.g., petrol, electric, diesel, etc.).
- **Energy consumption (z)**: The energy consumption of the vehicle (in Wh/km).
- **Fuel consumption**: The target variable representing the fuel consumption of the vehicle (in liters per 100 kilometers).
- **Other features**: Several other characteristics such as engine power, emissions, electric range (for electric vehicles), and more.

The dataset is split into three portions:

- **Training set**: The first 700,000 records used to train the model.
- **Evaluation set**: The next 200,000 records used to evaluate the model's performance.
- **Production set**: The remaining records are used for future predictions once the model is deployed in production.

3. Data Preprocessing

To make the data suitable for model training, several preprocessing steps were applied:

3.1 Handling Missing Values

The dataset was checked for missing values in various columns. Missing values in the numerical columns (e.g., mass, engine capacity, fuel consumption) were imputed using the **median** of the respective columns. For categorical features like fuel type and manufacturer, missing values were filled using the **mode** (the most common category). This ensures that the data is complete and ready for machine learning.

3.2 Text Cleaning

Some features, particularly categorical ones (e.g., fuel type), might have extraneous characters or spaces. These were removed by stripping leading/trailing spaces from column names and ensuring consistency in feature naming.

3.3 Feature Engineering

Several columns with too many missing values or irrelevant information were dropped to reduce complexity. Columns like z (Wh/km) and Erwltp (g/km) had too many missing values, so they were discarded from the dataset. Additionally, features that don't contribute significantly to predicting fuel consumption, such as the r column, were also dropped.

3.4 One-Hot Encoding

Categorical columns like **fuel type** and **manufacturer** were one-hot encoded. One-hot encoding converts categorical features into a format that can be fed into machine learning algorithms by creating binary variables for each possible category. This step ensures that the machine learning model can understand and use categorical data effectively.

3.5 Feature and Target Separation

The dataset was split into features (X) and the target variable (y), where the target is **fuel consumption**, and the features include other attributes like mass, engine capacity, fuel type, and energy consumption.

4. Model Building

The primary objective of this project is to predict fuel consumption using a machine learning model. A **Random Forest Regressor** was chosen for this task, as it is a powerful ensemble learning method that performs well with both categorical and numerical data.

4.1 Model Training

The Random Forest Regressor was trained using the **training set** of 700,000 records. Random Forest works by creating multiple decision trees, each trained on a random subset of the data. The predictions from these trees are aggregated to give the final output, which in this case is the predicted fuel consumption of the vehicle.

4.2 Model Evaluation

After training the model, it was evaluated on the **evaluation set** of 200,000 records to determine its performance. The following evaluation metrics were used:

- **RMSE (Root Mean Squared Error)**: This metric measures the average magnitude of errors in the predictions, giving more weight to larger errors.
- **R^2 (Coefficient of Determination)**: This value indicates how well the model explains the variance in fuel consumption. A high R^2 indicates that the model performs well.
- **MAE (Mean Absolute Error)**: This metric measures the average absolute errors in the predictions, indicating how far the model's predictions are from the true values.
- **MSLE (Mean Squared Logarithmic Error)**: This metric is useful when the target variable has a wide range or exponential growth. It penalizes large differences in predictions, especially when they are large values.
- **Explained Variance Score**: This metric indicates the proportion of the variance in fuel consumption that is explained by the model.

5. Model Evaluation Results

The model performed exceptionally well on the evaluation set, producing the following key results:

- **RMSE (Root Mean Squared Error)**: 0.1893
  This indicates that, on average, the model's predictions deviate by around **0.1893 units of fuel consumption**, which is very low.
- **R^2 (Coefficient of Determination)**: 0.989
  This indicates that **98.9%** of the variance in fuel consumption can be explained by the model, which is an excellent performance.
- **MAE (Mean Absolute Error)**: 0.0496
  The **average error** is very small, indicating that the model's predictions are very close to the actual values.
- **MSLE (Mean Squared Logarithmic Error)**: 0.0011
  This very low value indicates that the model performs well even when predicting fuel consumption values with a wide range.

- **Explained Variance Score**: 0.989
  This confirms that the model captures almost all of the variability in fuel consumption.

These results demonstrate that the Random Forest Regressor model is highly accurate and effective in predicting fuel consumption based on the features in the dataset.

6. Results & Discussion

6.1 Model Insights

The model successfully predicts fuel consumption with a high degree of accuracy. The high R^2 value suggests that the model is able to capture the relationship between various features (such as vehicle mass, engine capacity, fuel type, etc.) and fuel consumption. The relatively low RMSE and MAE indicate that the model's predictions are quite close to the actual values.

6.2 Feature Importance

One of the advantages of using a Random Forest model is that it allows us to assess the importance of each feature in predicting the target variable. Features such as **vehicle mass**, **engine capacity**, and **fuel type** were found to be the most important predictors of fuel consumption. This makes sense, as larger, more powerful vehicles tend to consume more fuel. The **energy consumption (z)** feature was also important, especially for electric vehicles, highlighting the significance of energy efficiency in fuel consumption prediction.

6.3 Limitations

While the model performs well overall, it is important to acknowledge a few limitations:

- The dataset may not fully account for all factors influencing fuel consumption, such as driving behavior, road conditions, or maintenance status of the vehicles.
- There may be potential biases in the data that affect the model’s performance, such as an overrepresentation of certain vehicle types or fuel types.

6.4 Future Improvements

To further improve the model, additional features such as driving conditions, vehicle age, and detailed maintenance history could be included. Moreover, employing more advanced algorithms like **Gradient Boosting** or **XGBoost** might help achieve even better predictive performance.

7. Conclusion

This project successfully developed a machine learning model to predict fuel consumption based on vehicle characteristics. The Random Forest Regressor model achieved high accuracy, with an **R^2 of 0.989**, indicating that it effectively captures the factors influencing fuel consumption. The evaluation metrics, including RMSE, MAE, and MSLE, all demonstrate the model's excellent performance. By identifying key features contributing to fuel consumption, this model can provide valuable insights for manufacturers and researchers working to optimize vehicle fuel efficiency and reduce environmental impact.

8.Original Code:

```
import sqlite3
import pandas as pd

# Connecting to the database
conn = sqlite3.connect('/Users/diboshbaruah/Desktop/Database.db')
data = pd.read_sql_query('SELECT * FROM Automobile_data', conn)

print("Dataset successfully loaded...\n")

# Displaying the first few rows to inspect the data
print("Displaying first few rows of the dataset:\n")
print(data.head())

# Checking data types before conversion
print("\nData types before conversion:")
print(data.dtypes)

# Closing the connection
conn.close()
```

```
Dataset successfully loaded...
```

```
Displaying first few rows of the dataset:

   r  m (kg)      Mt  Ewltp (g/km)        Ft Fm  ec (cm3)  ep (KW)  z (Wh/km)
\
0  1  1262.0  1352.0         133.0    petrol  M     999.0     84.0        NaN
1  1  2434.0  2559.0           0.0  electric  E       NaN    300.0      193.0
2  1  1984.0  2095.0           0.0  electric  E       NaN    220.0      157.0
3  1  1314.0  1386.0         149.0    petrol  M    1498.0     78.0        NaN
4  1  1075.0  1145.0         119.0       lpg  B    1242.0     51.0        NaN

   Erwltp (g/km)  Fuel consumption   Electric range (km)
0           1.56                5.8                  NaN
1            NaN                NaN                445.0
2            NaN                NaN                455.0
3            NaN                6.6                  NaN
4            NaN                6.0                  NaN

Data types before conversion:
r                        int64
m (kg)                 float64
Mt                     float64
Ewltp (g/km)           float64
Ft                      object
Fm                      object
ec (cm3)               float64
ep (KW)                float64
z (Wh/km)              float64
Erwltp (g/km)          float64
Fuel consumption       float64
Electric range (km)    float64
dtype: object
```

```python
#Checking for missing values in the dataset
print("\nChecking for missing values:")
print(data.isnull().sum())
```

```
Checking for missing values:
r                           0
m (kg)                     39
Mt                      48308
Ewltp (g/km)             1969
Ft                          0
Fm                          0
ec (cm3)               155677
ep (KW)                  3304
z (Wh/km)              780824
Erwltp (g/km)          462075
Fuel consumption       296752
Electric range (km)    783035
dtype: int64
```

```
import pandas as pd
from sklearn.impute import SimpleImputer

# Stripping leading/trailing spaces from column names
data.columns = data.columns.str.strip()

# Handling Missing Values for Categorical Features
# Impute missing values for categorical features using the mode
data['Ft'] = data['Ft'].fillna(data['Ft'].mode()[0])
data['Fm'] = data['Fm'].fillna(data['Fm'].mode()[0])

# Removing rows where the target variable 'Fuel consumption' is missing
data = data[data['Fuel consumption'].notnull()]

# Imputing Missing Values for Numerical Features
# Defining the list of numerical columns to impute
numerical_columns = ['m (kg)', 'Mt', 'Ewltp (g/km)', 'ec (cm3)', 'ep (KW)',
'z (Wh/km)', 'Erwltp (g/km)']

# Creating an imputer for numerical columns, using median strategy
imputer = SimpleImputer(strategy='median')

# Imputing missing values for numerical columns using .loc[]
existing_numerical_columns = [col for col in numerical_columns if col in
data.columns]
data.loc[:, existing_numerical_columns] =
imputer.fit_transform(data[existing_numerical_columns])

# Droping columns with too many missing values
columns_to_drop = ['z (Wh/km)', 'Erwltp (g/km)']
columns_to_drop = [col for col in columns_to_drop if col in data.columns]
data = data.drop(columns=columns_to_drop)

# Imputing missing `Electric range (km)` based on fuel type
# Setting `Electric range (km)` to 0 for non-electric vehicles
data['Electric range (km)'] = data.apply(lambda row: 0 if row['Ft'] !=
'electric' else row['Electric range (km)'], axis=1)

# Imputing missing values for electric vehicles
data['Electric range (km)'] = data['Electric range
(km)'].fillna(data['Electric range (km)'].median())
```

```
# Checking if there are any missing values after imputation
print("\nMissing values after imputation:")
print(data.isnull().sum())
```

```
Missing values after imputation:
r                       0
m (kg)                  0
Mt                      0
Ewltp (g/km)            0
Ft                      0
Fm                      0
ec (cm3)                0
ep (KW)                 0
Fuel consumption        0
Electric range (km)     0
dtype: int64
```

```
# Dropping features that may not be useful for the model (e.g., 'r' and 'Mt')
data = data.drop(columns=['r', 'Mt'])

# Performing one-hot encoding on categorical columns
data_encoded = pd.get_dummies(data, columns=['Ft', 'Fm'], drop_first=True)  #
drop_first=True avoids multicollinearity

# Checking the columns after encoding
print(data_encoded.columns)
```

```
Index(['m (kg)', 'Ewltp (g/km)', 'ec (cm3)', 'ep (KW)', 'Fuel consumption',
       'Electric range (km)', 'Ft_PETROL', 'Ft_PETROL/ELECTRIC', 'Ft_diesel',
       'Ft_diesel/electric', 'Ft_e85', 'Ft_electric', 'Ft_hydrogen',
'Ft_lpg',
       'Ft_ng', 'Ft_petrol', 'Ft_petrol/electric', 'Fm_E', 'Fm_F', 'Fm_H',
       'Fm_M', 'Fm_P'],
      dtype='object')
```

```
# Splitting the dataset into training, evaluation, and production sets
train_data = data_encoded[:700000]
eval_data = data_encoded[700000:900000]
prod_data = data_encoded[900000:]

# Defining the features (X) and target (y)
X_train = train_data.drop(columns=['Fuel consumption', 'Electric range
(km)'])
y_train = train_data['Fuel consumption']

X_eval = eval_data.drop(columns=['Fuel consumption', 'Electric range (km)'])
y_eval = eval_data['Fuel consumption']

X_prod = prod_data.drop(columns=['Fuel consumption', 'Electric range (km)'])
y_prod = prod_data['Fuel consumption']
```

```python
# Printing the shapes of the splits
print("Training Data Shape:", train_data.shape)
print("Evaluation Data Shape:", eval_data.shape)
print("Production Data Shape:", prod_data.shape)
```

```
Training Data Shape: (700000, 22)
Evaluation Data Shape: (3248, 22)
Production Data Shape: (0, 22)
```

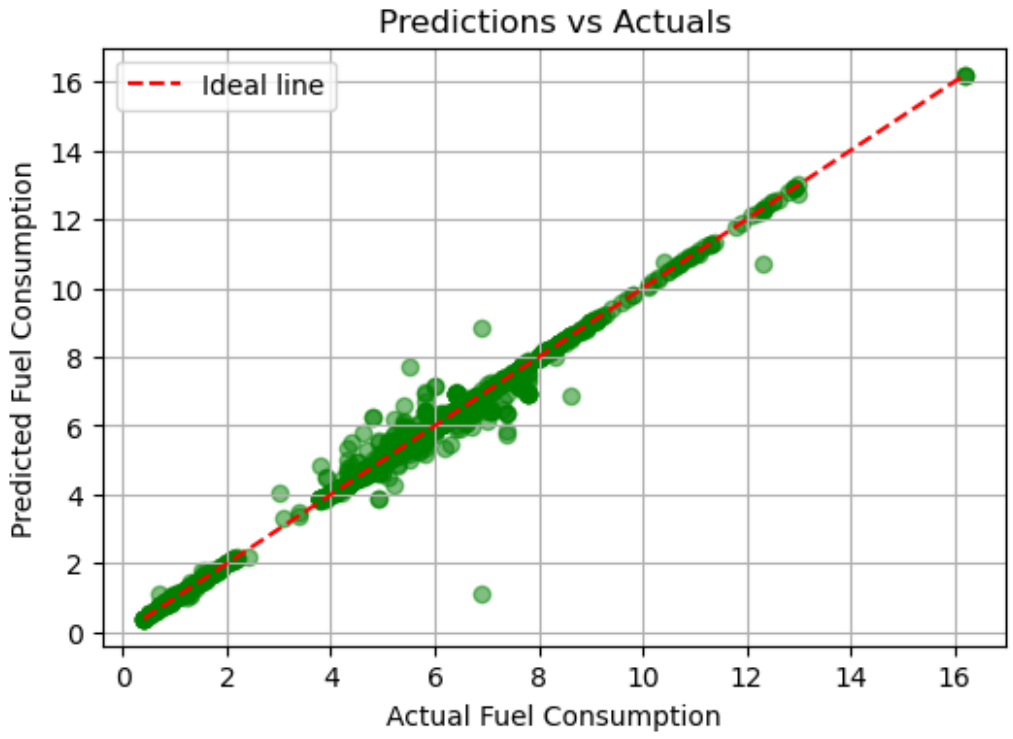
```python
from sklearn.ensemble import RandomForestRegressor

# Initializing and training the RandomForestRegressor model
model = RandomForestRegressor(n_estimators=20, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=20, random_state=42)
```

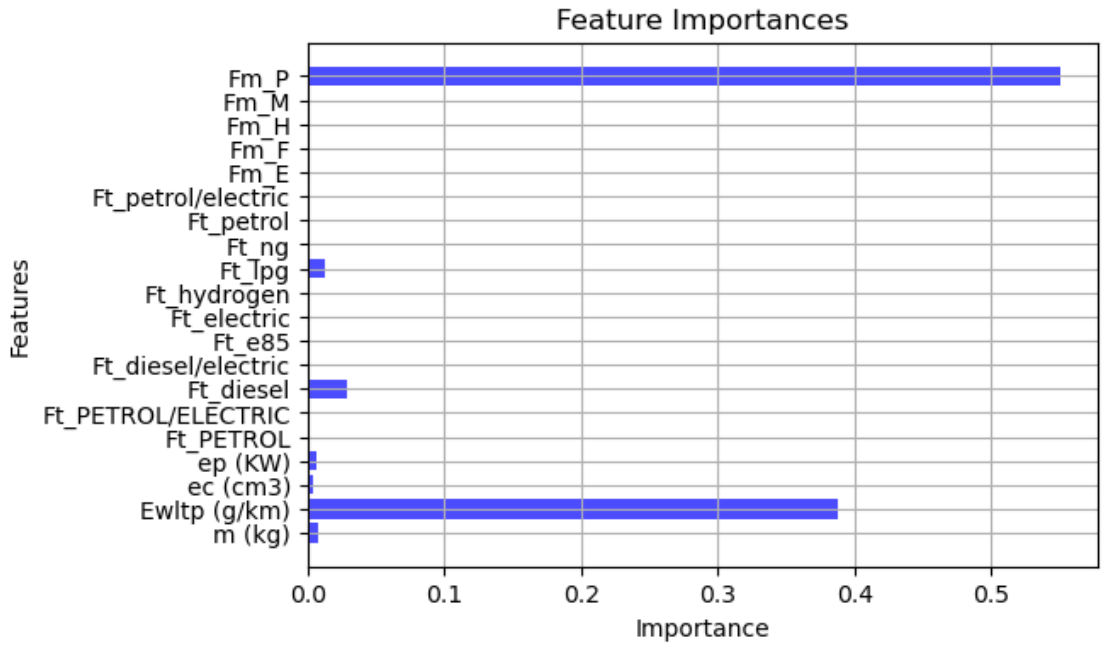
```python
from sklearn.metrics import mean_squared_error, r2_score,
mean_absolute_error, explained_variance_score, mean_squared_log_error

# Predicting fuel consumption on the evaluation set
y_pred = model.predict(X_eval)

# Computing RMSE
rmse = mean_squared_error(y_eval, y_pred) ** 0.5

# Calculating other evaluation metrics
mae = mean_absolute_error(y_eval, y_pred)
msle = mean_squared_log_error(y_eval, y_pred)
evs = explained_variance_score(y_eval, y_pred)

# Evaluating the model's R² performance
r2 = r2_score(y_eval, y_pred)

# Printing all evaluation metrics
print(f"RMSE (Root Mean Squared Error): {rmse}")
print()
print(f"R² (Coefficient of Determination): {r2}")
print()
print(f"MAE (Mean Absolute Error): {mae}")
print()
print(f"MSLE (Mean Squared Logarithmic Error): {msle}")
print()
print(f"Explained Variance Score: {evs}")
```

```
RMSE (Root Mean Squared Error): 0.18932240011494722

R² (Coefficient of Determination): 0.9893529472262357

MAE (Mean Absolute Error): 0.049580650031978715

MSLE (Mean Squared Logarithmic Error): 0.0010879482953989011

Explained Variance Score: 0.9893599018449503
```

```python
import matplotlib.pyplot as plt

# Plotting predictions vs actual values
plt.figure(figsize=(6, 4))
plt.scatter(y_eval, y_pred, color='green', alpha=0.5)
plt.plot([min(y_eval), max(y_eval)], [min(y_eval), max(y_eval)], color='red',
linestyle='--', label='Ideal line')
plt.title('Predictions vs Actuals')
plt.xlabel('Actual Fuel Consumption')
plt.ylabel('Predicted Fuel Consumption')
plt.legend()
plt.grid(True)
plt.show()
```



```
# Plot feature importances
importances = model.feature_importances_
features = X_train.columns

plt.figure(figsize=(6, 4))
plt.barh(features, importances, color='blue', alpha=0.7)
plt.title('Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.grid(True)
plt.show()
```



** Now running the saved scripts on jupyter notebook - train_model.py // predict_fraud.py // app.py ***

```
# Importing the training script
!python train_model.py
```

```
Dataset successfully loaded...


Initiating RandomForestRegressor....

Model training complete and saved.
```

```
# Importing the Predict script
!python predict_model.py
```

```
Data loading successful... initiating prediction

Predictions:
[5.8 6.6 6.  5.5 4.3 2.  1.1 4.8 5.4 5.4]
```

```
import subprocess

# Now running the Flask app using subprocess
subprocess.Popen(["python", "app.py"])
```

```
<Popen: returncode: None args: ['python', 'app.py']>
```

```
import requests

data = {
    "Ft": ["electric"],
    "Fm": ["manual"],
    "m (kg)": [1500],
    "Mt": [3],
    "Ewltp (g/km)": [120],
    "ec (cm3)": [2000],
    "ep (KW)": [100],
    "Electric range (km)": [200]
}

response = requests.post("http://127.0.0.1:5000/predict", json=data)

print(response.json())  # This will print the prediction result
```

```
[6.014975380226921]

127.0.0.1 - - [30/Dec/2024 11:30:52] "POST /predict HTTP/1.1" 200 -
```