Academic Year	Module	Assessment Number	Assessment Type
2025	5CS037-Concept and Technologies of AI	Final	Report

REGRESSION TASK

Student Id	: 2408620
Student Name	: Rodik Awal
Section	: L5CG7
Module Leader	: Siman Giri
Tutor	: Siman Giri
Submitted on	: 11/02/2025

Contents

Regression Report

Introduction

In line with SDG 13: Climate Action, this report analyzes a regression task that was carried out on a dataset pertaining to the prediction of CO2 emissions. Vehicle specifications and their effects on the environment are included in the dataset. Data preprocessing, exploratory data analysis (EDA), model building, and evaluation using various machine learning models are all part of the task.

Dataset

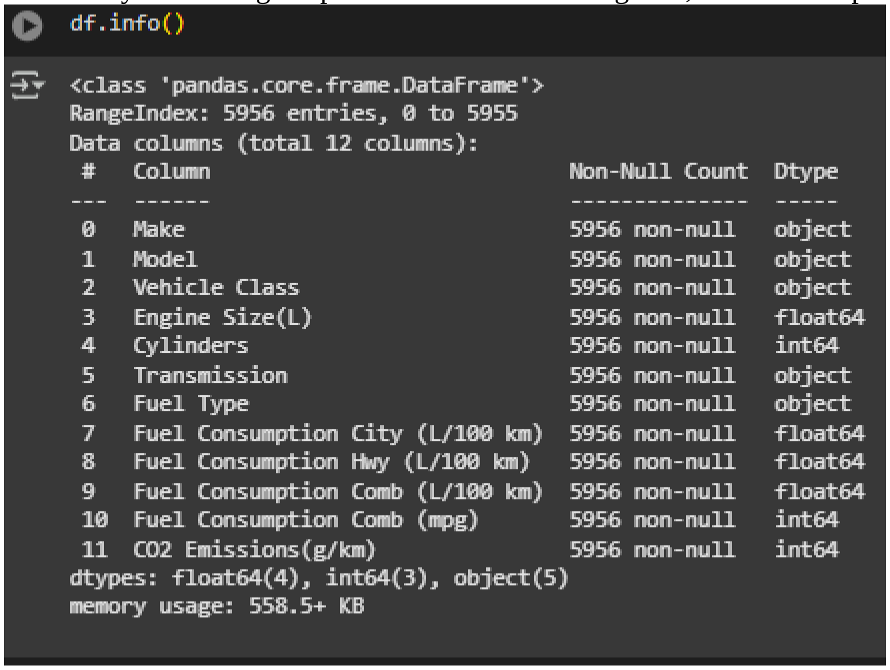
The regression task makes use of the "CO2 Emissions.csv" dataset. The target variable is CO2 Emissions (g/km), a continuous numerical variable that depends on the independent variables (features). It includes features like engine size, fuel consumption, fuel type, transmission type, and CO2 emissions.

Objective

By analyzing metrics like mean squared error (MSE), r-squared (R^2), and other important performance indicators, the objective is to create the best predictive model.

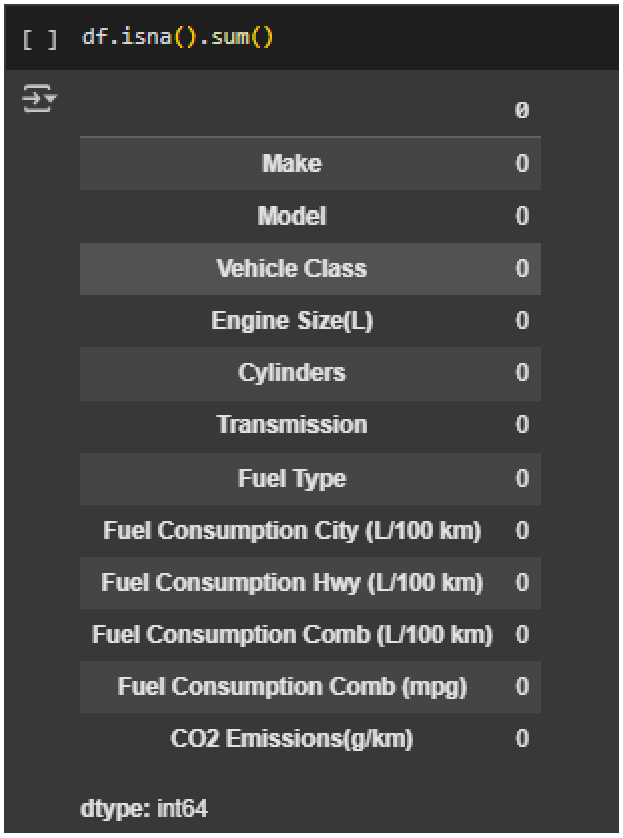
Data Exploration

To identify and manage duplicate values and missing data, basic data exploration is carried out.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5956 entries, 0 to 5955
Data columns (total 12 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Make                              5956 non-null   object 
 1   Model                             5956 non-null   object 
 2   Vehicle Class                     5956 non-null   object 
 3   Engine Size(L)                    5956 non-null   float64
 4   Cylinders                         5956 non-null   int64  
 5   Transmission                      5956 non-null   object 
 6   Fuel Type                         5956 non-null   object 
 7   Fuel Consumption City (L/100 km)  5956 non-null   float64
 8   Fuel Consumption Hwy (L/100 km)   5956 non-null   float64
 9   Fuel Consumption Comb (L/100 km)  5956 non-null   float64
 10  Fuel Consumption Comb (mpg)       5956 non-null   int64  
 11  CO2 Emissions(g/km)               5956 non-null   int64  
dtypes: float64(4), int64(3), object(5)
memory usage: 558.5+ KB
```



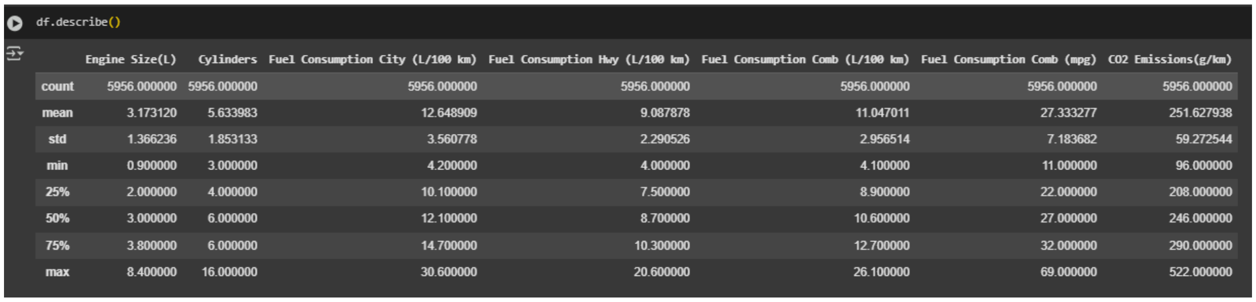
```
[ ]  df.isna().sum()
```

	0
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

Exploratory Data Analysis (EDA) and Visualization

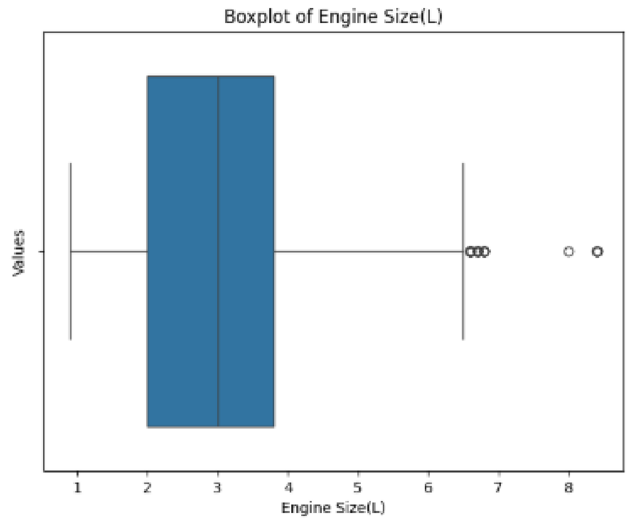
Statistical Summary

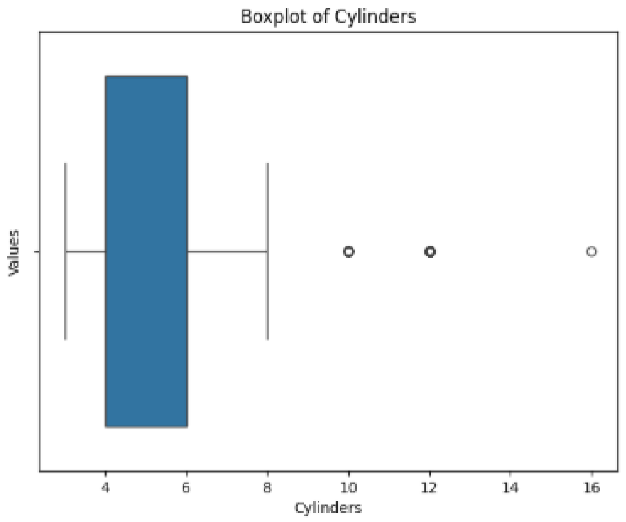


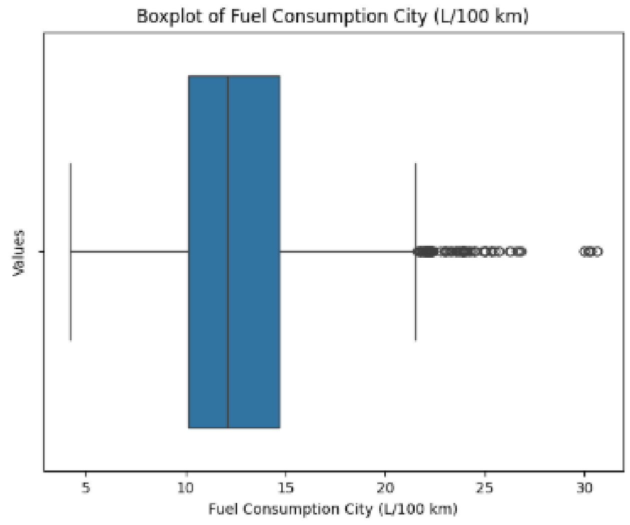
```
df.describe()
```

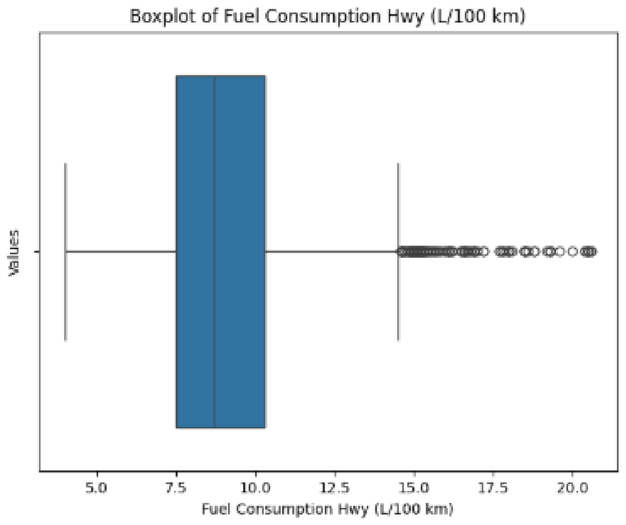
	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
count	5956.000000	5956.000000	5956.000000	5956.000000	5956.000000	5956.000000	5956.000000
mean	3.173120	5.633983	12.648909	9.087878	11.047011	27.333277	251.627938
std	1.366236	1.853133	3.560778	2.290526	2.956514	7.183682	59.272544
min	0.900000	3.000000	4.200000	4.000000	4.100000	11.000000	96.000000
25%	2.000000	4.000000	10.100000	7.500000	8.900000	22.000000	208.000000
50%	3.000000	6.000000	12.100000	8.700000	10.600000	27.000000	246.000000
75%	3.800000	6.000000	14.700000	10.300000	12.700000	32.000000	290.000000
max	8.400000	16.000000	30.600000	20.600000	26.100000	69.000000	522.000000

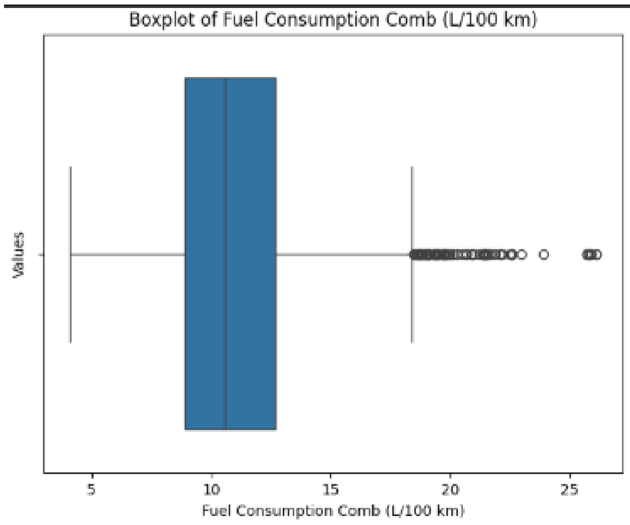
Data Cleaning

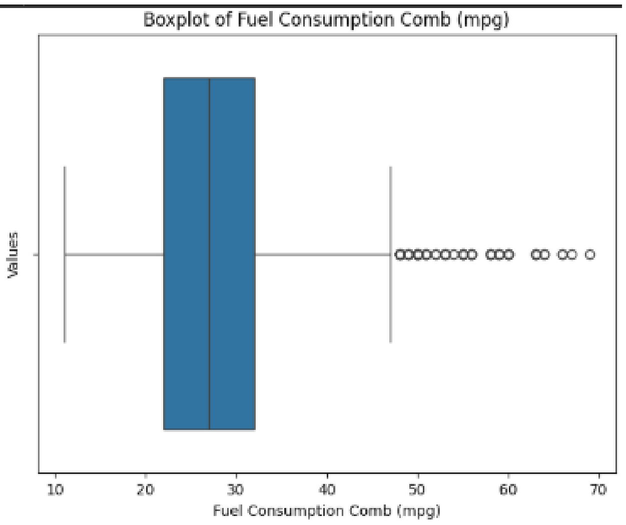


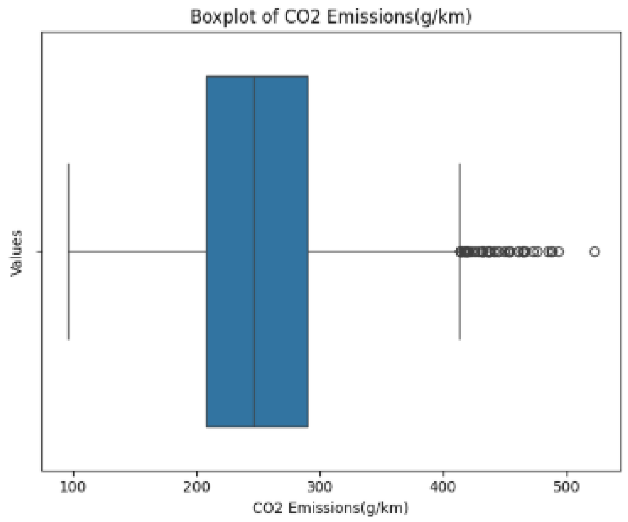




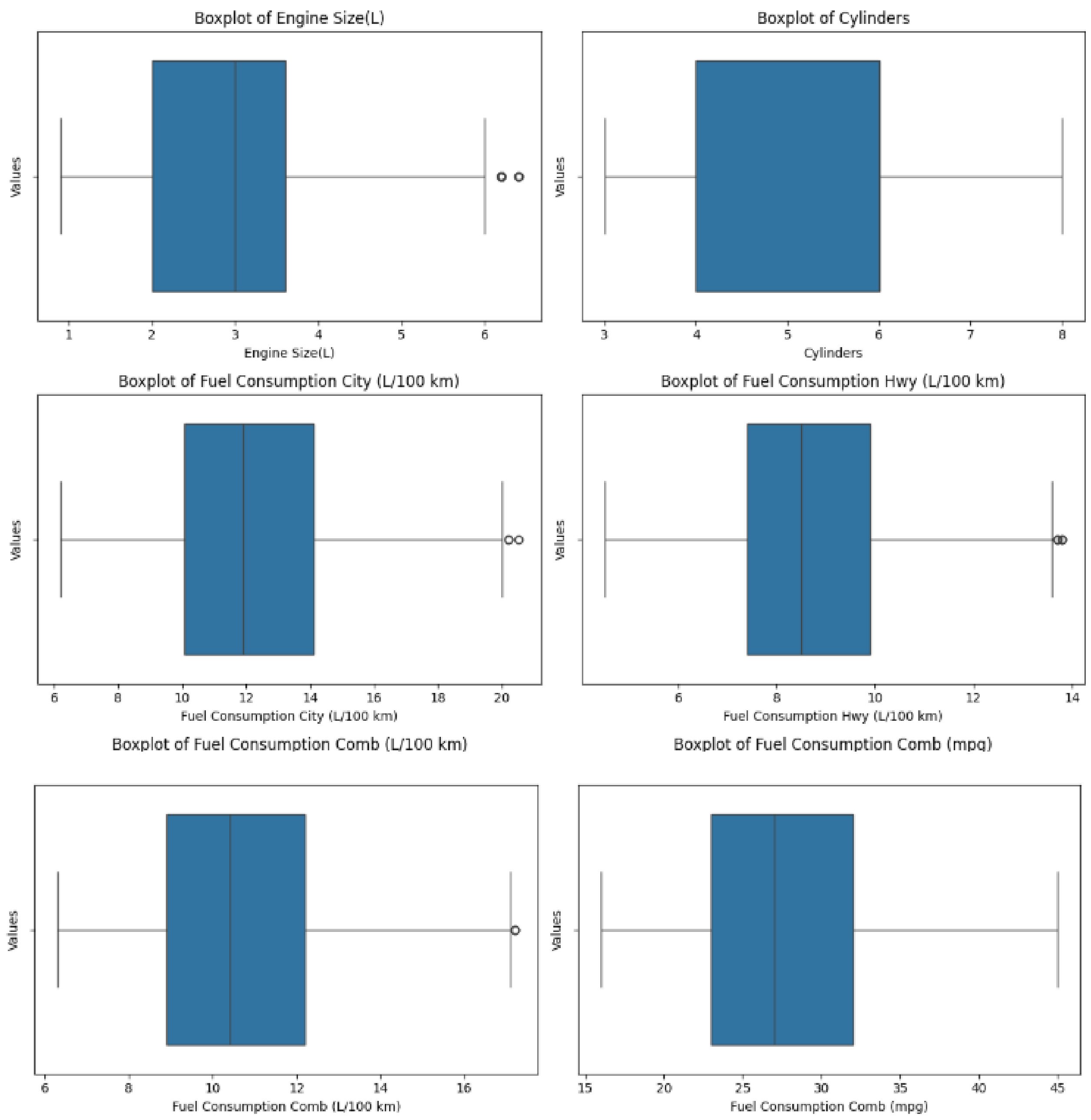






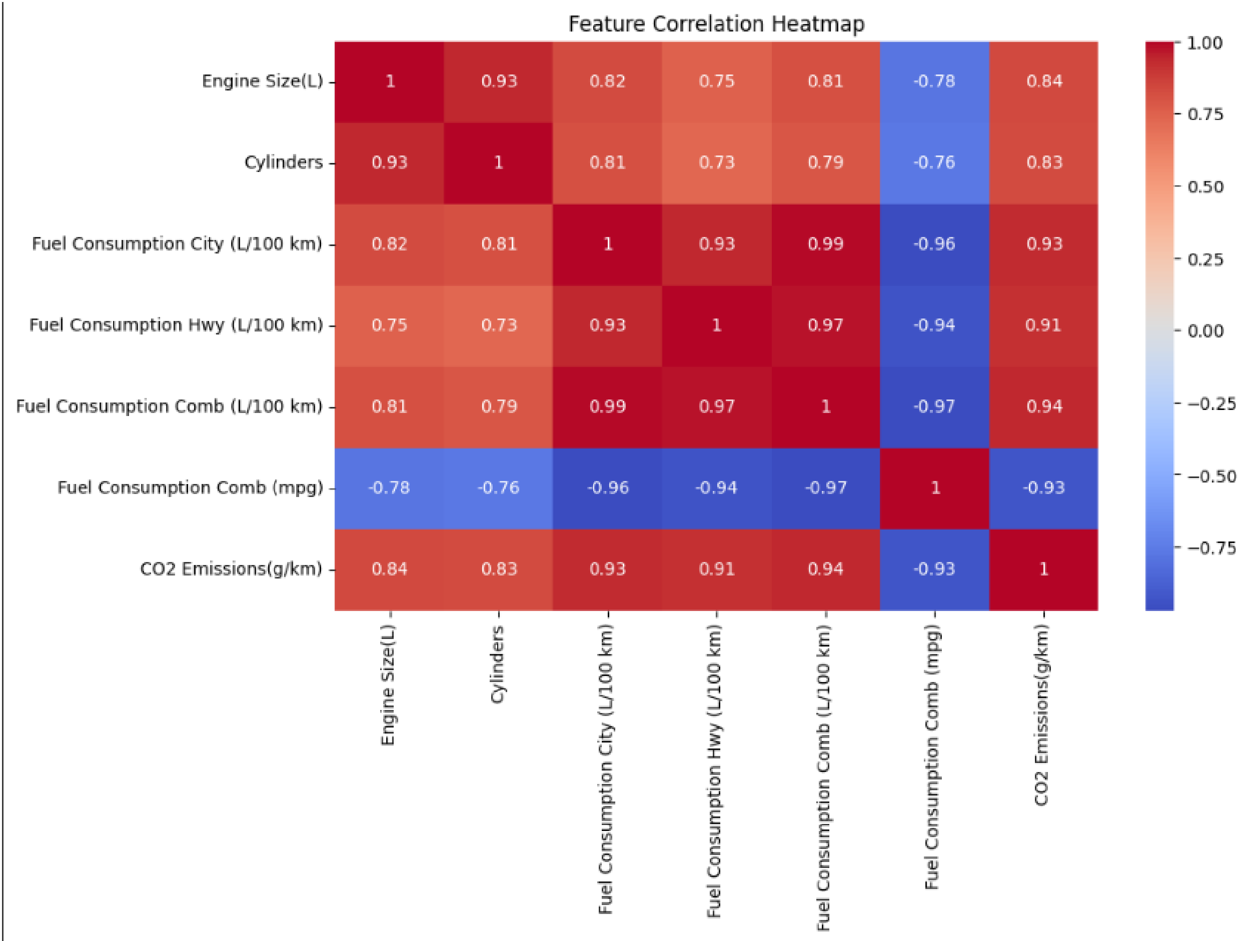


The box plot after filtering out the outliers.



Exploration Data Analysis(EDA)

correlation between the goal variable and the characteristics.

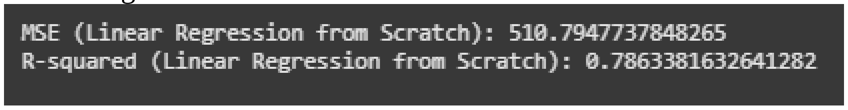


	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
Engine Size(L)	1	0.93	0.82	0.75	0.81	-0.78	0.84
Cylinders	0.93	1	0.81	0.73	0.79	-0.76	0.83
Fuel Consumption City (L/100 km)	0.82	0.81	1	0.93	0.99	-0.96	0.93
Fuel Consumption Hwy (L/100 km)	0.75	0.73	0.93	1	0.97	-0.94	0.91
Fuel Consumption Comb (L/100 km)	0.81	0.79	0.99	0.97	1	-0.97	0.94
Fuel Consumption Comb (mpg)	-0.78	-0.76	-0.96	-0.94	-0.97	1	-0.93
CO2 Emissions(g/km)	0.84	0.83	0.93	0.91	0.94	-0.93	1

- Vehicles with higher fuel consumption emit much more CO2 emissions, as evidenced by the strong link between fuel consumption (L/100 km) and CO2 emissions (g/km).
- Cylinders and CO2 Emissions have a strong correlation with Engine Size (L), indicating that larger engines with more cylinders produce more CO2.

Module Building

Linear Regression from scratch.



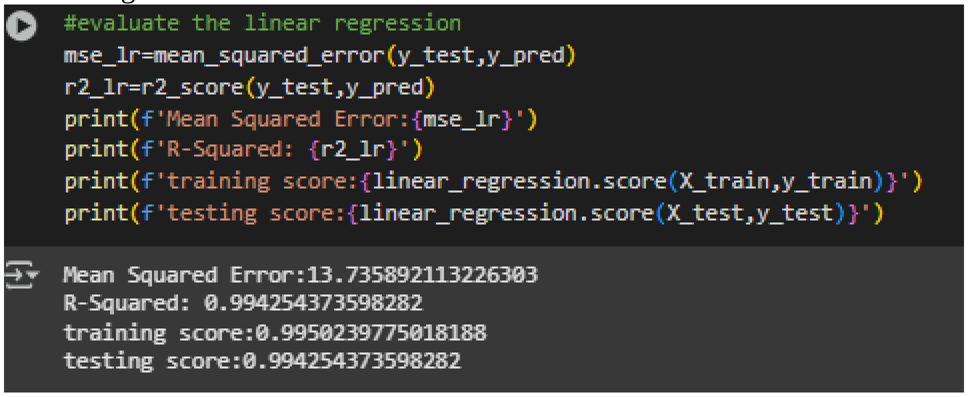
```
MSE (Linear Regression from Scratch): 510.7947737848265
R-squared (Linear Regression from Scratch): 0.7863381632641282
```

Module Evaluating:
The model was assessed using the metrics R-squared and MSE (Mean Squared Error):

- The average squared difference between actual and predicted values is measured by the MSE metric. Better performance is indicated by a lower MSE. The MSE for the model is 510.794773784265.
- The R-squared score shows how much of the variance in the target variable can be accounted for by the independent variables. A better fit is indicated by a score nearer 1. The R-squared score for the model is 0.7863381632641282.
  Build Two models

The models that perform better are checked using the random forest regressor and linear regression.

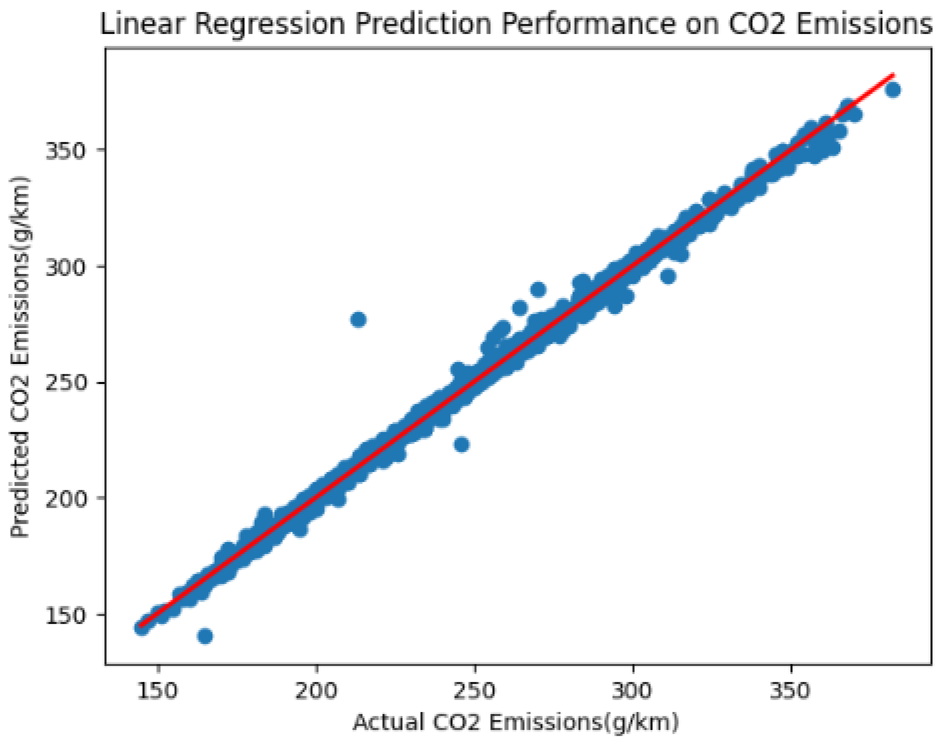
Linear Regression Evaluation



```
#evaluate the linear regression
mse_lr=mean_squared_error(y_test,y_pred)
r2_lr=r2_score(y_test,y_pred)
print(f'Mean Squared Error:{mse_lr}')
print(f'R-Squared: {r2_lr}')
print(f'training score:{linear_regression.score(X_train,y_train)}')
print(f'testing score:{linear_regression.score(X_test,y_test)}')
```

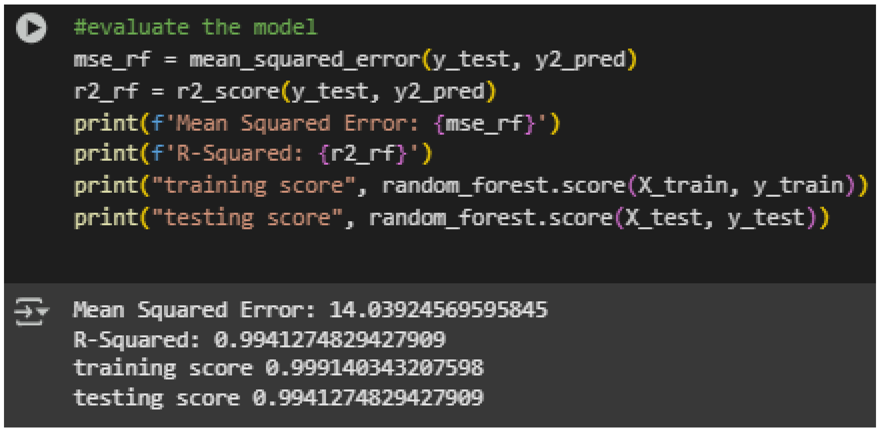
```
Mean Squared Error:13.735892113226303
R-Squared: 0.994254373598282
training score:0.9950239775018188
testing score:0.994254373598282
```

The model does not show significant overfitting or under fitting, as evidenced by the nearly identical training and testing scores. On the other hand, the low Mean Squared Error (MSE) indicates that there is a significant prediction error. In spite of this, the model explains a significant amount of the variance in the data, as indicated by the respectable R2 score.



The model works well overall, as seen by a strong match between anticipated and actual values. Outliers or nonlinear interactions between features and the target variable may be the cause of its difficulties with greater values. The model's capacity to predict for greater CO2 emission values may be impacted by these considerations.

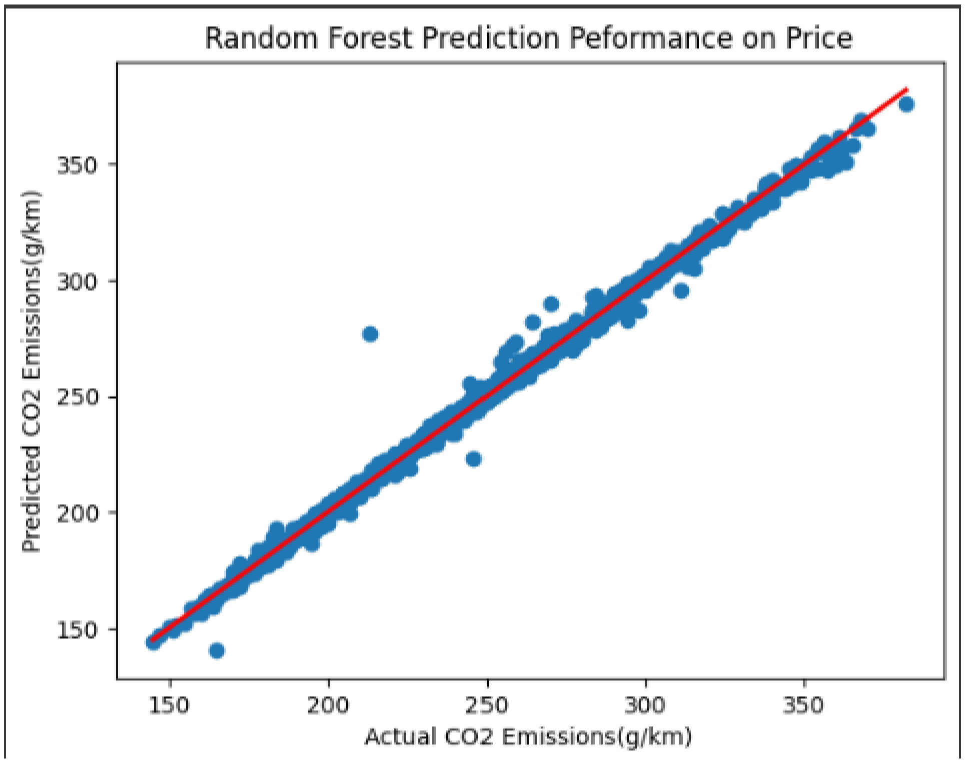
Evaluation of Random Forest Regressor

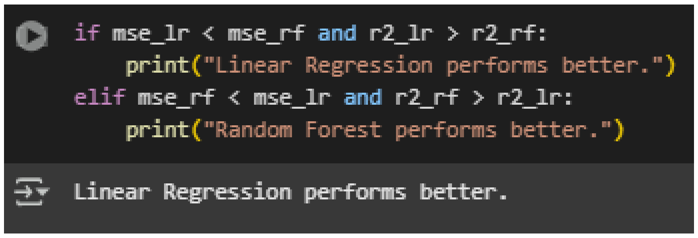


```
#evaluate the model
mse_rf = mean_squared_error(y_test, y2_pred)
r2_rf = r2_score(y_test, y2_pred)
print(f'Mean Squared Error: {mse_rf}')
print(f'R-Squared: {r2_rf}')
print("training score", random_forest.score(X_train, y_train))
print("testing score", random_forest.score(X_test, y_test))
```

```
Mean Squared Error: 14.03924569595845
R-Squared: 0.9941274829427909
training score 0.999140343207598
testing score 0.9941274829427909
```

Since this model's R-squared score is nearer 1, it performs better and shows a better fit. The Mean Squared Error (MSE) is also less than that of the linear regression model. But the clear difference between the test and training scores points to overfitting, which could make the model less able to generalize to new data.





```
if mse_lr < mse_rf and r2_lr > r2_rf:
    print("Linear Regression performs better.")
elif mse_rf < mse_lr and r2_rf > r2_lr:
    print("Random Forest performs better.")
```

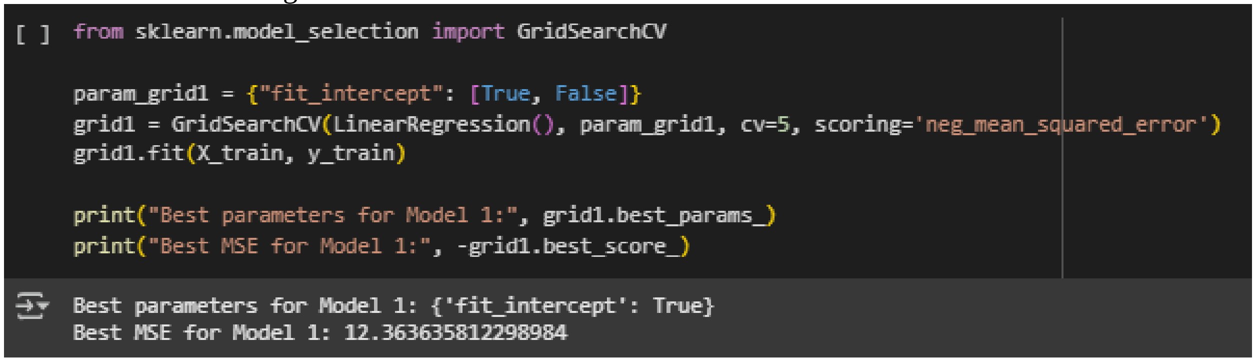
```
Linear Regression performs better.
```

The evaluation shows linear regression has performed better than Random Forest Regressor

Hyper-parameter

GridSearchCV is used for hyper parameter optimization to enhance model performance.

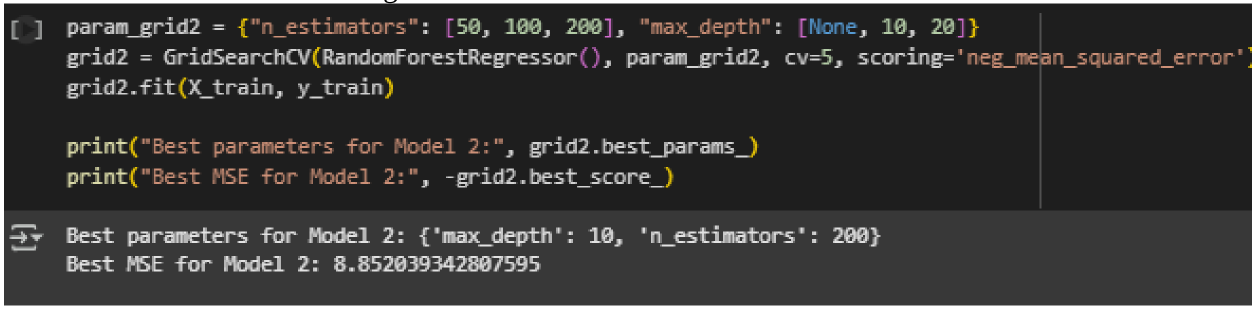
For Model 1: Linear Regression



```
[ ] from sklearn.model_selection import GridSearchCV

    param_grid1 = {"fit_intercept": [True, False]}
    grid1 = GridSearchCV(LinearRegression(), param_grid1, cv=5, scoring='neg_mean_squared_error')
    grid1.fit(X_train, y_train)

    print("Best parameters for Model 1:", grid1.best_params_)
    print("Best MSE for Model 1:", -grid1.best_score_)
```

```
Best parameters for Model 1: {'fit_intercept': True}
Best MSE for Model 1: 12.363635812298984
```

For model 2: Random Forest Regressor



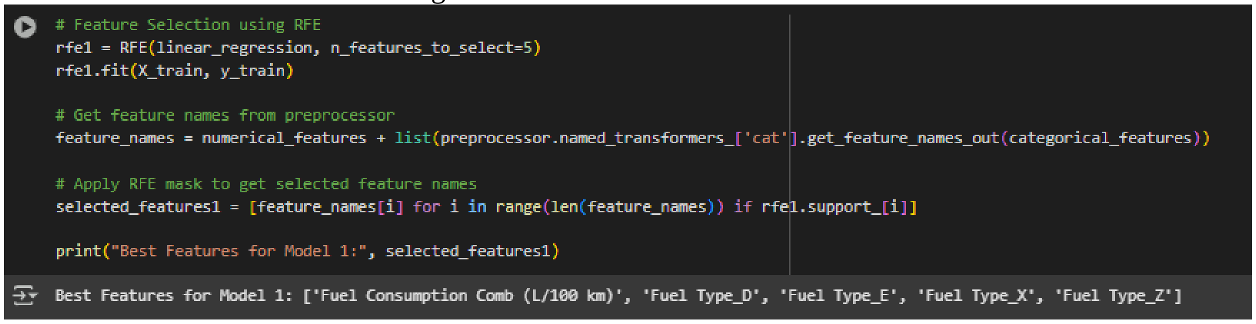
```
[ ] param_grid2 = {"n_estimators": [50, 100, 200], "max_depth": [None, 10, 20]}
    grid2 = GridSearchCV(RandomForestRegressor(), param_grid2, cv=5, scoring='neg_mean_squared_error'
    grid2.fit(X_train, y_train)

    print("Best parameters for Model 2:", grid2.best_params_)
    print("Best MSE for Model 2:", -grid2.best_score_)
```

```
Best parameters for Model 2: {'max_depth': 10, 'n_estimators': 200}
Best MSE for Model 2: 8.852039342807595
```

Feature Selection

To enhance model performance, the best features are chosen using Recursive Feature Elimination (RFE).

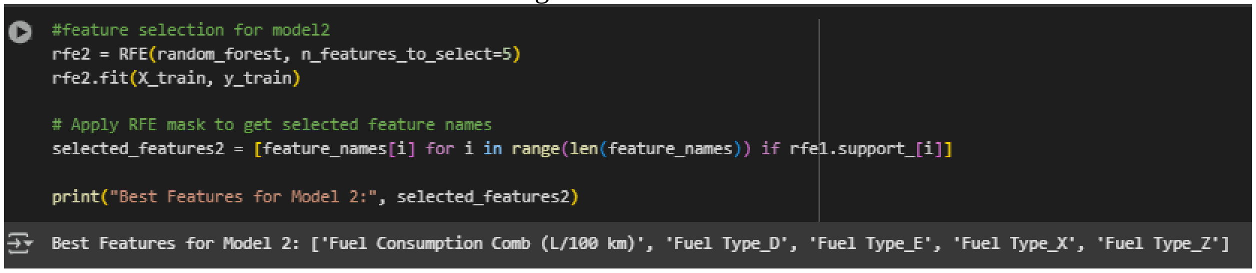
Best Feature for Model 1: Linear Regression



```
# Feature Selection using RFE
rfe1 = RFE(linear_regression, n_features_to_select=5)
rfe1.fit(X_train, y_train)

# Get feature names from preprocessor
feature_names = numerical_features + list(preprocessor.named_transformers_['cat'].get_feature_names_out(categorical_features))

# Apply RFE mask to get selected feature names
selected_features1 = [feature_names[i] for i in range(len(feature_names)) if rfe1.support_[i]]

print("Best Features for Model 1:", selected_features1)
```

```
Best Features for Model 1: ['Fuel Consumption Comb (L/100 km)', 'Fuel Type_D', 'Fuel Type_E', 'Fuel Type_X', 'Fuel Type_Z']
```

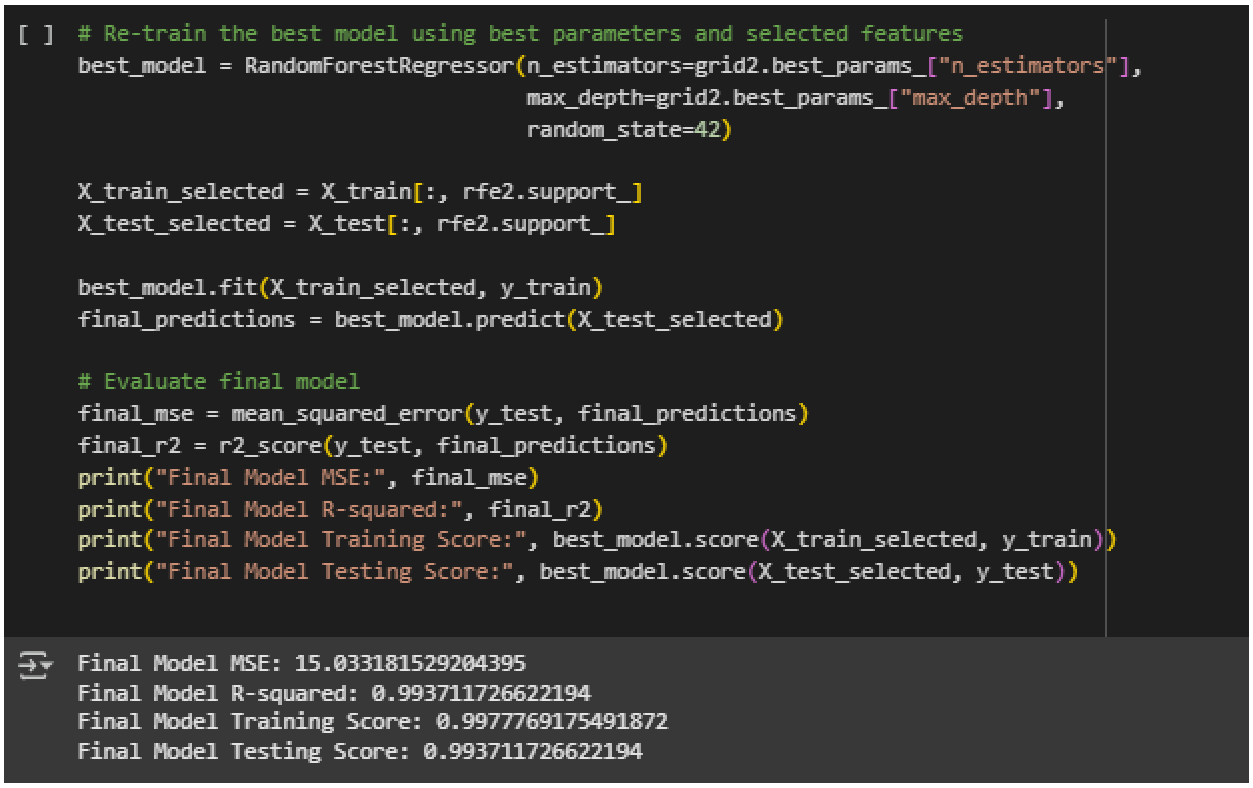
Best Feature for Model 2: Random Forest Regressor



```
#feature selection for model2
rfe2 = RFE(random_forest, n_features_to_select=5)
rfe2.fit(X_train, y_train)

# Apply RFE mask to get selected feature names
selected_features2 = [feature_names[i] for i in range(len(feature_names)) if rfe1.support_[i]]

print("Best Features for Model 2:", selected_features2)
```

```
Best Features for Model 2: ['Fuel Consumption Comb (L/100 km)', 'Fuel Type_D', 'Fuel Type_E', 'Fuel Type_X', 'Fuel Type_Z']
```

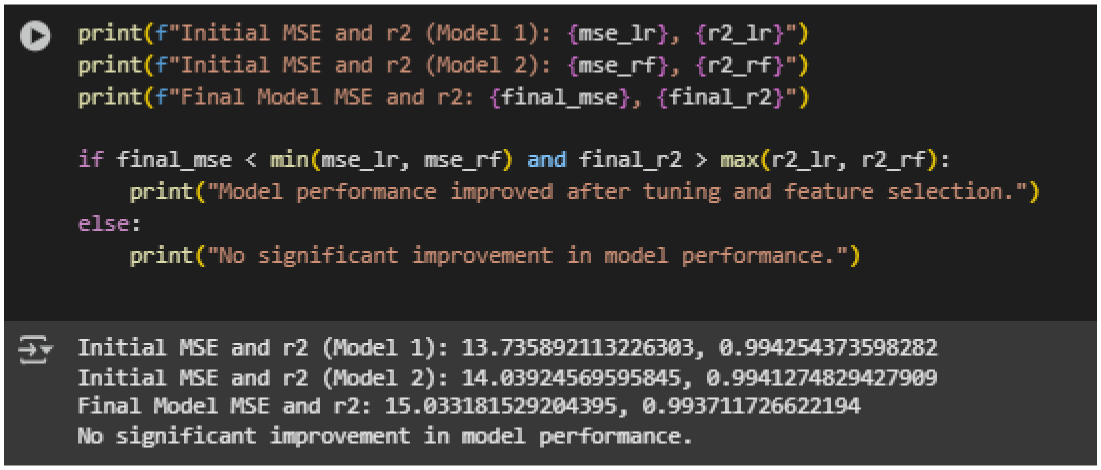
Final Model



```
[ ] # Re-train the best model using best parameters and selected features
    best_model = RandomForestRegressor(n_estimators=grid2.best_params_["n_estimators"],
                                       max_depth=grid2.best_params_["max_depth"],
                                       random_state=42)

    X_train_selected = X_train[:, rfe2.support_]
    X_test_selected = X_test[:, rfe2.support_]

    best_model.fit(X_train_selected, y_train)
    final_predictions = best_model.predict(X_test_selected)

    # Evaluate final model
    final_mse = mean_squared_error(y_test, final_predictions)
    final_r2 = r2_score(y_test, final_predictions)
    print("Final Model MSE:", final_mse)
    print("Final Model R-squared:", final_r2)
    print("Final Model Training Score:", best_model.score(X_train_selected, y_train))
    print("Final Model Testing Score:", best_model.score(X_test_selected, y_test))
```

```
Final Model MSE: 15.033181529204395
Final Model R-squared: 0.993711726622194
Final Model Training Score: 0.9977769175491872
Final Model Testing Score: 0.993711726622194
```

Result



```
print(f"Initial MSE and r2 (Model 1): {mse_lr}, {r2_lr}")
print(f"Initial MSE and r2 (Model 2): {mse_rf}, {r2_rf}")
print(f"Final Model MSE and r2: {final_mse}, {final_r2}")

if final_mse < min(mse_lr, mse_rf) and final_r2 > max(r2_lr, r2_rf):
    print("Model performance improved after tuning and feature selection.")
else:
    print("No significant improvement in model performance.")
```

```
Initial MSE and r2 (Model 1): 13.735892113226303, 0.994254373598282
Initial MSE and r2 (Model 2): 14.03924569595845, 0.9941274829427909
Final Model MSE and r2: 15.033181529204395, 0.993711726622194
No significant improvement in model performance.
```

At first, with a higher r2 score and a lower Mean Squared Error (MSE), the random forest regressor outperformed the linear regression model. The performance of the final model was assessed following the tuning procedure and feature selection. According to the results, there was no noticeable rise in the final model's MSE or r2 score over the original models. As a result, it was determined that the model's performance was not significantly improved by the tuning and feature selection.

Conclusion

First, with a higher r2 score and a lower MSE, the random forest regressor performed better than linear regression. However, performance metrics did not significantly improve with the final model following feature selection and hyper parameter tuning. The analysis highlights how important it is to regularly evaluate the model and how different methods might be required to increase predictive accuracy.