FUELWISE

Machine Learning Project

FUELWISE
STRIVE FOR SUSTAINABLITY

OUTLINE

- Team Members
- Introduction
- The Problem
- Proposed Solution
- Methods & Approaches
- Project Outcome
- Next Steps
- Conclusion

TEAM MEMBERS

Team Leader

Deem Alrashidi

Group Member

Lama Alhujaili

Group Member

Sara Thaer

Group Member

Shahad Alsadah

Group Member

Sana Araj

Group Member

Shahad Adel

INTRODUCTION

Fuel consumption prediction utilizes advanced analytics and machine learning models to forecast the amount of fuel vehicles will use, enabling more efficient operations and reduced environmental impact.

Importance of Accurately Predicting Fuel Consumption:

Cost Efficiency

Operational Optimization

Reduced Carbon Footprints

THE PROBLEM

The problem of current fuel consumption prediction methods is:

- Lack of accuracy
- High variability in results
- Inefficiencies

And this can have a huge impact!

PROPOSED SOLUTION

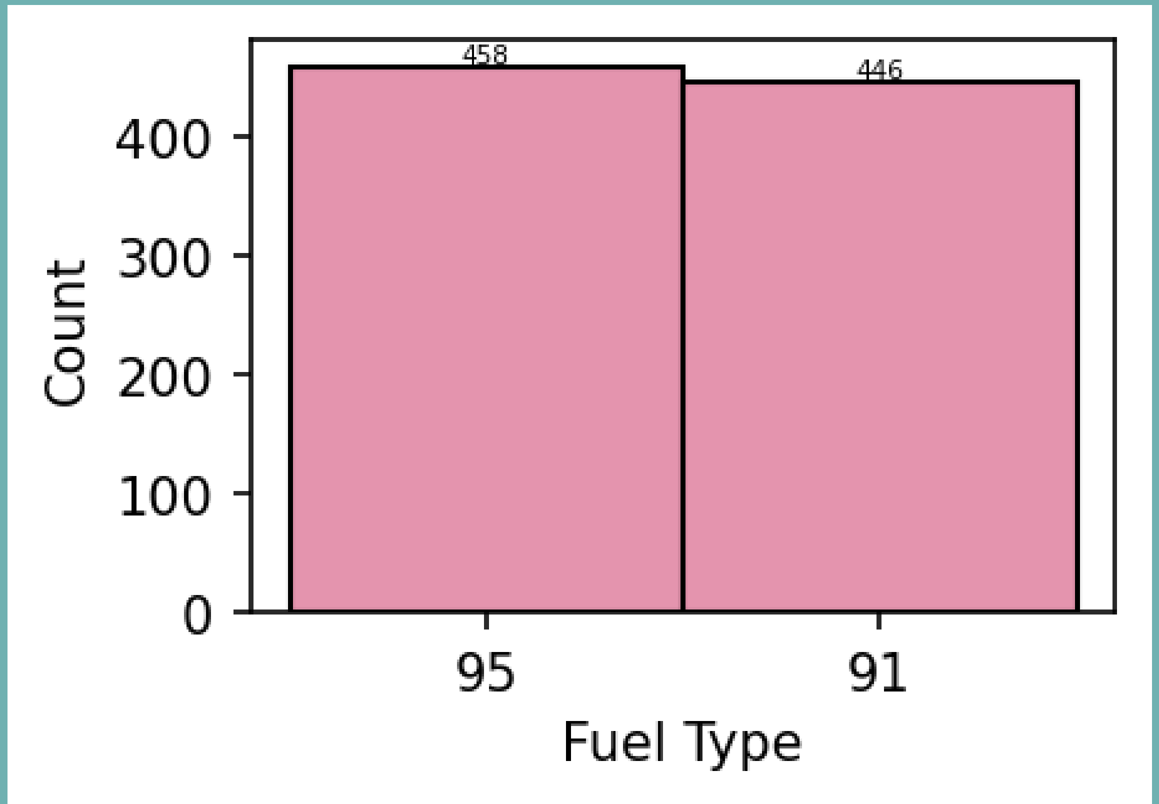
We will use machine learning-based approach to predict fuel consumption more accurately.

Benefits of using machine learning model:

Adaptability to new data	Improvement in prediction accuracy	Automation of estimation processes.

DATA CLEANING

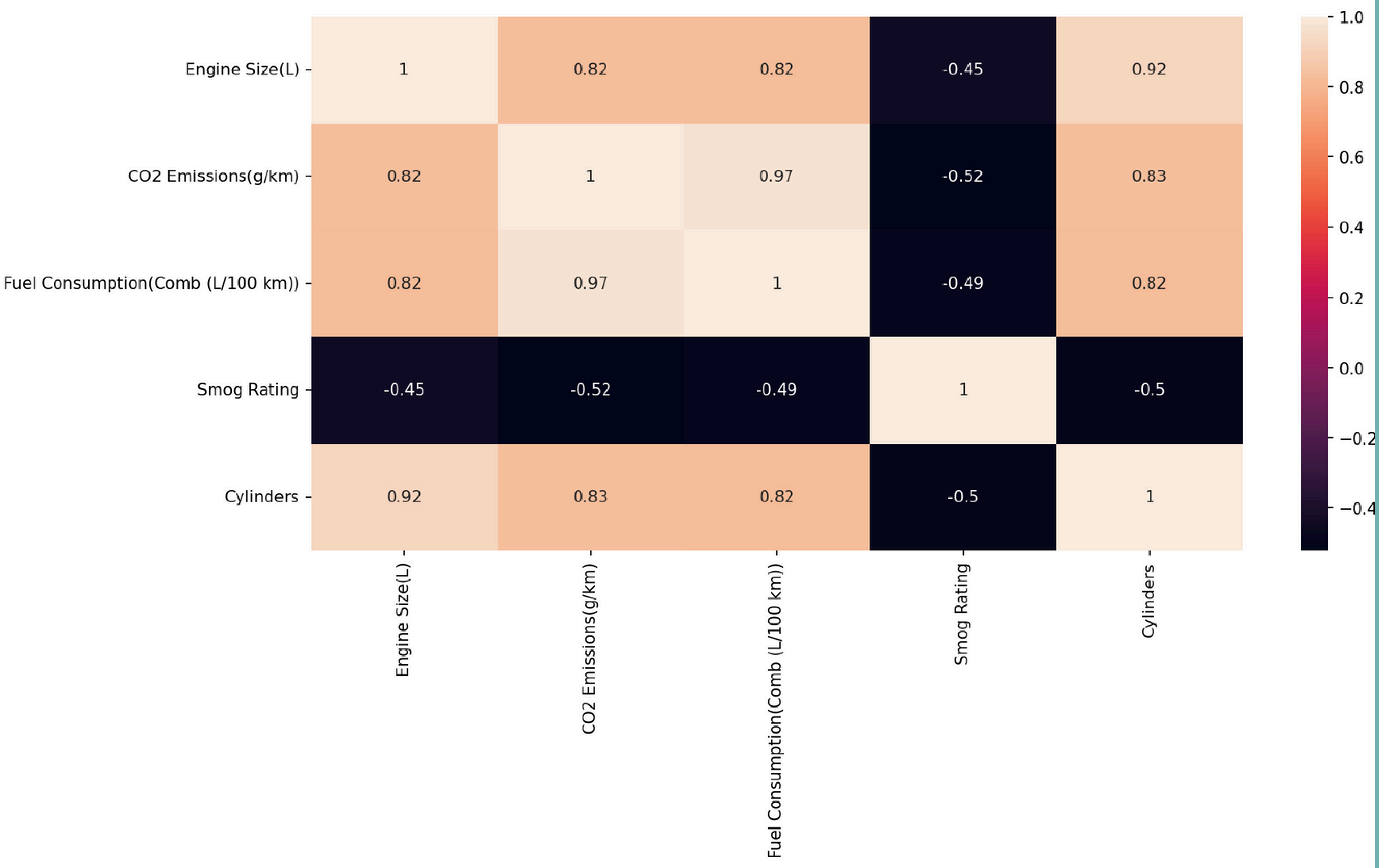
```
[ ]  df = df.replace({'Fuel Type' : {'Z':'95', 'X': '91'}})
     df = df[~df['Fuel Type'].isin(['E','D'])]
     df
```




```
df.isna().sum()

Make                                    0
Model                                   0
Vehicle Class                           0
Engine Size(L)                          0
Cylinders                               0
Fuel Type                               0
Fuel Consumption(Comb (L/100 km))       0
CO2 Emissions(g/km)                     0
CO2 Rating                              0
Smog Rating                             0
dtype: int64
```

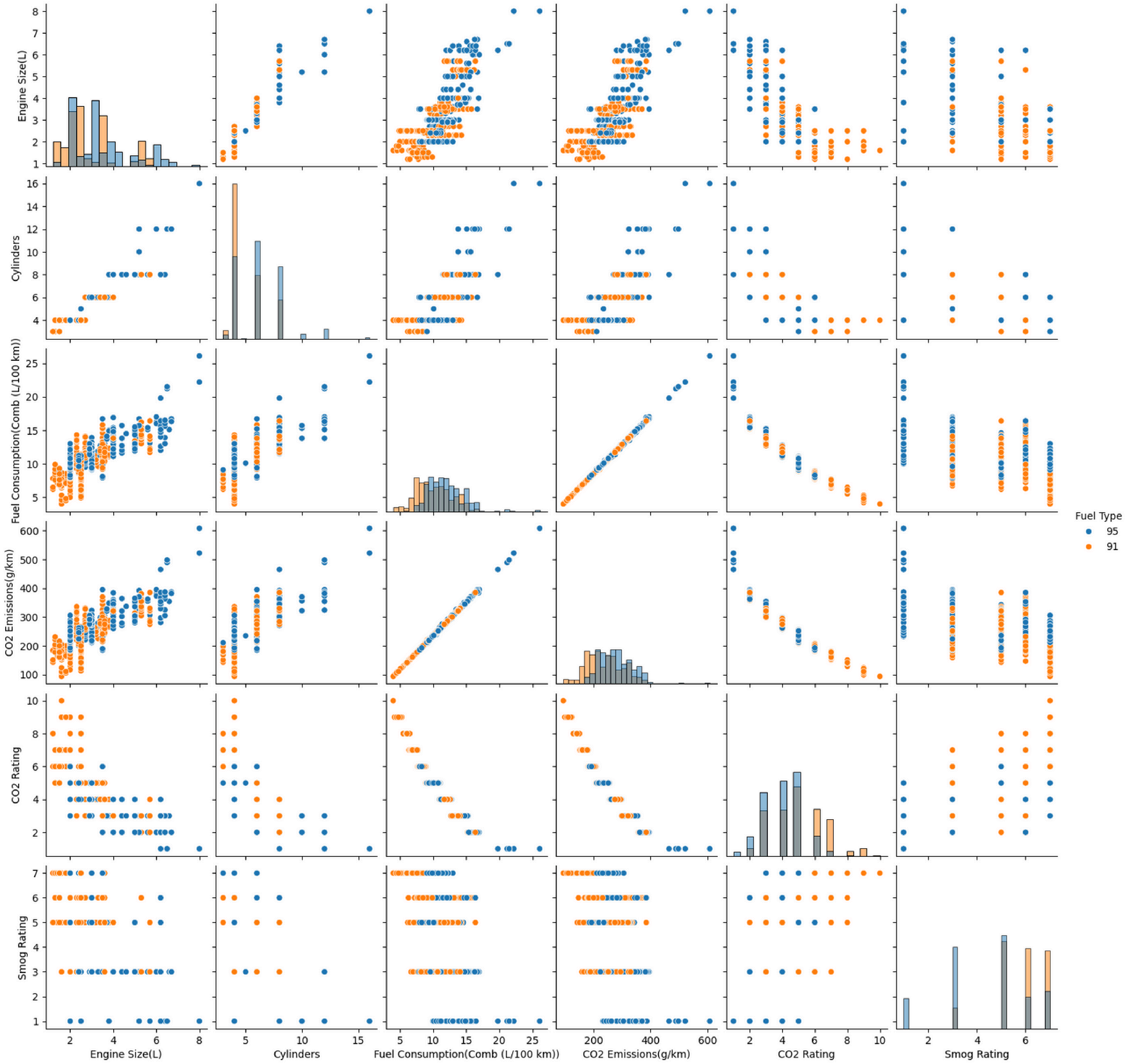
```
Correlation= data[['Engine Size(L)','CO2 Emissions(g/km)', 'Fuel Consumption(Comb (L/100 km))', 'Smog Rating','Cylinders']].corr()
Correlation
```

	Engine Size(L)	CO2 Emissions(g/km)	Fuel Consumption(Comb (L/100 km))	Smog Rating	Cylinders
Engine Size(L)	1.000000	0.824188	0.818694	-0.448239	0.920698
CO2 Emissions(g/km)	0.824188	1.000000	0.971671	-0.520437	0.833241
Fuel Consumption(Comb (L/100 km))	0.818694	0.971671	1.000000	-0.490473	0.821718
Smog Rating	-0.448239	-0.520437	-0.490473	1.000000	-0.502149
Cylinders	0.920698	0.833241	0.821718	-0.502149	1.000000



DATA ANALYSIS

Correlation



DATA ANALYSIS

Bivariate analysis

THE FEATURES

Before Preprocessing and Cleaning

Investigate data (Missing values, Descriptions, Data Types)

+ Code + Text

```
data=pd.read_csv("dataset.csv")
data.head()
```

	Model Year	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)	CO2 Rating	Smog Rating
0	2022	Acura	ILX	Compact	2.4	4	AM8	Z	9.9	7.0	8.6	33	200	6	3
1	2022	Acura	MDX SH-AWD	SUV: Small	3.5	6	AS10	Z	12.6	9.4	11.2	25	263	4	5
2	2022	Acura	RDX SH-AWD	SUV: Small	2.0	4	AS10	Z	11.0	8.6	9.9	29	232	5	6
3	2022	Acura	RDX SH-AWD A-SPEC	SUV: Small	2.0	4	AS10	Z	11.3	9.1	10.3	27	242	5	6
4	2022	Acura	TLX SH-AWD	Compact	2.0	4	AS10	Z	11.2	8.0	9.8	29	230	5	7

THE FEATURES

After Preprocessing and Cleaning

```
x.head()
```

	Engine Size(L)	Cylinders	Fuel Consumption(Comb (L/100 km))	CO2 Emissions(g/km)	CO2 Rating	Smog Rating	Vehicle Class_X	Make_X	Model_Y	91	95
1	3.5	6	11.2	263	4	5	6.0	25.0	576.0	0	1
2	2.0	4	9.9	232	5	6	6.0	25.0	511.0	0	1
3	2.0	4	10.3	242	5	6	6.0	25.0	512.0	0	1
4	2.0	4	9.8	230	5	7	2.0	25.0	513.0	0	1
5	2.0	4	9.8	231	5	7	2.0	25.0	514.0	0	1

METHODS & APPROACHES

SVM

- SVM is a robust algorithm for classification and regression tasks.
- Accuracy is 99.3%

SVM

```python
from sklearn.svm import SVR

# Assuming you've defined your SVM regressor with appropriate parameters
svm_regressor = SVR(kernel='rbf')
# Fit the SVR model to your training data
svm_regressor.fit(xtrain, ytrain)
# Predict on the test set
y_pred = svm_regressor.predict(xtest)

# Evaluate the model
mse = mean_squared_error(ytest, y_pred)
print("Mean Squared Error:", mse)

# Calculate accuracy (you may want to use a different metric for regression tasks)
accuracy = svm_regressor.score(xtest, ytest)
print("Accuracy:", accuracy*100)
```

```
Mean Squared Error: 0.028949171032108784
Accuracy: 99.28363455213638
```

METHODS & APPROACHES

KNN

- Straightforward and intuitive algorithm.
- Its performance depends on the choice of the hyperparameter k.
- The Accuracy is 97%

KNN

```
from sklearn.neighbors import KNeighborsRegressor

knn_regressor = KNeighborsRegressor(n_neighbors=3)

knn_regressor.fit(xtrain, ytrain)
```

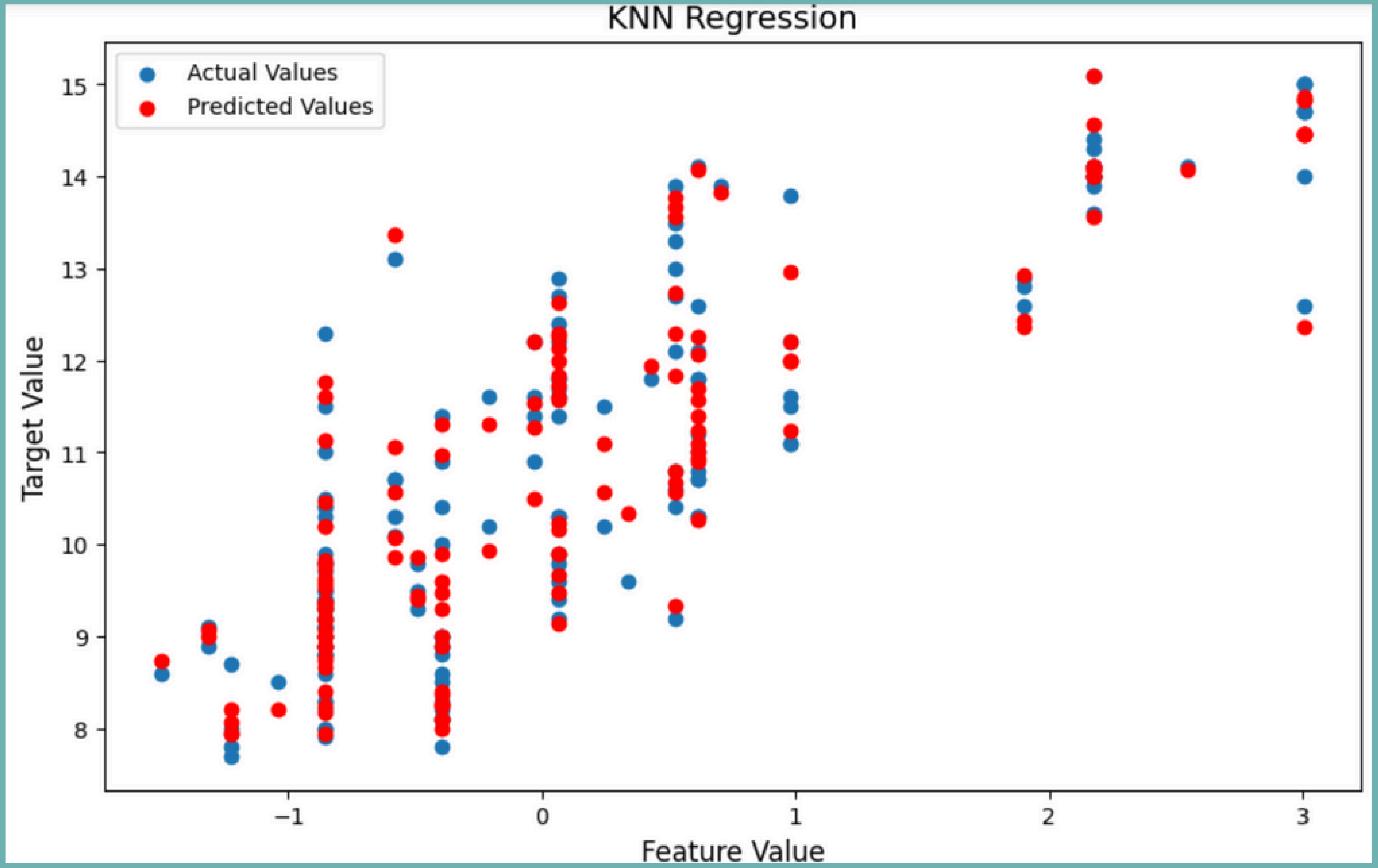
```
KNeighborsRegressor
KNeighborsRegressor(n_neighbors=3)
```

```
[ ] y_pred = knn_regressor.predict(xtest)
```

```
[ ] mse = mean_squared_error(ytest, y_pred)
    print("Mean Squared Error:", mse)

    accuracy = knn_regressor.score(xtest, ytest)
    print("R^2 Score:", accuracy)
```

```
Mean Squared Error: 0.12236781609195396
R^2 Score: 0.9697193141449272
```

METHODS & APPROACHES



METHODS & APPROACHES

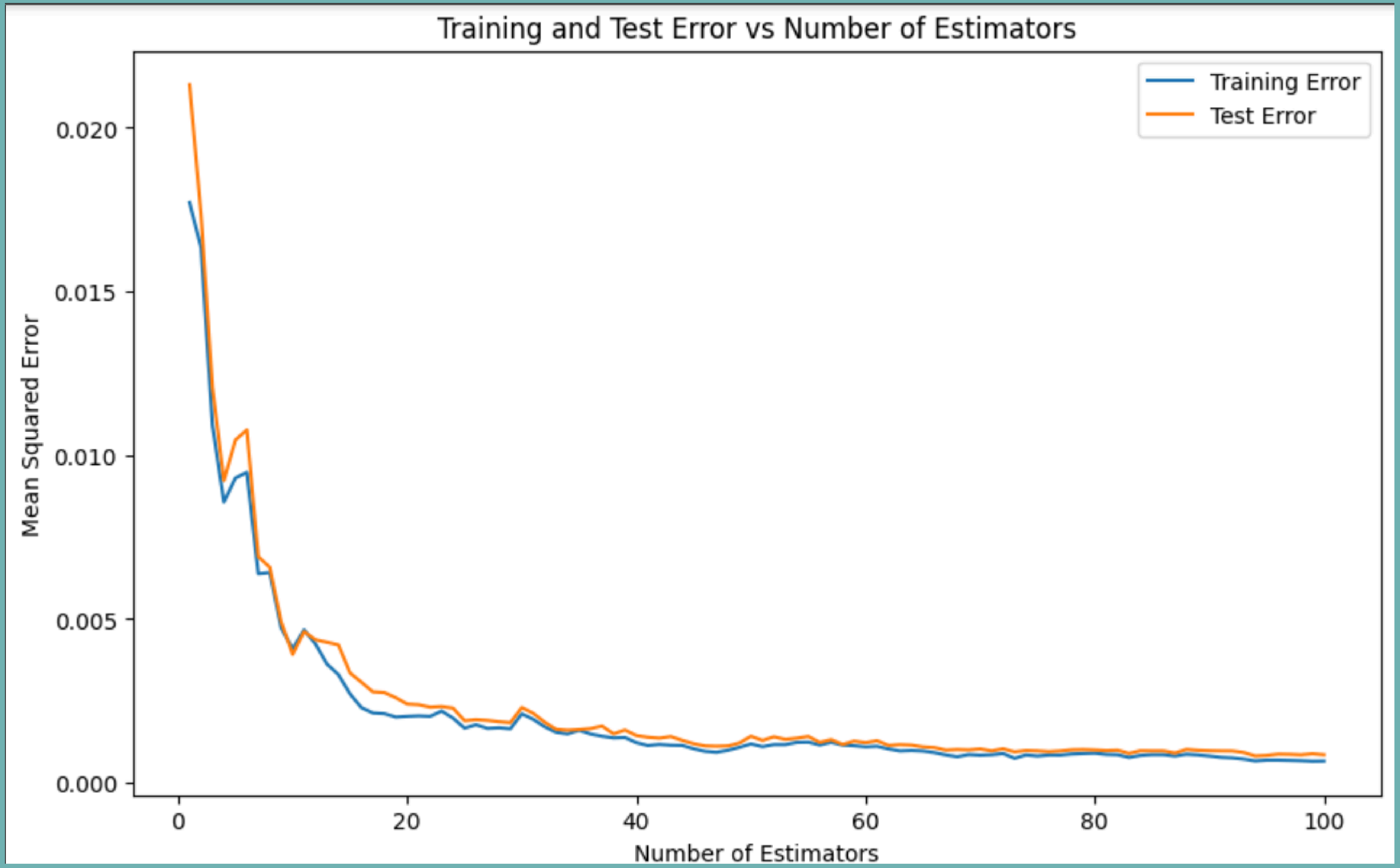
AdaBoost Regressor

- Works on multiple weak models
- adjusts the weights
- deals with complex data and avoids overfitting
- The Accuracy is 99.9%

```
import numpy as np
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Assuming xtrain, ytrain, xtest, ytest are already defined

# Create a DecisionTreeRegressor as the base estimator
base_estimator = DecisionTreeRegressor(max_depth=4)

# Create the AdaBoost regressor
ada_regressor = AdaBoostRegressor(base_estimator=base_estimator, n_estimators=50, learning_rate=1.0, random_state=42)

# Fit the AdaBoost regressor to your training data
ada_regressor.fit(xtrain, ytrain)

# Predict on the test set
y_pred = ada_regressor.predict(xtest)

# Evaluate the model
mse = mean_squared_error(ytest, y_pred)
rmse = np.sqrt(mse)
print("Root Mean Square Error:", rmse)

# Calculate the R^2 score (coefficient of determination)
accuracy = ada_regressor.score(xtest, ytest)
print("R^2 Score:", accuracy*100)

print("training score = ",ada_regressor.score(xtrain,ytrain))
```

```
Root Mean Square Error: 0.0377194013811472
R^2 Score: 99.96479307606988
training score =  0.9996615319816969
```

METHODS & APPROACHES



METHODS & APPROACHES

Linear Regression

- A statistical method used for prediction
- It provides simple and interpretable way to understand the relationship between the variables.
- The Accuracy is 100%

Linear Regression

```
#To check if there is overfitting or under fitting
print("training score = ",lr.score(xtrain,ytrain))
print("testing score = ",lr.score(xtest,ytest))
```

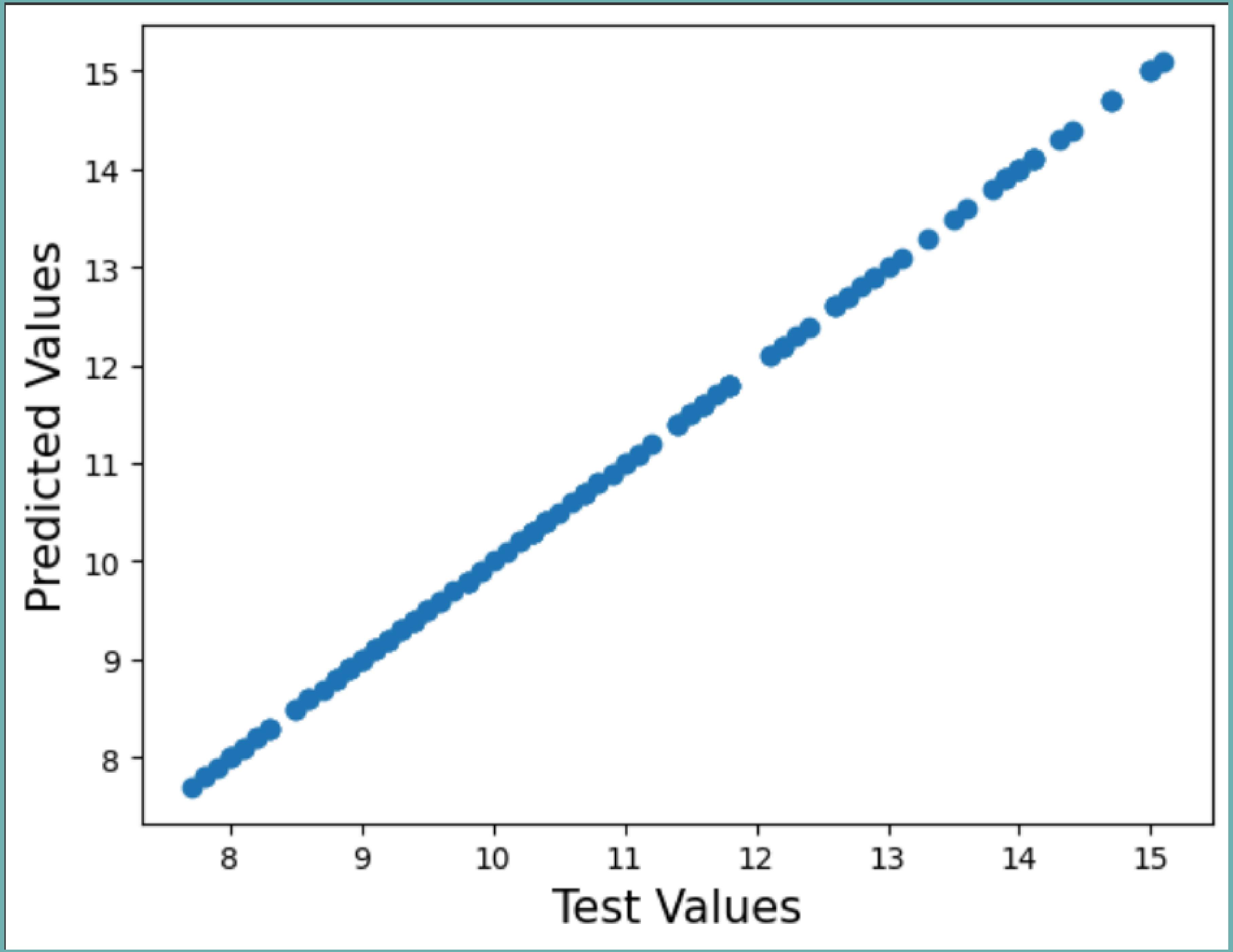
```
training score =  1.0
testing score =  1.0
```

```
ypred = lr.predict(xtest)
```

```
r2_score(ytest, ypred)
```

```
1.0
```

METHODS & APPROACHES



METHODS & APPROACHES

Artificial Neural Network

- particularly useful because of their ability to model complex relationships between input variables and the target output.
- The Accuracy is 99%

Neural Network

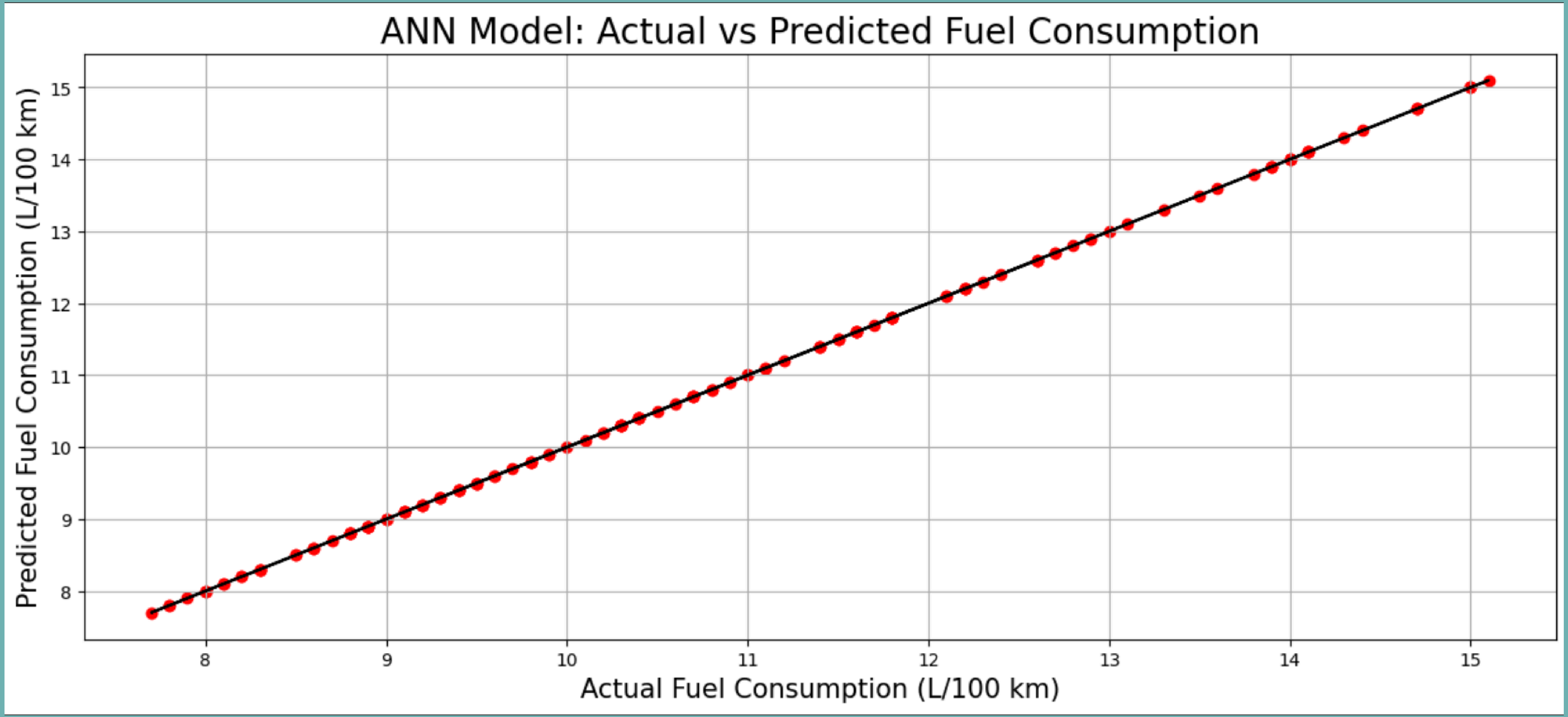
```
from sklearn.neural_network import MLPRegressor
nn = MLPRegressor(hidden_layer_sizes=(900,), activation='relu', solver='adam', max_iter=1000)
nn.fit(xtrain, ytrain)
```

```
MLPRegressor
MLPRegressor(hidden_layer_sizes=(900,), max_iter=1000)
```

```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
# Accuracy
accuracy_nn = nn.score(xtest, ytest) * 100
print("Accuracy:", accuracy_nn)
# RMSE
rmse_nn = np.sqrt(mean_squared_error(ytest, ypred))
print("RMSE:", rmse_nn)
# MAE
mae_nn = mean_absolute_error(ytest, ypred)
print("MAE:", mae_nn)
# R2
r2_nn = r2_score(ytest, ypred)
print("R2:", r2_nn)
```

```
Accuracy: 99.39959865306272
RMSE: 1.8513413036726634e-15
MAE: 1.5435928397547003e-15
R2: 1.0
```

METHODS & APPROACHES



METHODS & APPROACHES

Random Forest

- The random forest method is an ensemble learning technique used primarily for classification and regression tasks.
- The Accuracy is 99%

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
```

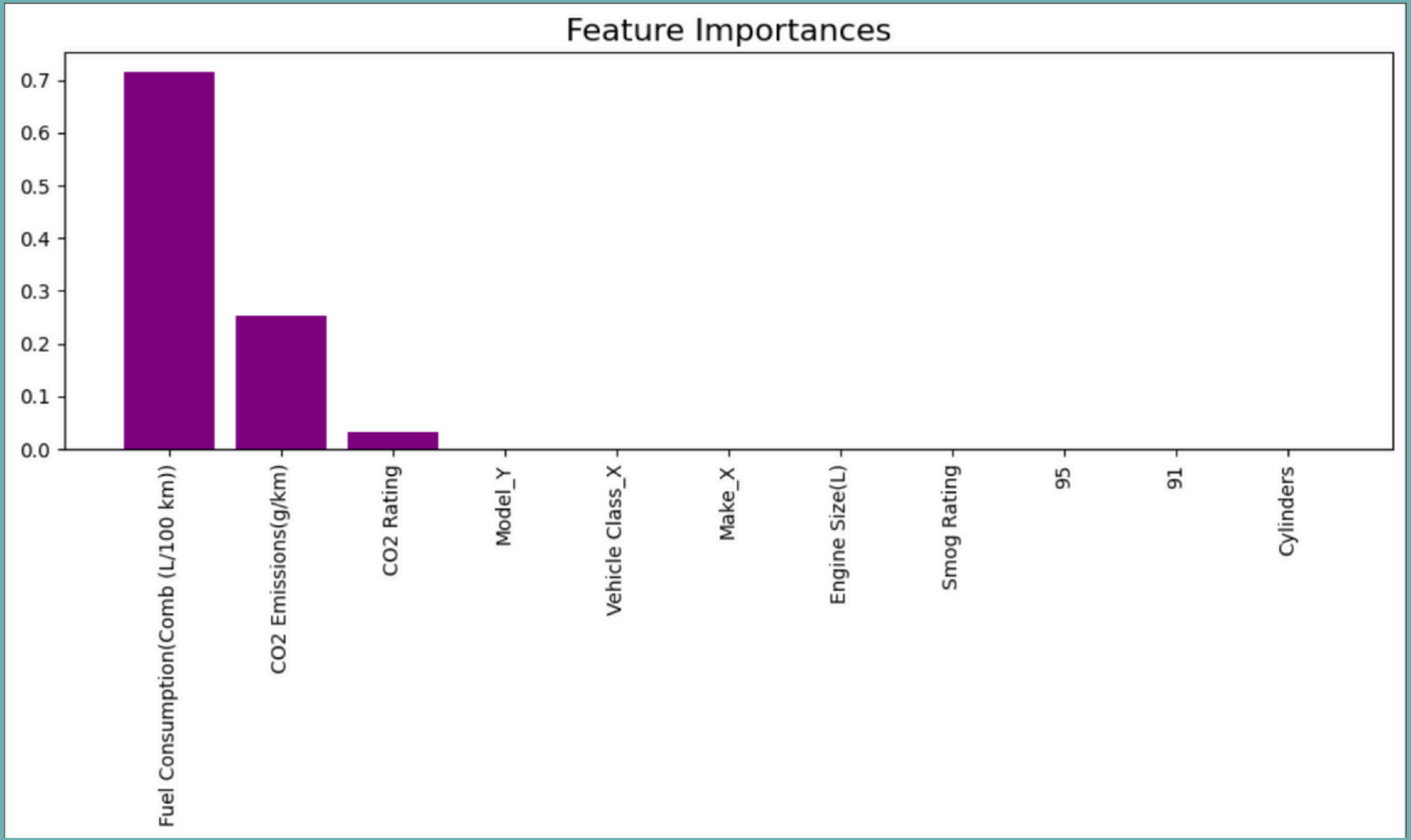
```
no_of_decision_tree = [10,20,30,40,50,60,70,80,90,100]
max_no_of_features = ['sqrt','log2']
max_depth = [6,7,8,9,10,11,12,13,14,15]
criterion_of_decision_tree = ["squared_error", "poisson"]
min_sample_split=[2,3,4,5,6]
```

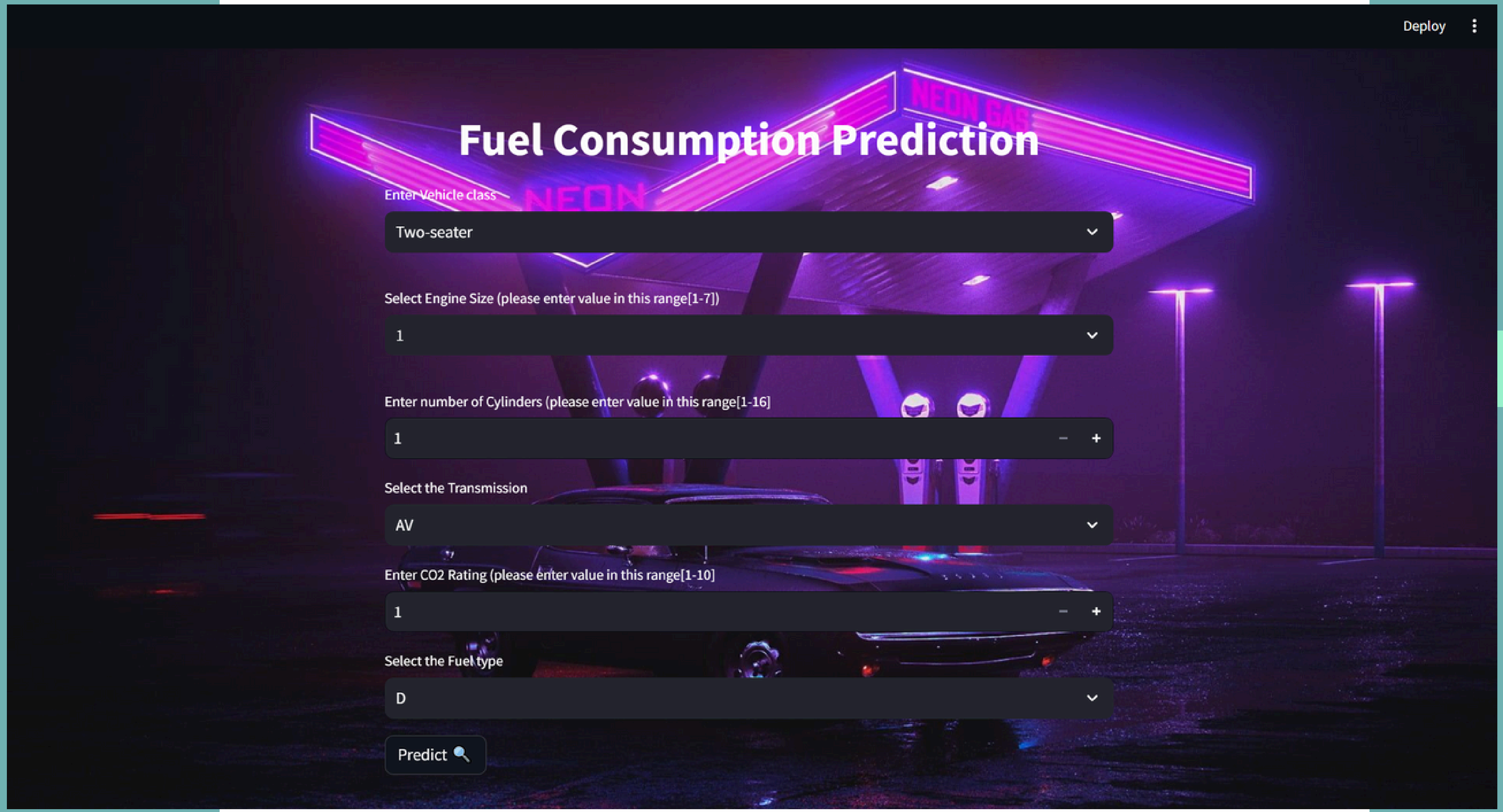
```
random_grid = {
      'n_estimators': no_of_decision_tree,
      'max_features': max_no_of_features,
      'max_depth': max_depth,
      'criterion': criterion_of_decision_tree,
      'min_samples_split': min_sample_split
}
```

```
from sklearn.model_selection import RandomizedSearchCV
rscv = RandomizedSearchCV(estimator = rf , param_distributions = random_grid , n_iter = 25 , cv = 5 ,n_jobs=-1)
rscv.fit(xtrain, ytrain)
```

```
RandomizedSearchCV
estimator: RandomForestRegressor
RandomForestRegressor
```

METHODS & APPROACHES





PROJECT OUTCOME

NEXT STEPS

DATA PROCESSING

Include more sophisticated data processing techniques.

LARGER DATESETS

Implement oour models on larger datasets

OUR WEBSITE

Improve our website to include an AI chatbot

EXPERIMENT

Experiment with various other factors and models.

CONCLUSION

STRIVE FOR SUSTAINABALITY