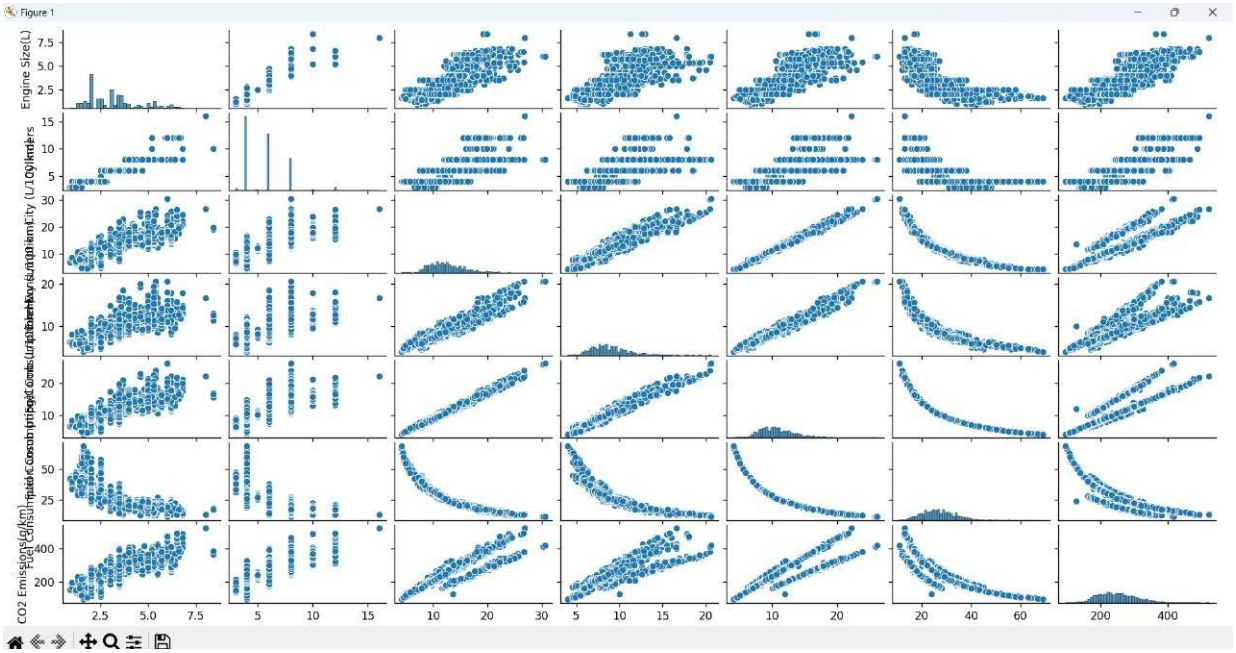
Report:

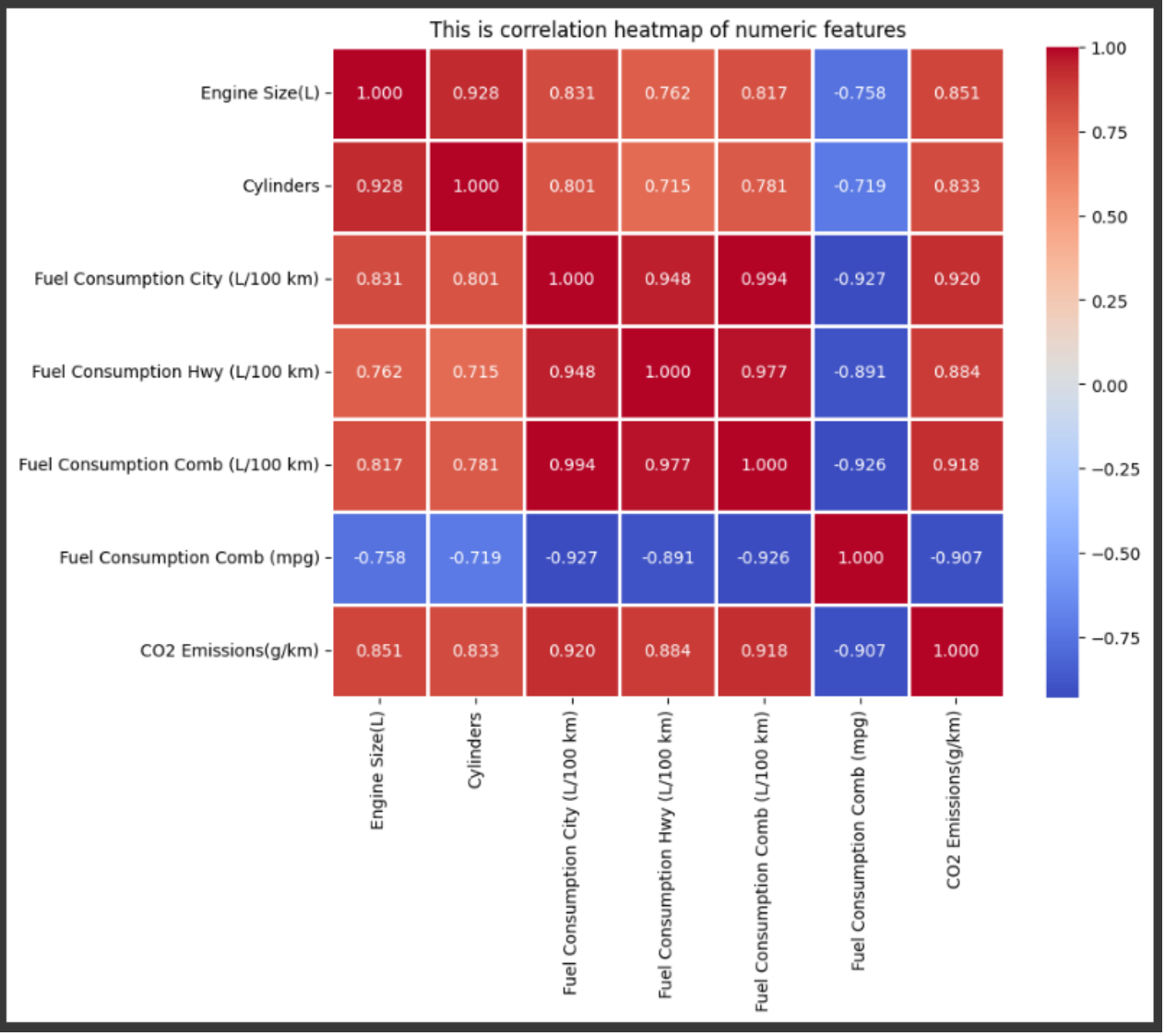
Name	ID
Ali Bedair	20206036
Youssef Abdellah	20215043

Analysis:

```
sns.pairplot(dataset[numeric_features])
plt.show()
```



```
Matrix_Correlation = dataset[numeric_features].corr()
plt.figure(figsize=(9, 7))
plt.title("This is correlation heatmap of numeric features")
sns.heatmap(Matrix_Correlation, annot=True, fmt='.3f', linewidths=1, cmap='coolwarm')
plt.show()
```



```
print("\nMissing values in each column:")
print(dataset.isnull().sum())

numeric_features = dataset.select_dtypes(include=['float64', 'int64']).columns
summary_info = dataset.describe().to_string()
print("\nSummary info of Dataset analysis:")
print(summary_info)
```

```
Missing values in each column:
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
Emission Class                    0
dtype: int64

Summary info of Dataset analysis:
       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     7385.000000  7385.000000                       7385.000000                      7385.000000                       7385.000000                  7385.000000          7385.000000
mean         3.160068     5.615030                         12.556534                         9.041706                         10.975071                    27.481652           250.584699
std          1.354170     1.828307                          3.500274                         2.224456                          2.892506                     7.231879            58.512679
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%          3.700000     6.000000                         14.600000                        10.200000                         12.600000                    32.000000           288.000000
max          8.400000    16.000000                         30.600000                        20.600000                         26.100000                    69.000000           522.000000
```

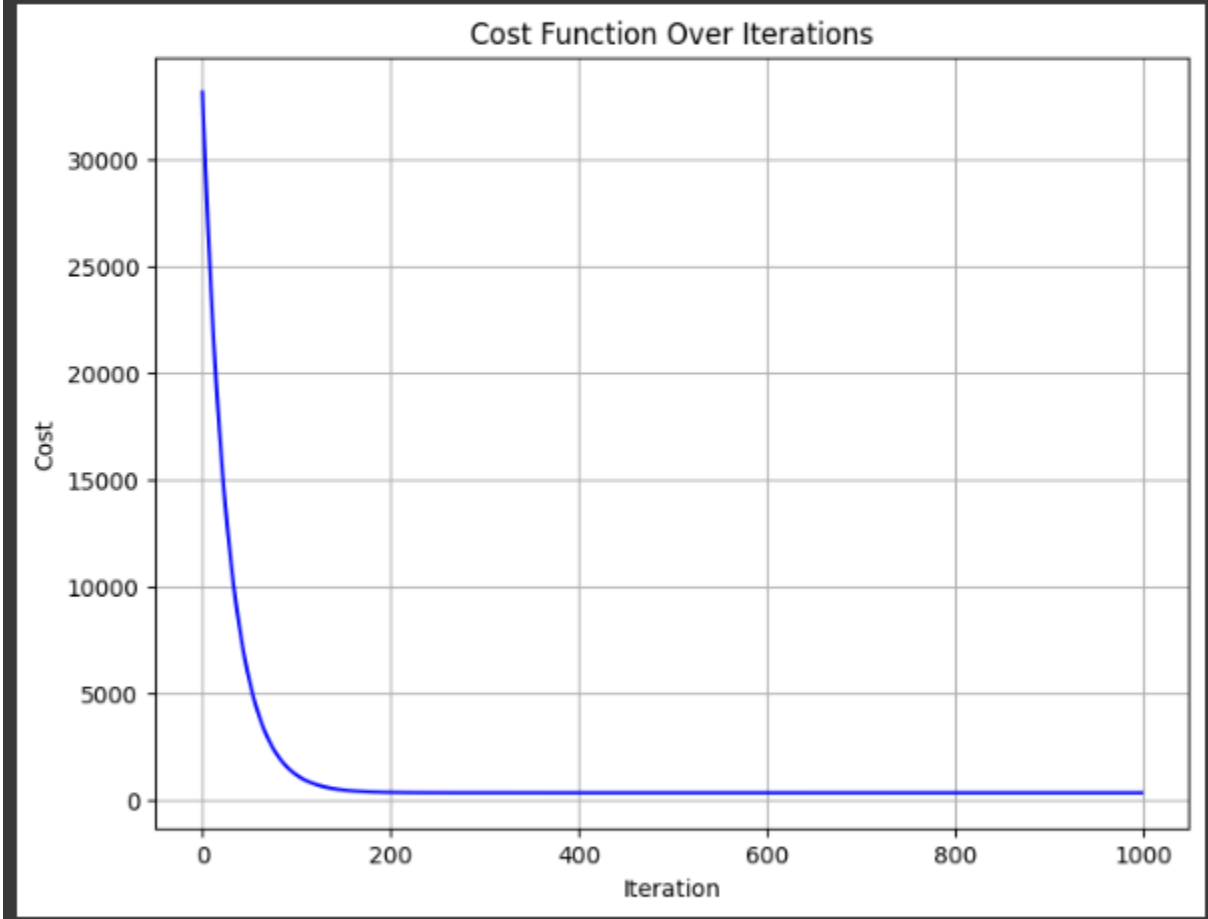
Selected features:

```
mean = independent_features_train.mean(axis=0)
standard_deviation = independent_features_train.std(axis=0)
Scaled_numeric_train = (independent_features_train - mean) / standard_deviation
Scaled_numeric_test = (independent_features_test - mean) / standard_deviation


print("Scaled Training Features (First 5 rows):\n", Scaled_numeric_train.head())
print("Scaled Test Features (First 5 rows):\n", Scaled_numeric_test.head())
```

```
Scaled Training Features (First 5 rows):
      Engine Size(L)  Fuel Consumption City (L/100 km)
6214        2.098659                          1.684512
900         1.212508                          1.079238
77         -0.855178                         -0.621294
4488        0.917124                          0.502786
6955        0.474048                         -0.073665
Scaled Test Features (First 5 rows):
      Engine Size(L)  Fuel Consumption City (L/100 km)
5340       -0.559795                         -0.390713
3783       -0.559795                         -0.967165
1582        0.252511                         -0.073665
309         0.326356                          0.185738
1849        1.138662                          0.099270
```

Cost function per iterations:

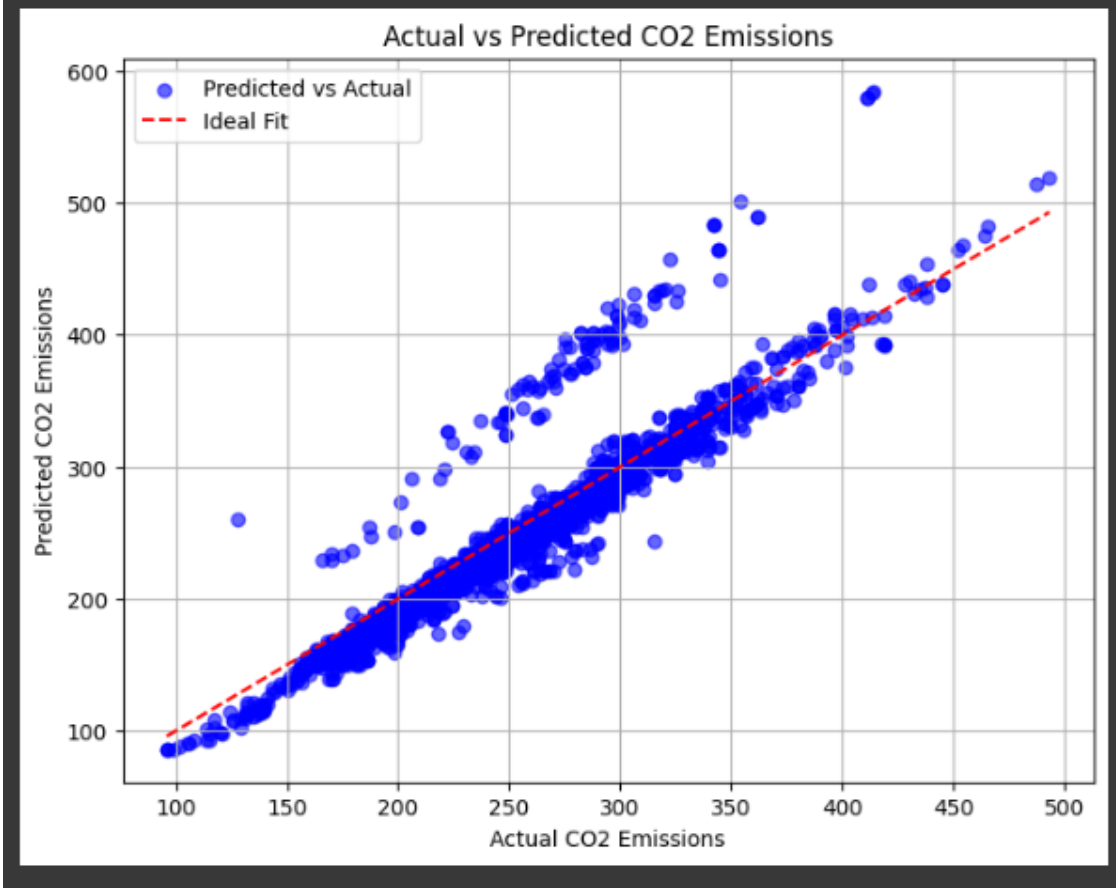
```
# Making predictions on the test set
predictions_test = X_test.dot(theta)
print("\nFirst 6 predictions on the test set:")
print(predictions_test[:6])


# test set using Scikit-learn's R2 score
r2 = r2_score(emission_test_values, predictions_test)
r2 = round(r2,5)
print(f"\nR2 score on the test set: {r2}")
```




```
Final parameter values (theta):
[ 1.77406186  4.71824917 18.30394381]

First 6 predictions on the test set:
[218.10203057 181.49414294 243.42644285 260.3718172  260.07070814
 174.64439034]

R2 score on the test set: 0.77806
```

Actual vs Predict CO2 emissions:

```
# Plot actual vs predicted values
plt.figure(figsize=(8, 6))
plt.scatter(emission_test_values, predictions_test, color='blue', alpha=0.6, label="Predicted vs Actual")
plt.plot([emission_test_values.min(), emission_test_values.max()],
         [emission_test_values.min(), emission_test_values.max()],
         color='red', linestyle='--', label="Ideal Fit")
plt.title("Actual vs Predicted CO2 Emissions")
plt.xlabel("Actual CO2 Emissions")
plt.ylabel("Predicted CO2 Emissions")
plt.legend()
plt.grid(True)
plt.show()
```



Logistic Regression :Accuracy

```
sgd_classifier = SGDClassifier()
sgd_classifier.fit(independent_features_train, emission_train_classes)

emission_test_classes_pred = sgd_classifier.predict(independent_features_test)

accuracy = accuracy_score(emission_test_classes, emission_test_classes_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9611913357400722
```

Confusion Matrix:

```
conf_matrix = confusion_matrix(emission_test_classes, emission_test_classes_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=encoder.classes_, yticklabels=encoder.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```