CO_2 Emission Prediction Using Machine Learning

1. Introduction

This project aims to analyze and predict **CO_2 emissions of vehicles in Canada** using machine learning techniques. The dataset used "CO2 Emissions_Canada.csv" contains various automobile attributes such as engine size, cylinders, and fuel consumption. The study involves data preprocessing, outlier handling, feature encoding, and model evaluation using multiple regression algorithms.

---

2. Data Loading and Exploration

The dataset was loaded using **pandas**:

```
df = pd.read_csv("CO2 Emissions_Canada.csv")
```

Initial Analysis:

- Dataset dimensions were displayed using df.shape.
- Column names and sample records were inspected with df.columns.tolist() and df.head().
- A subset of relevant columns was selected for analysis:
    - Make
    - Engine Size (L)
    - Cylinders
    - Fuel Consumption Comb (L/100 km)
    - CO2 Emissions (g/km)

This selection focuses on numerical and manufacturer-related features most relevant to emission prediction.

---

3. Data Cleaning and Preparation

Duplicate records were identified and removed:

```
df.duplicated().sum()
```

```
df.drop_duplicates(inplace=True)
```

Columns were renamed to eliminate spaces and special characters for easier reference:

df.columns = [c.strip().replace(' ', '_').replace('(', '').replace(')', '').replace('/', '_').replace('__','_') for c in df.columns]

Missing values were checked using:

df.isna().sum()

No missing values were found.

---

4. Outlier Detection and Treatment

Outliers in continuous features can distort model accuracy. Therefore, the **Interquartile Range (IQR) method** was used to identify and replace outliers.

For each feature (Engine_SizeL, Cylinders, Fuel_Consumption_Comb_L_100_km):

- Calculate **Q1**, **Q3**, and **IQR**.
- Define lower and upper bounds as:
- lower_bound = Q1 - 1.5 * IQR
- upper_bound = Q3 + 1.5 * IQR
- Count and replace outliers with the column mean:
- df.loc[df[col] < lower_bound, col] = mean_value
- df.loc[df[col] > upper_bound, col] = mean_value

The outlier summary was stored in a new DataFrame outlier_df, confirming successful replacement.

---

5. Feature Encoding

The categorical feature **'Make'** was encoded using LabelEncoder to convert company names into numeric representations:

le = LabelEncoder()

df['Make_encoded'] = le.fit_transform(df['Make'])

A mapping of encoded values was displayed:

Company Digit_Assigned

Company	**Digit_Assigned**
Audi	3
...	...

---

6. Feature Selection and Target Variable

The model used the following features:

['Make_encoded', 'Engine_SizeL', 'Cylinders', 'Fuel_Consumption_Comb_L_100_km']

The **target variable** was:

'CO2_Emissionsg_km'

---

7. Exploratory Data Analysis (EDA)

A distribution plot of CO_2 emissions was created using seaborn:

sns.histplot(df[target], bins=30, kde=True)

The data showed a near-normal distribution, indicating suitability for regression-based modeling.

---

8. Train-Test Split

Data was split into training and testing sets (80/20 ratio):

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

---

9. Model Development

a) Linear Regression

A baseline regression model was trained:

model = LinearRegression()

model.fit(X_train, y_train)

Evaluation:

- R^2 Score: measures how well predictions approximate actual values.
- RMSE: measures prediction error magnitude.

Results:

R^2 Score: ~0.77

RMSE: ~28

A scatter plot between actual and predicted emissions showed a strong positive correlation.

---

b) Random Forest Regressor

An ensemble model was implemented to improve performance:

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)

rf_model.fit(X_train, y_train)
```

Results:

R^2 Score: ~0.97

RMSE: ~9

The **Random Forest** model significantly outperformed Linear Regression due to its ability to handle nonlinearities and interactions.

---

c) Support Vector Regression (SVR)

To explore non-linear modeling, SVR with an RBF kernel was trained:

```
svr = SVR(kernel='rbf', C=1000.0, epsilon=0.2, gamma='auto')
```

Before fitting, the features were scaled:

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

Results:

R^2 Score: ~0.87

RMSE: ~20

SVR captured nonlinear relationships but required careful parameter tuning for optimal performance.

---

10. Model Comparison

A bar plot was created to visually compare model performance:

RMSE Comparison

- Linear Regression: highest error
- Random Forest: lowest error
- SVR: moderate error

R^2 Comparison

- Random Forest achieved the highest R^2 (~0.97), indicating the best fit.

These plots provided clear evidence of Random Forest's superior predictive accuracy.

---

11. Model Evaluation and Visualization

Visual comparisons of actual vs. predicted emissions were plotted for each model.
Random Forest predictions aligned most closely with the ideal (diagonal) line, confirming minimal deviation from true values.

---

12. Prediction on New Data

A new input sample was tested to predict CO_2 emissions:

```
new_data = pd.DataFrame([{

  'Make_encoded': 3,

  'Engine_SizeL': 2,

  'Cylinders': 4,

  'Fuel_Consumption_Comb_L_100_km': 8.8

}])
```

Predicted Emissions:

Model	**Predicted CO_2 (g/km)**
Linear Regression	≈ 204 g/km
Random Forest	≈ 205 g/km
SVR	≈ 205 g/km

The Random Forest prediction was closest to the actual emission value.

---

13. Conclusion

This project demonstrated an end-to-end machine learning pipeline for predicting CO_2 emissions using real-world automotive data.

Key outcomes:

- **Outlier handling** improved model stability.
- **Random Forest** achieved the best performance with the lowest RMSE and highest R^2.
- The model can effectively estimate CO_2 emissions based on key vehicle parameters.