Tanisha Jawale -UCE2022525

Srushti Kage-UCE2022531

Problem Statement: Prediction of CO2 Emissions from Vehicles

INTRODUCTIONS:

As concerns about environmental sustainability continue to grow, there is a pressing need to mitigate greenhouse gas emissions, particularly from transportation sources. Among these emissions, carbon dioxide (CO2) from vehicles contributes significantly to air pollution and climate change. Therefore, accurately predicting CO2 emissions from vehicles is crucial for policymakers, manufacturers, and consumers to implement effective measures to reduce environmental impact.

OBJECTIVE:

The goal of this project is to develop a predictive model that can accurately estimate CO2 emissions from vehicles based on various features such as vehicle type, engine specifications, fuel type, and mileage. By leveraging machine learning techniques, we aim to create models that not only predict CO2 emissions with high accuracy but also provide insights into the factors influencing emissions.

APPROACH:

Data Collection and Exploration:

- Gather a comprehensive dataset containing information on vehicle attributes and corresponding CO2 emissions.
- Explore the dataset to understand its structure, identify potential challenges (e.g., missing data, outliers), and gain insights into the relationships between features and CO2 emissions.

Data Preprocessing:

- Handle missing values, outliers, and any inconsistencies in the data.
- Encode categorical variables and normalize numerical features to prepare the data for modeling.

Feature Selection/Reduction:

- Employ feature selection techniques to identify the most relevant features for predicting CO2 emissions.
- Utilize dimensionality reduction methods if necessary to reduce the complexity of the dataset while preserving important information.

Model Selection and Training:

- Select a diverse set of regression models including Multiple Linear Regression, Lasso Regression, Decision Tree Regression, Ridge Regression, and Random Forest Regression.
- Train each model using the preprocessed data and evaluate their performance using appropriate metrics.

Model Evaluation and Interpretation:

- Assess the predictive performance of each model using metrics such as Mean Squared Error, R-squared, and Mean Absolute Error.
- Interpret the coefficients or feature importances of the models to understand the impact of different factors on CO2 emissions.
-

Prediction and Visualization:

- Use the trained models to make predictions on new or unseen data.
- Visualize the predicted CO2 emissions alongside actual values to assess the accuracy of the models and identify any patterns or trends.

CODE:

```
# Importing data & Checking for NULL Values

import pandas as pd
import numpy as np
```

```
dF=pd.read_csv('/content/CO2 Emissions_Canada.csv')
missing_vals=dF.isnull().sum()
print(missing_vals)
#checking for distinct company names

dF['Make'].unique()
#checking for distinct model names

dF['Model'].unique()
import numpy as np

# Putting different transmission sub-catagories into their respective
catagories

dF['Transmission']=np.where(dF['Transmission'].isin(['A4','A5','A6','A7','
A8','A9','A10']),'Automatic',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['AV','AV7','AV8','AV6
','AV10']),'Autonomous',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['AS4','AS5','AS6','AS
8','AS7','AS9','AS10']),'Automatic Selection',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['M6','M5','M7']),'Man
ual',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['AM5','AM6','AM7','AM
8','AM9','AM10']),'Automated Manual',dF['Transmission'])
print(dF['Transmission'].unique())
print()
print(dF['Transmission'].value_counts())


#finding correlation
print(dF.dtypes)
print()

# Check for non-numeric values in 'CO2 Emissions(g/km)' (binary values)
non_numeric_co2 = dF[~dF['CO2 Emissions(g/km)'].apply(lambda x:
isinstance(x, (int, float)))]
print(non_numeric_co2)
print()

# Check for non-numeric values in all columns
```

```
non_numeric_values = dF.applymap(lambda x: isinstance(x, str))
print(non_numeric_values.any())
print()

# Exclude non-numeric columns
numeric_columns = dF.select_dtypes(include=['float64', 'int64'])

# Calculate correlation of numeric columns w.r.t CO2
correlation = numeric_columns.corr()['CO2 Emissions(g/km)'].sort_values()
print(correlation)

import matplotlib.pyplot as plt
import seaborn as sns

# Calculate correlation
correlation = numeric_columns.corr()['CO2
Emissions(g/km)'].sort_values(ascending=False)

# Convert correlation Series to DataFrame with one column
correlation_df = correlation.to_frame()

plt.rcParams['figure.figsize'] = (10, 8)
sns.heatmap(correlation_df, cmap='coolwarm', linewidth=0.5, annot=True)

plt.title('Correlation')
plt.show()
# DATA VISUALIZATION
# Frequency distribution of cars of diff. companies

import matplotlib.pyplot as plt

plt.figure(figsize=(20,5))

dF.groupby('Make')['Make'].count().sort_values(ascending=False).plot(kind=
'bar',color='red')

plt.title('Frequency distribution of cars of different companies',
fontsize=25)
plt.xlabel('Company', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
# Frequency distribution of models

import matplotlib.pyplot as plt

plt.figure(figsize=(20,5))

dF.groupby('Model')['Model'].count().sort_values(ascending=False)[:25].plo
t(kind='bar')

plt.title('Distribution of models', fontsize=25)
plt.xlabel('Models', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
# Frequency distribution of vehicle class

import matplotlib.pyplot as plt

plt.figure(figsize=(20,5))

dF.groupby('Vehicle Class')['Vehicle
Class'].count().sort_values(ascending=False).plot(kind='bar',
color='green')

plt.title('Vehicle class distribution', fontsize=25)
plt.xlabel('Vehicle Class', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
# Frequency distribution of transmission

import matplotlib.pyplot as plt
import pandas as pd

dF = pd.read_csv('/content/CO2 Emissions_Canada.csv')
```

```
plt.figure(figsize=(20,5))

dF.groupby('Transmission')['Transmission'].count().sort_values(ascending=F
alse).plot(kind='bar', color='magenta')

plt.title('Distribution of transmission', fontsize=25)
plt.xlabel('Transmission type', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
#checking for distinct model names

dF['Model'].unique()
#checking for distinct company names

dF['Make'].unique()
import numpy as np

# Putting different transmission sub-catagories into their respective
catagories

dF['Transmission']=np.where(dF['Transmission'].isin(['A4','A5','A6','A7','
A8','A9','A10']),'Automatic',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['AV','AV7','AV8','AV6
','AV10']),'Autonomous',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['AS4','AS5','AS6','AS
8','AS7','AS9','AS10']),'Automatic Selection',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['M6','M5','M7']),'Man
ual',dF['Transmission'])
dF['Transmission']=np.where(dF['Transmission'].isin(['AM5','AM6','AM7','AM
8','AM9','AM10']),'Automated Manual',dF['Transmission'])
print(dF['Transmission'].unique())
print()
print(dF['Transmission'].value_counts())
# Renaming fuel types for better understanding

dF['Fuel Type']= np.where(dF['Fuel Type']=='X','Regular gasoline',dF['Fuel
Type'])
```

```
dF['Fuel Type']= np.where(dF['Fuel Type']=='Z','Premium gasoline',dF['Fuel
Type'])
dF['Fuel Type']= np.where(dF['Fuel Type']=='E','Ethanol',dF['Fuel Type'])
dF['Fuel Type']= np.where(dF['Fuel Type']=='D','Diesel',dF['Fuel Type'])
dF['Fuel Type']= np.where(dF['Fuel Type']=='N','Natural gas',dF['Fuel
Type'])

print(dF['Fuel Type'].unique())
print(dF['Vehicle Class'].unique())

#Counting no. of values of diff. fule types

dF['Fuel Type'].value_counts()

# Checking for rows where the fule type is natural gas & store the row
index in ind

dF_N = dF[dF['Fuel Type']== 'Natural gas']

ind = dF_N.index

print(dF_N)
# Dropping the rows where the fule type is natural gas

for i in ind:
    dF.drop(i, axis=0, inplace=True)
import pandas as pd



d_v =pd.get_dummies(dF['Fuel Type'], prefix='Fuel', drop_first=True)
dv = pd.get_dummies(dF["Transmission"], drop_first=True)
d_v.head()
#Concatinating all 3 dataframes

df = [dF, d_v,dv]

data = pd.concat(df, axis=1)
data.head()
#Dropping the transmission & fule type columns from data(cancatinated dF)
```

```
df = [dF, d_v,dv]
data = pd.concat(df, axis=1)
data.drop(['Fuel Type'], inplace=True, axis=1)
data.drop(['Transmission'], inplace=True, axis=1)

#It replaces each value in the Make/Model/Vehicle_Class column with its
corresponding frequency stored in dict
# converts the categorical values into numerical representations

df_freq = data['Make'].value_counts().to_dict()
mod_freq = data['Model'].value_counts().to_dict()
veh_freq = data['Vehicle Class'].value_counts().to_dict()
data['Make'] = data['Make'].map(df_freq)
data['Model'] = data['Model'].map(mod_freq)
data['Vehicle Class'] = data['Vehicle Class'].map(veh_freq)
#Divide the dF into target & features for prediction purpose

X = data.drop('CO2 Emissions(g/km)', axis=1)  # X contains all cols.
except CO2 (features)
y = data['CO2 Emissions(g/km)']    # Y conatains the CO2 col. (target)
# MULTIPLE REGRESSION


import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

scaler =StandardScaler()

X = data.drop('CO2 Emissions(g/km)', axis=1)
y = data['CO2 Emissions(g/km)']

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

regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)
my_dict={}
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)

r2 = r2_score(y_test, y_pred)

my_dict['ML']=r2
# Plotting of train set
plt.title("CO2 Emissions (Training Set)")
plt.xlabel("Actual CO2 Emissions (g/km)")
plt.ylabel("Predicted CO2 Emissions (g/km)")
plt.scatter(y_train, y_pred_train, color="lightcoral", label="Training
Data")
plt.plot(y_train, y_train, color="firebrick", label="Perfect Prediction")
plt.show()

# Plotting of test set
plt.title("CO2 Emissions (Testing Set)")
plt.xlabel("Actual CO2 Emissions (g/km)")
plt.ylabel("Predicted CO2 Emissions (g/km)")
plt.scatter(y_test, y_pred, color="lightcoral", label="Testing Data")
plt.plot(y_test, y_test, color="firebrick", label="Perfect Prediction")
plt.show()
# LASSO REGRESSION

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt


X = data.drop('CO2 Emissions(g/km)', axis=1)
y = data['CO2 Emissions(g/km)']

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

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)

y_pred = lasso.predict(X_test)
y_pred_train = lasso.predict(X_train)

r2 = r2_score(y_test, y_pred)
my_dict['Lasso']=r2

# Plotting of train set
plt.title("CO2 Emissions (Training Set)")
plt.xlabel("Actual CO2 Emissions (g/km)")
plt.ylabel("Predicted CO2 Emissions (g/km)")
plt.scatter(y_train, y_pred_train, color="lightcoral", label="Training
Data")
plt.plot(y_train, y_train, color="firebrick", label="Perfect Prediction")
plt.show()

# Plotting of test set
plt.title("CO2 Emissions (Testing Set)")
plt.xlabel("Actual CO2 Emissions (g/km)")
plt.ylabel("Predicted CO2 Emissions (g/km)")
plt.scatter(y_test, y_pred, color="lightcoral", label="Testing Data")
plt.plot(y_test, y_test, color="firebrick", label="Perfect Prediction")
plt.show()

# RANDOM FOREST

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

X = data.drop('CO2 Emissions(g/km)', axis=1)
```

```
y = data['CO2 Emissions(g/km)']

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

rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

y_pred = rf_regressor.predict(X_test)
y_pred_train = rf_regressor.predict(X_train)

r2 = r2_score(y_test, y_pred)
my_dict['Random']=r2

# Plotting of train set
plt.title("CO2 Emissions (Training Set)")
plt.xlabel("Actual CO2 Emissions (g/km)")
plt.ylabel("Predicted CO2 Emissions (g/km)")
plt.scatter(y_train, y_pred_train, color="lightcoral", label="Training
Data")
plt.plot(y_train, y_train, color="firebrick", label="Perfect Prediction")
plt.show()

# Plotting of test set
plt.title("CO2 Emissions (Testing Set)")
plt.xlabel("Actual CO2 Emissions (g/km)")
plt.ylabel("Predicted CO2 Emissions (g/km)")
plt.scatter(y_test, y_pred, color="lightcoral", label="Testing Data")
plt.plot(y_test, y_test, color="firebrick", label="Perfect Prediction")
plt.show()

#Decision tree
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score,mean_squared_error
model = DecisionTreeRegressor(random_state = 42)

model.fit(X_train, y_train)
dtr_pred = model.predict(X_test)
print(dtr_pred)
print(np.sqrt(mean_squared_error(y_test,dtr_pred )))
```

```
r2=r2_score(y_test, dtr_pred)
my_dict['DTR']=r2
frames = [dtr_pred, y_test.values]
result_pred = pd.DataFrame(data=frames)
result_pred = result_pred.T
result_pred.head()

#Ridge regression

from sklearn.linear_model import Ridge
model=Ridge(alpha=0.3)
model.fit(X_train,y_train)
print('Coefficient:',model.intercept_)
print('Coeff',model.coef_)
y_ridge = model.predict(X_test)
print(np.sqrt(mean_squared_error(y_ridge,y_test)))
r2=r2_score(y_test,y_ridge)
my_dict['Ridge']=r2
frames = [y_ridge, y_test.values]
result_pred = pd.DataFrame(data=frames)
result_pred = result_pred.T
print(result_pred)
ridge_pred = result_pred.rename(columns={0: 'pred_values',
1:'real_values'})
ridge_pred['pred_values'] = (ridge_pred['pred_values'].map(lambda x:
round(x,2)))

ridge_pred['diff'] = abs(ridge_pred['real_values']
-ridge_pred['pred_values'])


print('mean diff: ', abs(ridge_pred['diff']).mean())
print(ridge_pred.head(10))

# MULTIPLE REGRESSION


import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

scaler =StandardScaler()

X = data.drop('CO2 Emissions(g/km)', axis=1)
y = data['CO2 Emissions(g/km)']

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

regressor = LinearRegression()

regressor.fit(X_train, y_train)
my_dict={}
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)

r2 = r2_score(y_test, y_pred)

# LASSO REGRESSION

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt



X = data.drop('CO2 Emissions(g/km)', axis=1)
y = data['CO2 Emissions(g/km)']

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)

y_pred = lasso.predict(X_test)
y_pred_train = lasso.predict(X_train)

r2 = r2_score(y_test, y_pred)
my_dict['Lasso']=r2

# RANDOM FOREST

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

X = data.drop('CO2 Emissions(g/km)', axis=1)
y = data['CO2 Emissions(g/km)']

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

rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

y_pred = rf_regressor.predict(X_test)
y_pred_train = rf_regressor.predict(X_train)

r2 = r2_score(y_test, y_pred)
my_dict['Random']=r2

#Decision tree
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score,mean_squared_error
model = DecisionTreeRegressor(random_state = 42)

model.fit(X_train, y_train)
dtr_pred = model.predict(X_test)
```

```
print(dtr_pred)
print(np.sqrt(mean_squared_error(y_test,dtr_pred )))
r2=r2_score(y_test, dtr_pred)
my_dict['DTR']=r2
frames = [dtr_pred, y_test.values]
result_pred = pd.DataFrame(data=frames)
result_pred = result_pred.T
result_pred.head()

#Ridge regression

from sklearn.linear_model import Ridge
model=Ridge(alpha=0.3)
model.fit(X_train,y_train)
print('Coefficient:',model.intercept_)
print('Coeff',model.coef_)
y_ridge = model.predict(X_test)
print(np.sqrt(mean_squared_error(y_ridge,y_test)))
r2=r2_score(y_test,y_ridge)
my_dict['Ridge']=r2
frames = [y_ridge, y_test.values]
result_pred = pd.DataFrame(data=frames)
result_pred = result_pred.T
print(result_pred)
ridge_pred = result_pred.rename(columns={0: 'pred_values',
1:'real_values'})
ridge_pred['pred_values'] = (ridge_pred['pred_values'].map(lambda x:
round(x,2)))

ridge_pred['diff'] = abs(ridge_pred['real_values']
-ridge_pred['pred_values'])



print('mean diff: ', abs(ridge_pred['diff']).mean())
print(ridge_pred.head(10))
```

OUTPUT:

```
Make                                0
Model                               0
Vehicle Class                       0
Engine Size(L)                      0
Cylinders                           0
Transmission                        0
Fuel Type                           0
Fuel Consumption City (L/100 km)    0
Fuel Consumption Hwy (L/100 km)     0
Fuel Consumption Comb (L/100 km)    0
Fuel Consumption Comb (mpg)         0
CO2 Emissions(g/km)                 0
dtype: int64
There are 1103 duplicated rows in the data
```

	count	mean	std	min	25%	50%	75%	max
Engine Size(L)	6282.0	3.161812	1.365201	0.9	2.0	3.0	3.7	8.4
Cylinders	6282.0	5.618911	1.846250	3.0	4.0	6.0	6.0	16.0
Fuel Consumption City (L/100 km)	6282.0	12.610220	3.553066	4.2	10.1	12.1	14.7	30.6
Fuel Consumption Hwy (L/100 km)	6282.0	9.070583	2.278884	4.0	7.5	8.7	10.3	20.6
Fuel Consumption Comb (L/100 km)	6282.0	11.017876	2.946876	4.1	8.9	10.6	12.7	26.1
Fuel Consumption Comb (mpg)	6282.0	27.411016	7.245318	11.0	22.0	27.0	32.0	69.0
CO2 Emissions(g/km)	6282.0	251.157752	59.290426	96.0	208.0	246.0	289.0	522.0

```
Make                 object
Model                object
Vehicle Class        object
Engine Size(L)      float64
Cylinders             int64
Transmission         object
Fuel Type            object
```

```
Fuel Consumption City (L/100 km)    float64
Fuel Consumption Hwy (L/100 km)     float64
Fuel Consumption Comb (L/100 km)    float64
Fuel Consumption Comb (mpg)           int64
CO2 Emissions(g/km)                   int64
dtype: object

Empty DataFrame
Columns: [Make, Model, Vehicle Class, Engine Size(L), Cylinders,
Transmission, Fuel Type, Fuel Consumption City (L/100 km), Fuel
Consumption Hwy (L/100 km), Fuel Consumption Comb (L/100 km), Fuel
Consumption Comb (mpg), CO2 Emissions(g/km)]
Index: []

Make                                 True
Model                                True
Vehicle Class                        True
Engine Size(L)                      False
Cylinders                           False
Transmission                         True
Fuel Type                            True
Fuel Consumption City (L/100 km)    False
Fuel Consumption Hwy (L/100 km)     False
Fuel Consumption Comb (L/100 km)    False
Fuel Consumption Comb (mpg)         False
CO2 Emissions(g/km)                 False
dtype: bool

Fuel Consumption Comb (mpg)        -0.906783
Cylinders                           0.834687
Engine Size(L)                      0.854802
Fuel Consumption Hwy (L/100 km)     0.883424
Fuel Consumption Comb (L/100 km)    0.916840
Fuel Consumption City (L/100 km)    0.918756
CO2 Emissions(g/km)                 1.000000
Name: CO2 Emissions(g/km), dtype: float64

#checking for distinct model names
array(['ILX', 'ILX HYBRID', 'MDX 4WD', ...,
       'Tacoma 4WD D-Cab TRD Off-Road/Pro', 'Atlas Cross Sport 4MOTION',
       'XC40 T4 AWD'], dtype=object)


array(['ACURA', 'ALFA ROMEO', 'ASTON MARTIN', 'AUDI', 'BENTLEY', 'BMW',
       'BUICK', 'CADILLAC', 'CHEVROLET', 'CHRYSLER', 'DODGE', 'FIAT',
       'FORD', 'GMC', 'HONDA', 'HYUNDAI', 'INFINITI', 'JAGUAR', 'JEEP',
       'KIA', 'LAMBORGHINI', 'LAND ROVER', 'LEXUS', 'LINCOLN', 'MASERATI',
```

```
       'MAZDA', 'MERCEDES-BENZ', 'MINI', 'MITSUBISHI', 'NISSAN',
       'PORSCHE', 'RAM', 'ROLLS-ROYCE', 'SCION', 'SMART', 'SRT', 'SUBARU',
       'TOYOTA', 'VOLKSWAGEN', 'VOLVO', 'GENESIS', 'BUGATTI'],
      dtype=object)

['Automatic Selection' 'Manual' 'Autonomous' 'Automated Manual'
 'Automatic']

Transmission
Automatic Selection    3127
Automatic              1851
Manual                 1185
Automated Manual        646
Autonomous              576
Name: count, dtype: int64
```

['Premium gasoline' 'Diesel' 'Regular gasoline' 'Ethanol' 'Natural gas']
['COMPACT' 'SUV - SMALL' 'MID-SIZE' 'TWO-SEATER' 'MINICOMPACT'
'SUBCOMPACT' 'FULL-SIZE' 'STATION WAGON - SMALL' 'SUV - STANDARD'
'VAN - CARGO' 'VAN - PASSENGER' 'PICKUP TRUCK - STANDARD' 'MINIVAN'
'SPECIAL PURPOSE VEHICLE' 'STATION WAGON - MID-SIZE'
'PICKUP TRUCK - SMALL']

Fuel Type
Regular gasoline 3039
Premium gasoline 2765
Ethanol 330
Diesel 147
Natural gas 1
Name: count, dtype: int64

```
         Make             Model Vehicle Class  Engine Size(L)  Cylinders
\
2439  CHEVROLET  IMPALA DUAL FUEL      MID-SIZE             3.6          6

     Transmission    Fuel Type  Fuel Consumption City (L/100 km)  \
2439          AS6  Natural gas                              15.2

      Fuel Consumption Hwy (L/100 km)  Fuel Consumption Comb (L/100 km)  \
2439                              9.5                               12.7

      Fuel Consumption Comb (mpg)  CO2 Emissions(g/km)
2439                           22                  213
```

	Fuel_Ethanol	Fuel_Premium gasoline	Fuel_Regular gasoline
0	False	True	False
1	False	True	False
2	False	True	False
3	False	True	False
4	False	True	False

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	FuelType	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	...	AS8	AS9	AV	AV10	AV6	AV7	AV8	M5	M6	M7	
0	ACURA	ILX	COMPACT	2.0	4	AS5	Premium gasoline	9.9	6.7	8.5	...	False	False	False	False	False	False	False	False	False	False	False
1	ACURA	ILX	COMPACT	2.4	4	M6	Premium gasoline	11.2	7.7	9.6	...	False	False	False	False	False	False	False	False	False	True	False
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Premium gasoline	6.0	5.8	5.9	...	False	False	False	False	False	False	True	False	False	False	False

3	ACURA	MDX 4WD	SUV - SMALL	3.5		6	AS6	Premium gasoline	12.7	9.1	11.1	. .	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	ACURA	RDX AWD	SUV - SMALL	3.5		6	AS6	Premium gasoline	12.1	8.7	10.6	. .	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

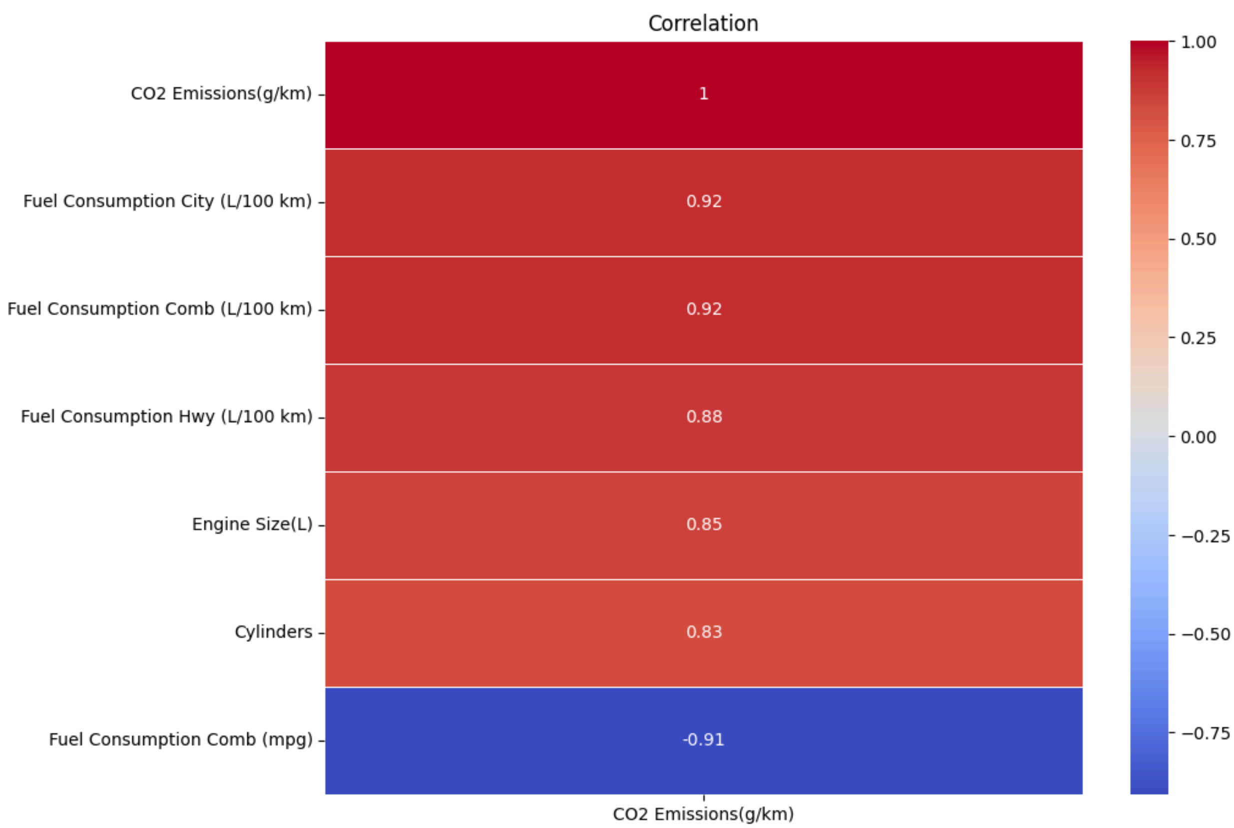
5 rows × 41 columns

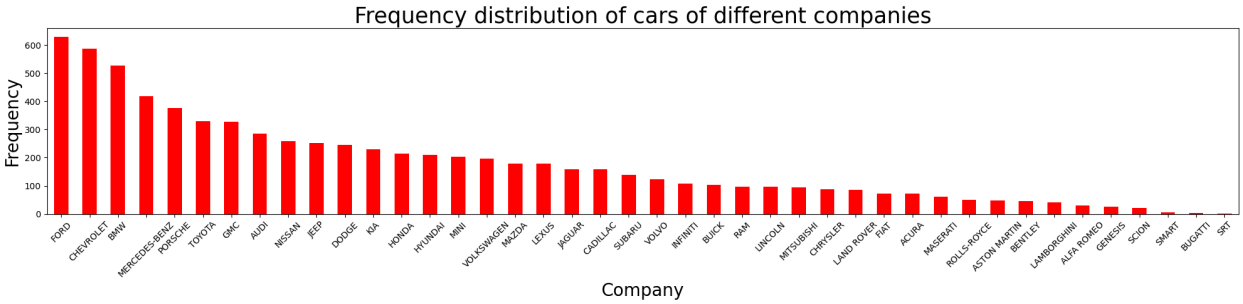
Make	**Model**	**Vehicle Class**	**Engine Size (L)**	**Cylinders**	**Fuel Consumption City (L/100 km)**	**Fuel Consumption Hwy (L/100 km)**	**Fuel Consumption Comb (L/100 km)**	**Fuel Consumption Comb (mpg)**	**Fuel_Ethanol**	**. .**	**AS8**	**AS9**	**AV**	**AV10**	**AV6**	**AV7**	**AV8**	**M5**	**M6**	**M7**	
0	51	7	903	2.0	4	9.9	6.7	8.5	33	False	. .	False	False	False	False	False	False	False	False	False	False	False	False
1	51	7	903	2.4	4	11.2	7.7	9.6	29	False	. .	False	False	False	False	False	False	False	False	False	True	False	False
2	51	2	903	1.5	4	6.0	5.8	5.9	48	False	. .	False	False	False	False	False	False	True	False	False	False	False	False

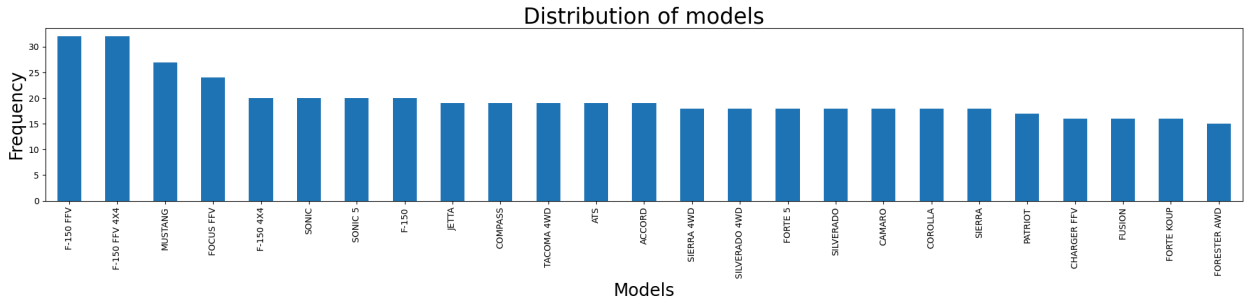
										se		se	se	se	se	se		se	se	se	se
3	51	1	1006	3.5	6	12.7	9.1	11.1	25	False	...	False	False	False	False	False	False	False	False	False	False
4	51	4	1006	3.5	6	12.1	8.7	10.6	27	False	...	False	False	False	False	False	False	False	False	False	False

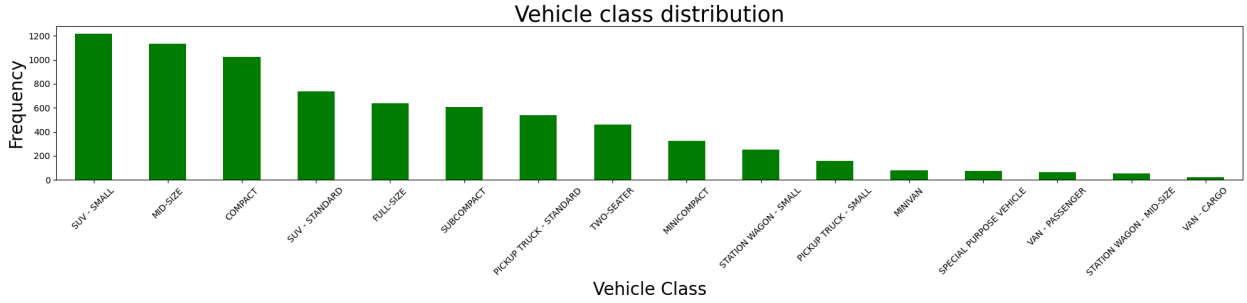
5 rows × 38 columns

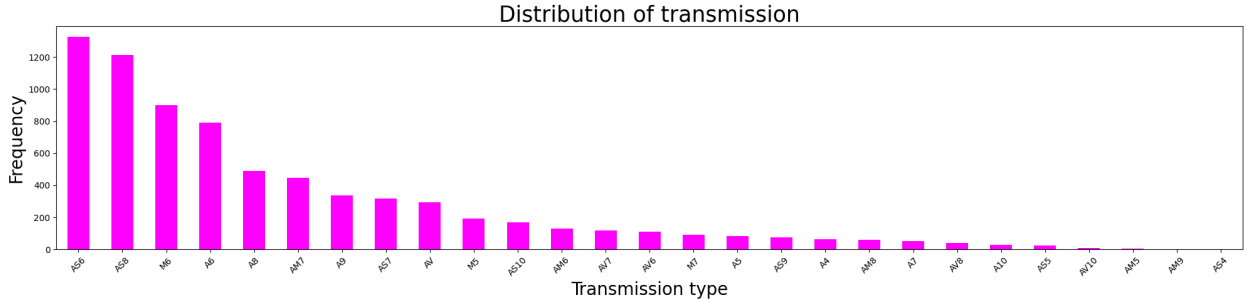
DATA VISUALIZATION:



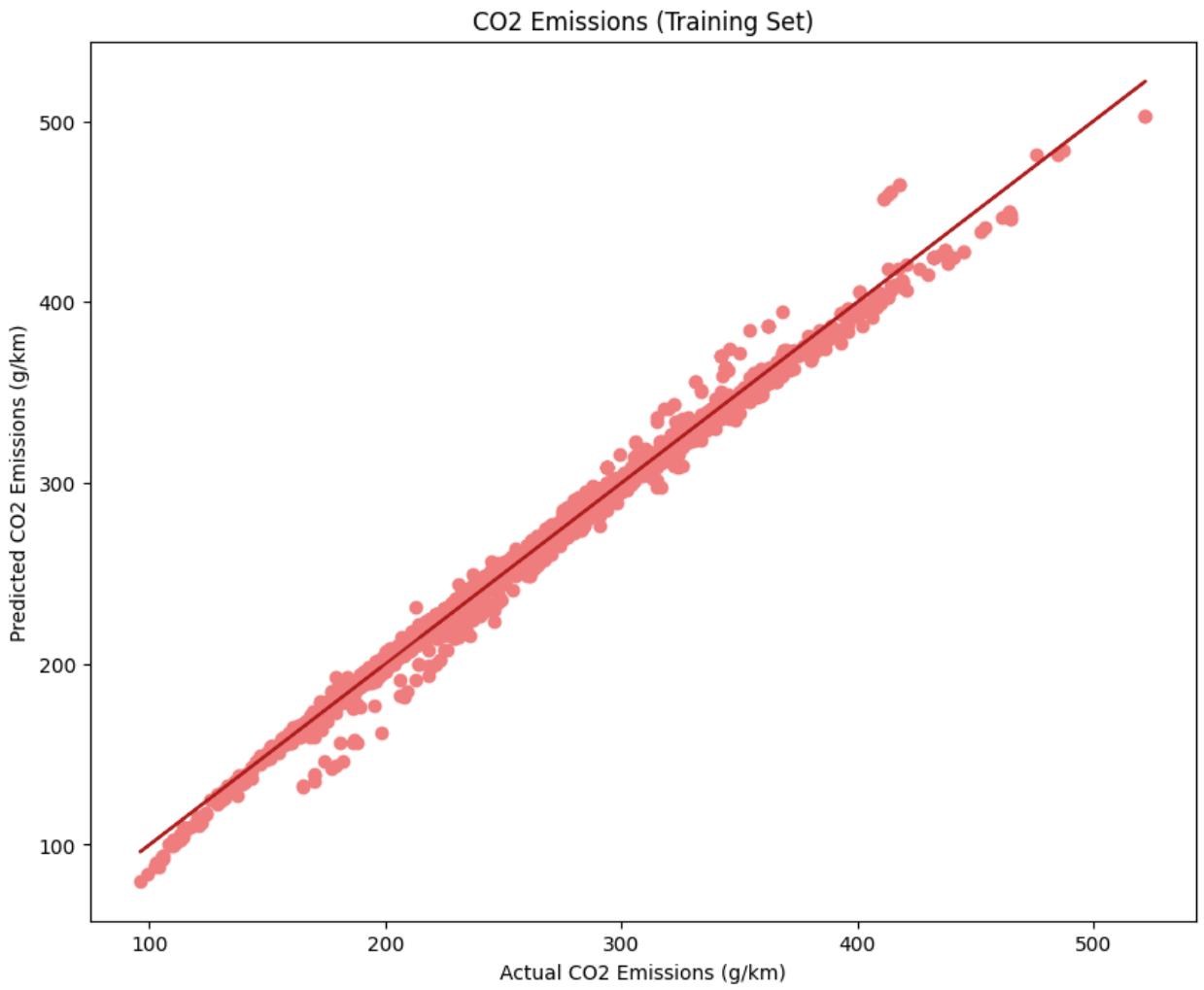

Frequency distribution of cars of different companies
Frequency
0
100
200
300
400
500
600
FORD
CHEVROLET
BMW
MERCEDES-BENZ
PORSCHE
TOYOTA
GMC
AUDI
NISSAN
JEEP
DODGE
KIA
HONDA
HYUNDAI
MINI
VOLKSWAGEN
MAZDA
LEXUS
JAGUAR
CADILLAC
SUBARU
VOLVO
INFINITI
BUICK
RAM
LINCOLN
MITSUBISHI
CHRYSLER
LAND ROVER
FIAT
ACURA
MASERATI
ROLLS-ROYCE
ASTON MARTIN
BENTLEY
LAMBORGHINI
ALFA ROMEO
GENESIS
SCION
SMART
BUGATTI
SRT
Company

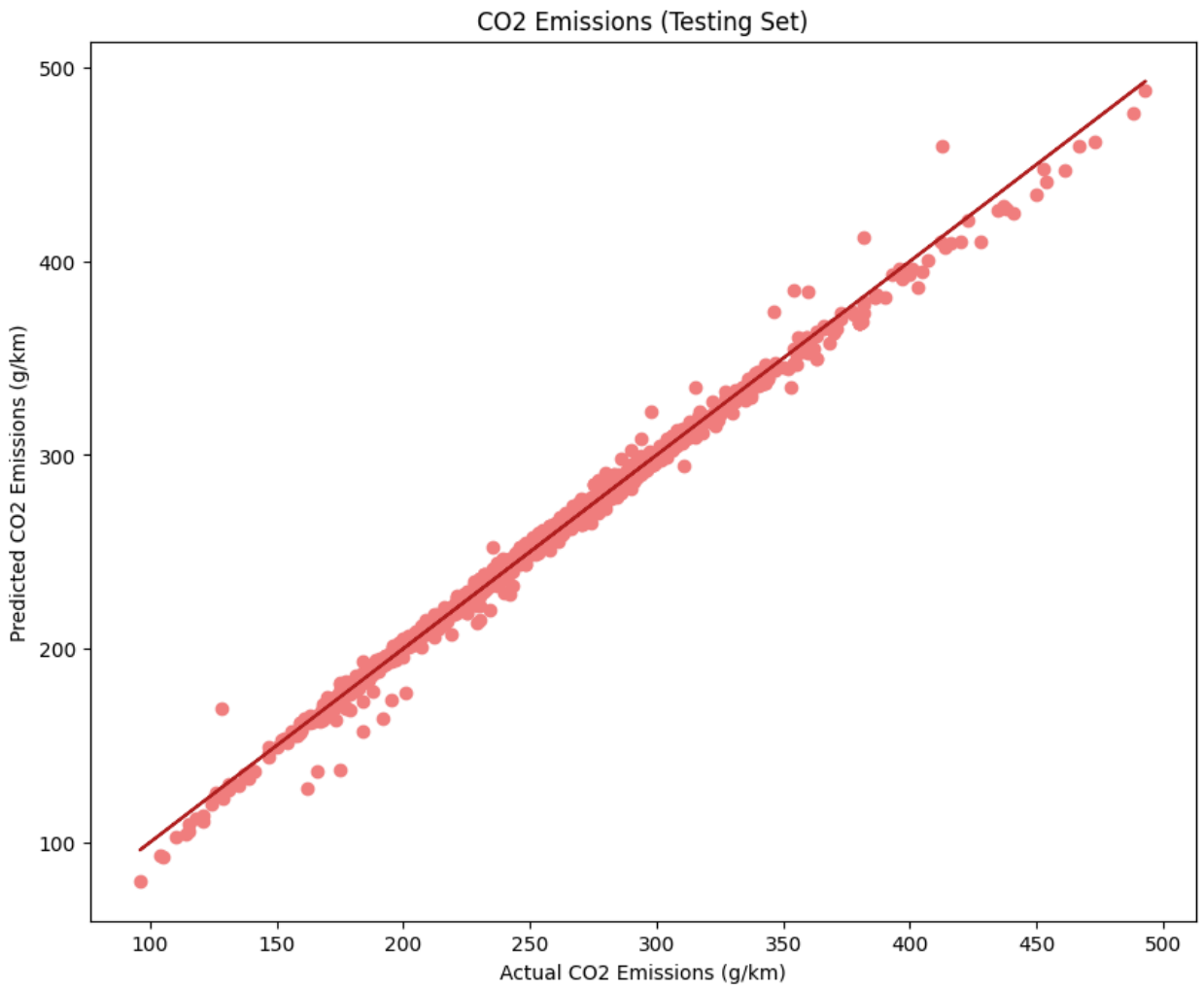


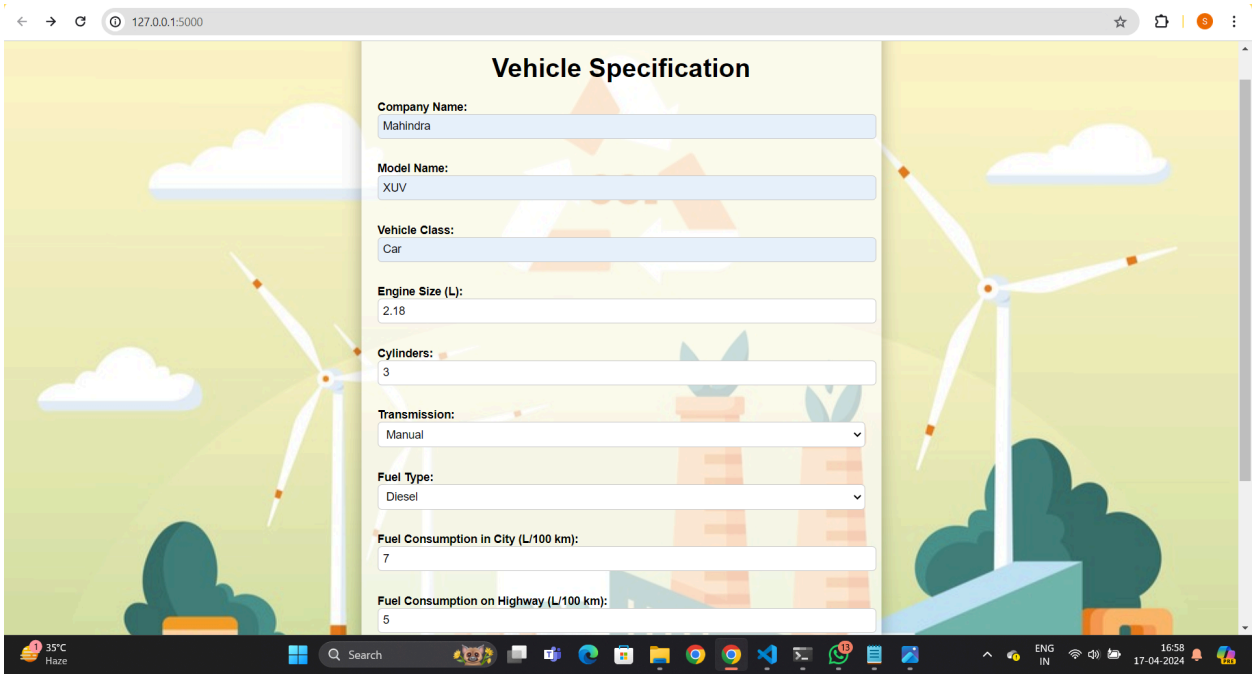


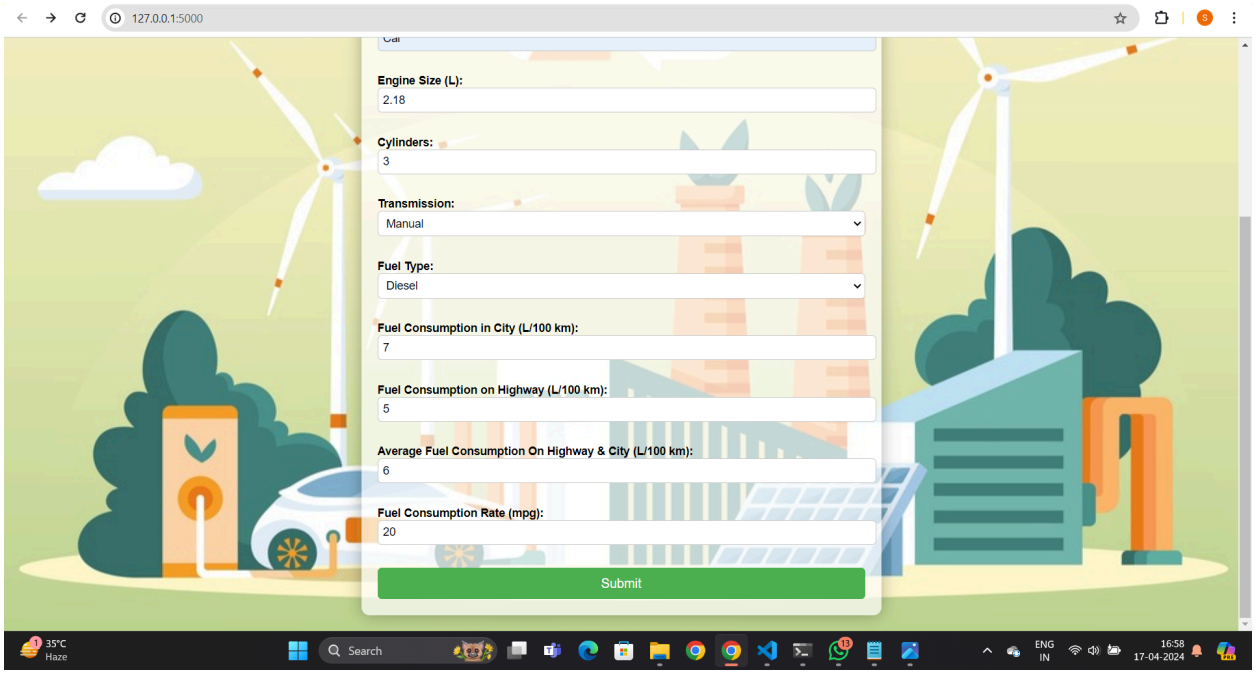



```
# MULTIPLE REGRESSION
```

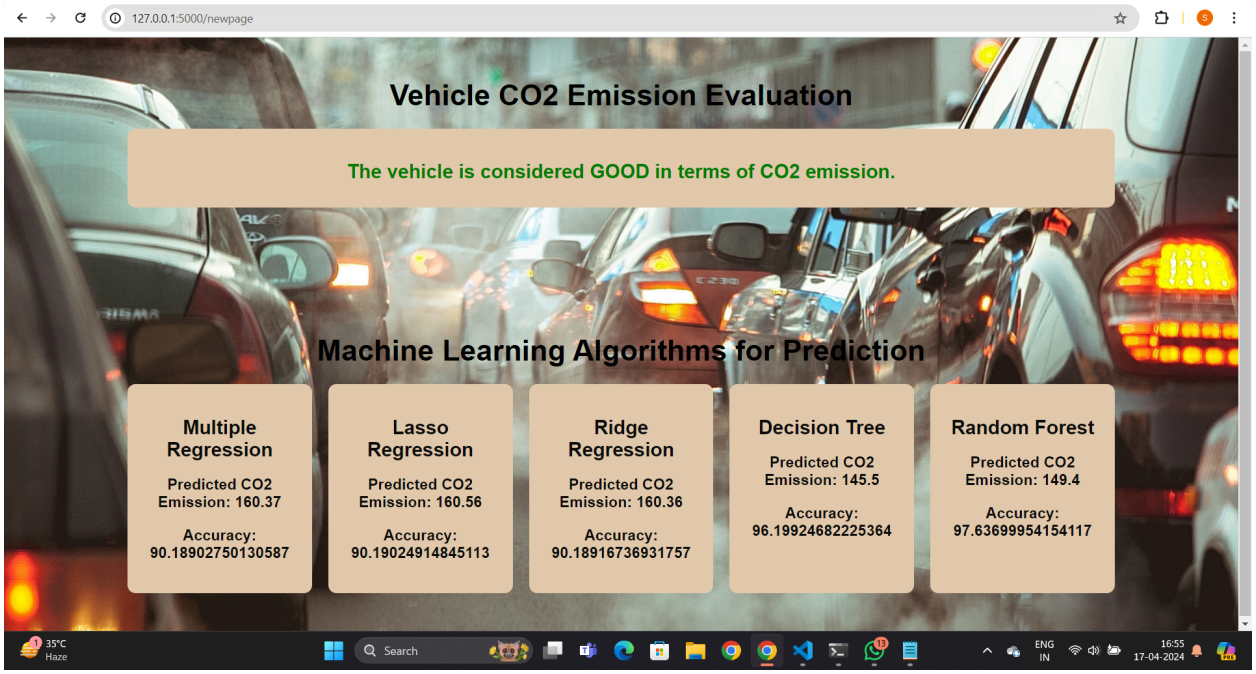

CO2 Emissions (Training Set)
Predicted CO2 Emissions (g/km)
Actual CO2 Emissions (g/km)
100
200
300
400
500

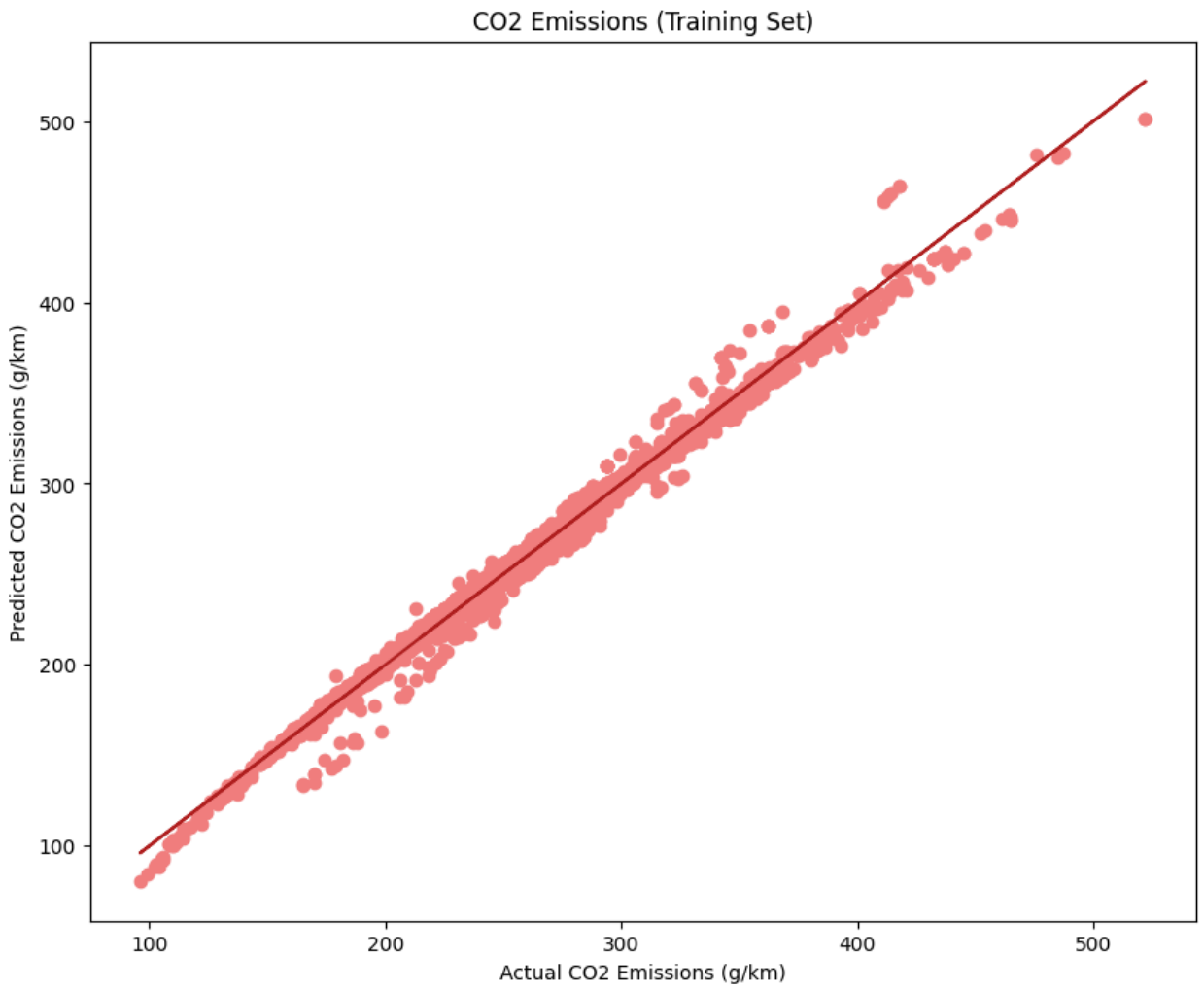

CO2 Emissions (Testing Set)
Predicted CO2 Emissions (g/km)
Actual CO2 Emissions (g/km)
100
150
200
250
300
350
400
450
500
100
200
300
400
500

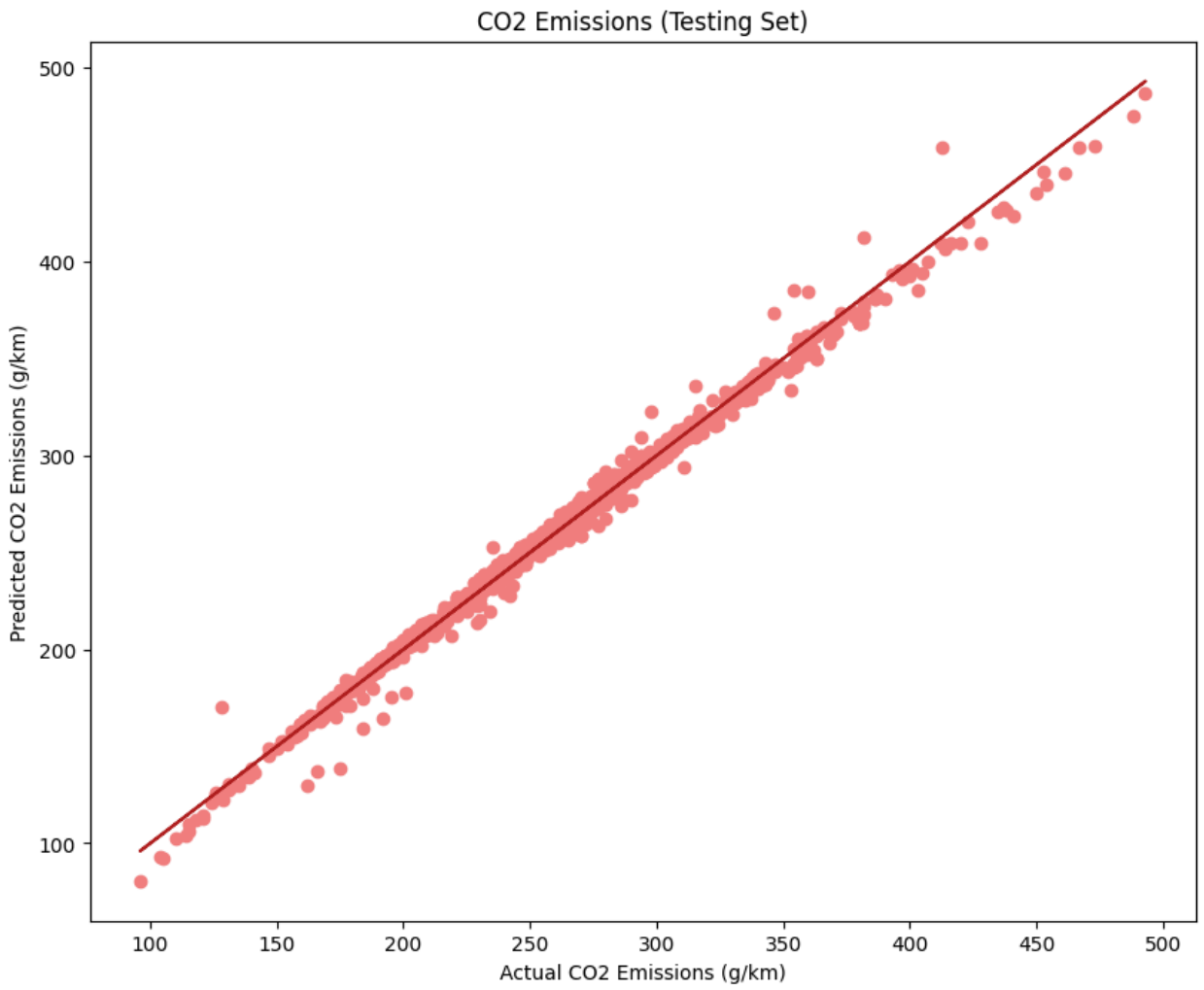

127.0.0.1:5000
Vehicle Specification
Company Name:
Mahindra
Model Name:
XUV
Vehicle Class:
Car
Engine Size (L):
2.18
Cylinders:
3
Transmission:
Manual
Fuel Type:
Diesel
Fuel Consumption in City (L/100 km):
7
Fuel Consumption on Highway (L/100 km):
5
35°C
Haze
Search
ENG
IN
16:58
17-04-2024

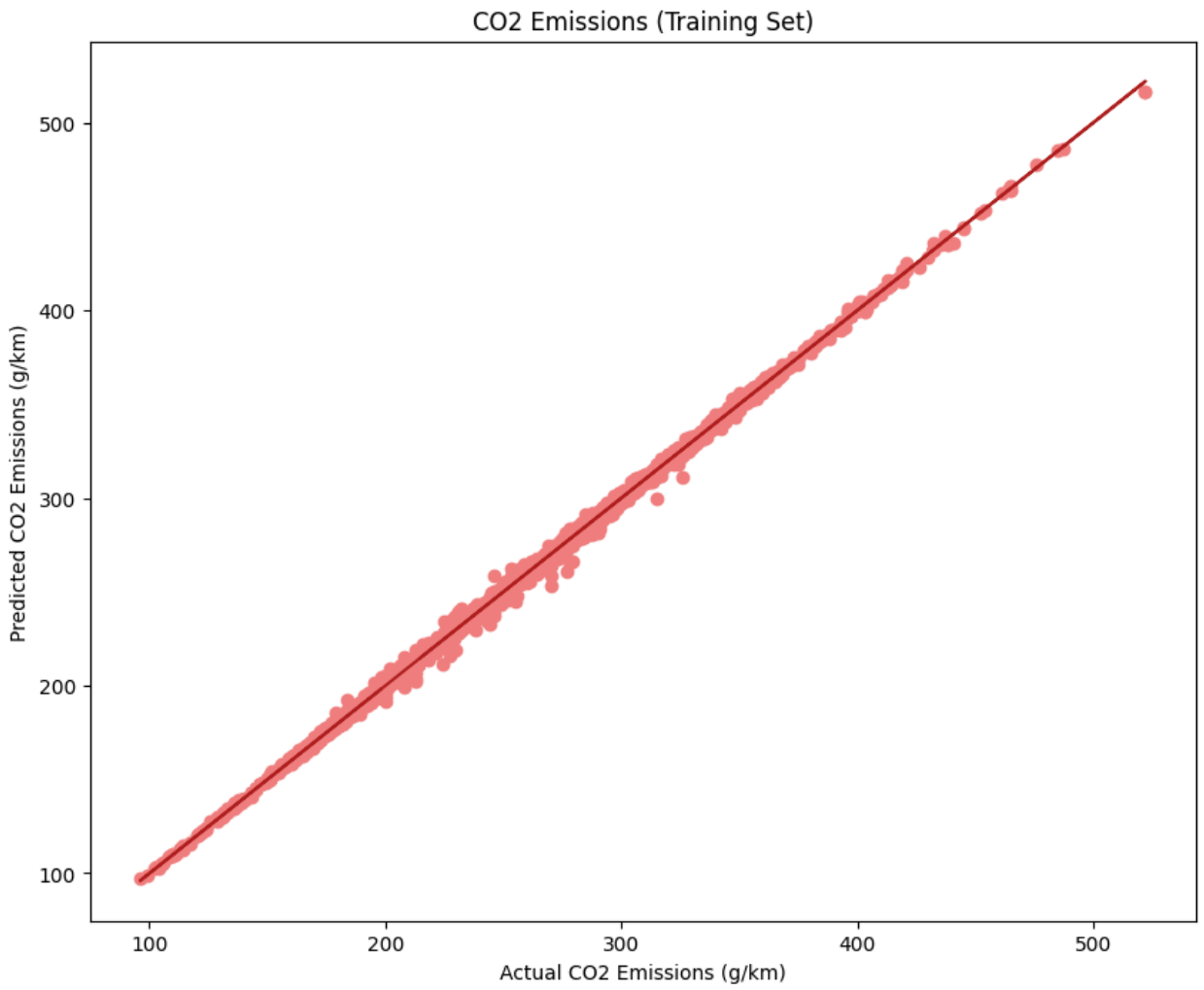

127.0.0.1:5000
Engine Size (L):
2.18
Cylinders:
3
Transmission:
Manual
Fuel Type:
Diesel
Fuel Consumption in City (L/100 km):
7
Fuel Consumption on Highway (L/100 km):
5
Average Fuel Consumption On Highway & City (L/100 km):
6
Fuel Consumption Rate (mpg):
20
Submit
35°C
Haze
Search
ENG
IN
16:58
17-04-2024

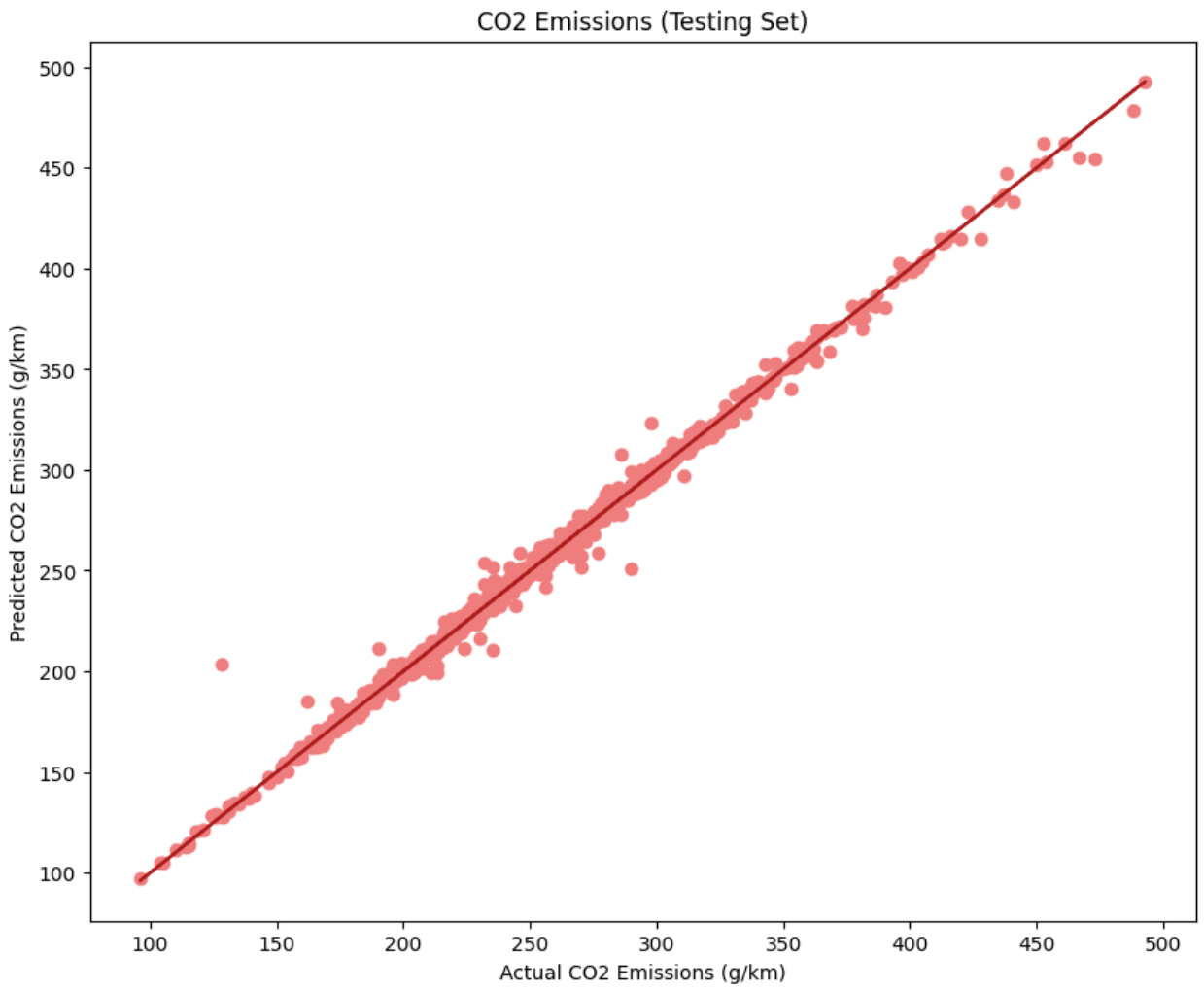

127.0.0.1:5000/newpage
Vehicle CO2 Emission Evaluation
The vehicle is considered BAD in terms of CO2 emission.
Machine Learning Algorithms for Prediction
Multiple Regression
Predicted CO2 Emission: 225.9
Accuracy: 90.18902750130587
Lasso Regression
Predicted CO2 Emission: 225.89
Accuracy: 90.19024914845113
Ridge Regression
Predicted CO2 Emission: 225.88
Accuracy: 90.18916736931757
Decision Tree
Predicted CO2 Emission: 150.33
Accuracy: 95.94480830409137
Random Forest
Predicted CO2 Emission: 176.24
Accuracy: 97.48088731502416


127.0.0.1:5000/newpage
Vehicle CO2 Emission Evaluation
The vehicle is considered GOOD in terms of CO2 emission.
Machine Learning Algorithms for Prediction
Multiple Regression
Predicted CO2 Emission: 160.37
Accuracy: 90.18902750130587
Lasso Regression
Predicted CO2 Emission: 160.56
Accuracy: 90.19024914845113
Ridge Regression
Predicted CO2 Emission: 160.36
Accuracy: 90.18916736931757
Decision Tree
Predicted CO2 Emission: 145.5
Accuracy: 96.19924682225364
Random Forest
Predicted CO2 Emission: 149.4
Accuracy: 97.63699954154117


CO2 Emissions (Training Set)
Predicted CO2 Emissions (g/km)
Actual CO2 Emissions (g/km)
100
200
300
400
500


CO2 Emissions (Testing Set)
Predicted CO2 Emissions (g/km)
Actual CO2 Emissions (g/km)
100
150
200
250
300
350
400
450
500
100
200
300
400
500


CO2 Emissions (Training Set)
Predicted CO2 Emissions (g/km)
Actual CO2 Emissions (g/km)
100
200
300
400
500



CONCLUSION:

Provide recommendations for policymakers, manufacturers, and consumers based on the model results, suggesting strategies for reducing emissions and promoting environmental sustainability in the transportation sector.