Multiple Linear Regression Assignment

Following the example during the session concerning the Simple Linear Regression model, build a Multiple Linear Regression (MLR) model to predict the CO2 Emissions, using at least 3 quantitative features as inputs. Write the equation of the regression line and evaluate that MLR model using at least 1 regression metric then, compare the result with the one obtained after the Simple Linear Regression evaluation.

Multiple Linear Regression

Step 1. Load Data and Initialize Spark Session

```python
# Create SparkSession
spark = SparkSession \
    .builder \
        .appName("MultipleLinearRegression") \
            .getOrCreate()

#Load the Data
path = "FuelConsumption.csv"
df = spark.read.csv(path, header=True, inferSchema=True)

df.show(5)
```

✓ 12.1s Python

MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	CO2EMISSIONS
2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	33	196
2014	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	29	221
2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	48	136
2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	25	255
2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	27	244

only showing top 5 rows

Consider the ENGINESIZE, CYLINDERS and FUELCONSUMPTION_COMB columns as explanatory variables

Step 2: Data Preparation

- **Select Relevant Columns:** We will filter only the ENGINESIZE, CYLINDERS and FUELCONSUMPTION_COMB columns for the regression model.
- **Handle Missing Values:** We gonna drop rows or fill missing values in predictors or input columns.
- **Assemble Features:** We'll combine predictors into a single features vector using VectorAssembler.

```python
# Select relevant features for predicting CO2EMISSIONS
#Create a subset of our dataset, selecting only the most relevant features for the analysis

df_MultipleLinearRegression = df.select("ENGINESIZE", "CYLINDERS", "FUELCONSUMPTION_COMB", "CO2EMISSIONS")

df_MultipleLinearRegression.show(5)

df_MultipleLinearRegression.describe().show()

# Handle missing values (if any)
for column in df_MultipleLinearRegression.columns:
    missing_value = df_MultipleLinearRegression.filter(F.col(column).isNull()).count()
    print(f"{column} has {missing_value} missing values")

#Assemble Features: We'll combine predictors into a single features vector using VectorAssembler.
assembler = VectorAssembler(inputCols=['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_COMB'], outputCol='features')
data = assembler.transform(df_MultipleLinearRegression)#.select('CO2EMISSIONS', 'features')
data.show()
```

✓ 2.8s

Outputs for this code are:

```
+----------+---------+--------------------+------------+
|ENGINESIZE|CYLINDERS|FUELCONSUMPTION_COMB|CO2EMISSIONS|
+----------+---------+--------------------+------------+
|       2.0|        4|                 8.5|         196|
|       2.4|        4|                 9.6|         221|
|       1.5|        4|                 5.9|         136|
|       3.5|        6|                11.1|         255|
|       3.5|        6|                10.6|         244|
+----------+---------+--------------------+------------+
only showing top 5 rows

+-------+------------------+-----------------+------------------+-----------------+
|summary|        ENGINESIZE|        CYLINDERS|FUELCONSUMPTION_COMB|     CO2EMISSIONS|
+-------+------------------+-----------------+------------------+-----------------+
|  count|              1067|             1067|              1067|             1067|
|   mean|3.3462980318650346| 5.794751640112465| 11.580880974695416|256.2286785379569|
| stddev|1.4158950514240645|1.7974472750409625|  3.48559484963484|63.37230444279997|
|    min|               1.0|                3|               4.7|              108|
|    max|               8.4|               12|              25.8|              488|
+-------+------------------+-----------------+------------------+-----------------+

ENGINESIZE has 0 missing values
CYLINDERS has 0 missing values
FUELCONSUMPTION_COMB has 0 missing values
CO2EMISSIONS has 0 missing values
+----------+---------+--------------------+------------+--------------+
|ENGINESIZE|CYLINDERS|FUELCONSUMPTION_COMB|CO2EMISSIONS|      features|
+----------+---------+--------------------+------------+--------------+
|       2.0|        4|                 8.5|         196| [2.0,4.0,8.5]|
|       2.4|        4|                 9.6|         221| [2.4,4.0,9.6]|
```

Now split the data by 80/20% for training and testing

Step 3: Train-Test Split

```
(variable) training_data: DataFrame esting sets**
training_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
```

✓ 0.0s

Step 4: Train the Multiple Linear Regression Model

```
#**Step 4:** Train the Multiple Linear Regression Model
MultipleLinearRegression = LinearRegression(featuresCol='features', labelCol='CO2EMISSIONS')


# Let's build our  Multiple Linear Regression  model by fitting our Multiple Linear Regression  estimator
mlr_model = MultipleLinearRegression.fit(training_data)
```

Use the intercept and the different coefficients to write the equation of the multiple regression line of your output feature

```
#Calculate the model coefficients and intercept
CoefficientsOfMultipleLinerRegression = mlr_model.coefficients
MLR_intercept = mlr_model.intercept

#Print the model coefficients and intercept**
CoefficientIndex = 0
for  Coefficient in CoefficientsOfMultipleLinerRegression:
    print(f"Coefficient beta{str(CoefficientIndex+1)}: {str(CoefficientsOfMultipleLinerRegression[CoefficientIndex])}")
    CoefficientIndex+=1

print('Intercept', MLR_intercept)
```

✓ 0.0s

```
Coefficient beta1: 10.39643979580147
Coefficient beta2: 7.870582501834508
Coefficient beta3: 9.457554938625346
Intercept 66.23791854165356
```

Now use MatPlotLib to express the results graphically

Step 5: Make Predictions

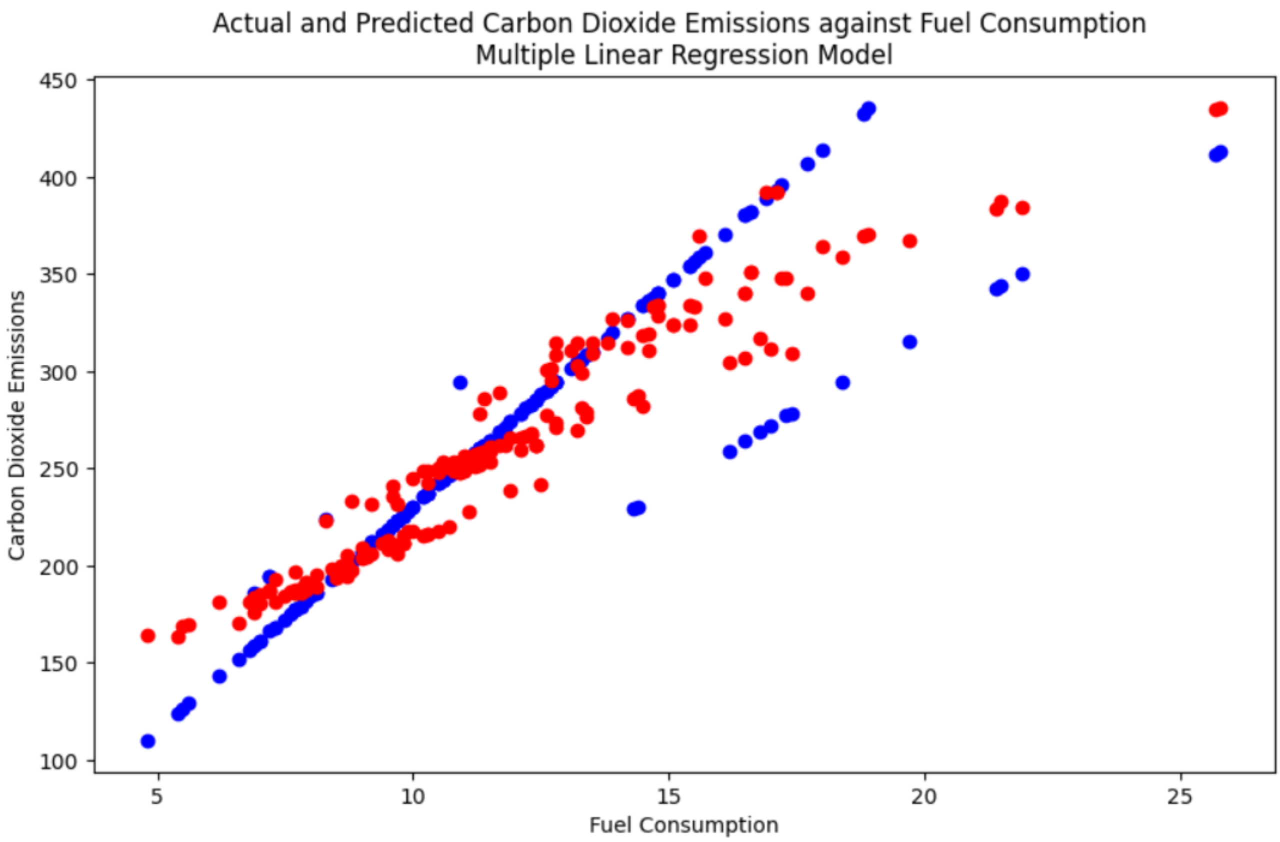
```python
#**Make predictions on the test data**
MultipleLinearRegression_prediction = mlr_model.transform(test_data)
MultipleLinearRegression_prediction.show()

MLR_pandas_df = MultipleLinearRegression_prediction.toPandas()

plt.figure(figsize=(10, 8))
plt.scatter(MLR_pandas_df['FUELCONSUMPTION_COMB'], MLR_pandas_df['prediction'])

plt.xlabel('Fuel Consumption')
plt.ylabel('Carbon Dioxide Emissions')

plt.title('Carbon Dioxide Emissions against Fuel Consumption')
plt.show()
```

The MatPlotLib code gives the following outputs and graph

ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS	features	prediction
1.0	4	6.6	152	[1.0,4.0,6.6]	170.53655093972034
1.2	4	6.9	159	[1.2,4.0,6.9]	175.45310538046826
1.4	4	5.4	124	[1.4,4.0,5.4]	163.34606093169052
1.4	4	7.3	168	[1.4,4.0,7.3]	181.31541531507867
1.4	4	7.8	179	[1.4,4.0,7.8]	186.04419278439133
1.4	4	7.9	182	[1.4,4.0,7.9]	186.98994827825388
1.4	4	8.1	186	[1.4,4.0,8.1]	188.88145926597895
1.4	4	8.7	200	[1.4,4.0,8.7]	194.55599222915416
1.5	4	7.5	172	[1.5,4.0,7.5]	184.24657028238389
1.5	4	7.7	177	[1.5,4.0,7.7]	186.13808127010896
1.5	4	7.7	177	[1.5,4.0,7.7]	186.13808127010896
1.5	4	8.5	196	[1.5,4.0,8.5]	193.70412522100924
1.6	4	7.0	161	[1.6,4.0,7.0]	180.55743679265134
1.6	4	7.6	175	[1.6,4.0,7.6]	186.23196975582658
1.6	4	7.7	177	[1.6,4.0,7.7]	187.1777252496891
1.6	4	7.8	179	[1.6,4.0,7.8]	188.12348074355162
1.6	4	8.0	184	[1.6,4.0,8.0]	190.01499173127672
1.6	4	8.8	202	[1.6,4.0,8.8]	197.58103568217697
1.6	4	9.7	223	[1.6,4.0,9.7]	206.09283512693978
1.8	4	5.5	126	[1.8,4.0,5.5]	168.45039234387363

only showing top 20 rows

As shown below, the actual Carbon Dioxide Emissions are shown in blue, the MLR model predictions are shown in blue

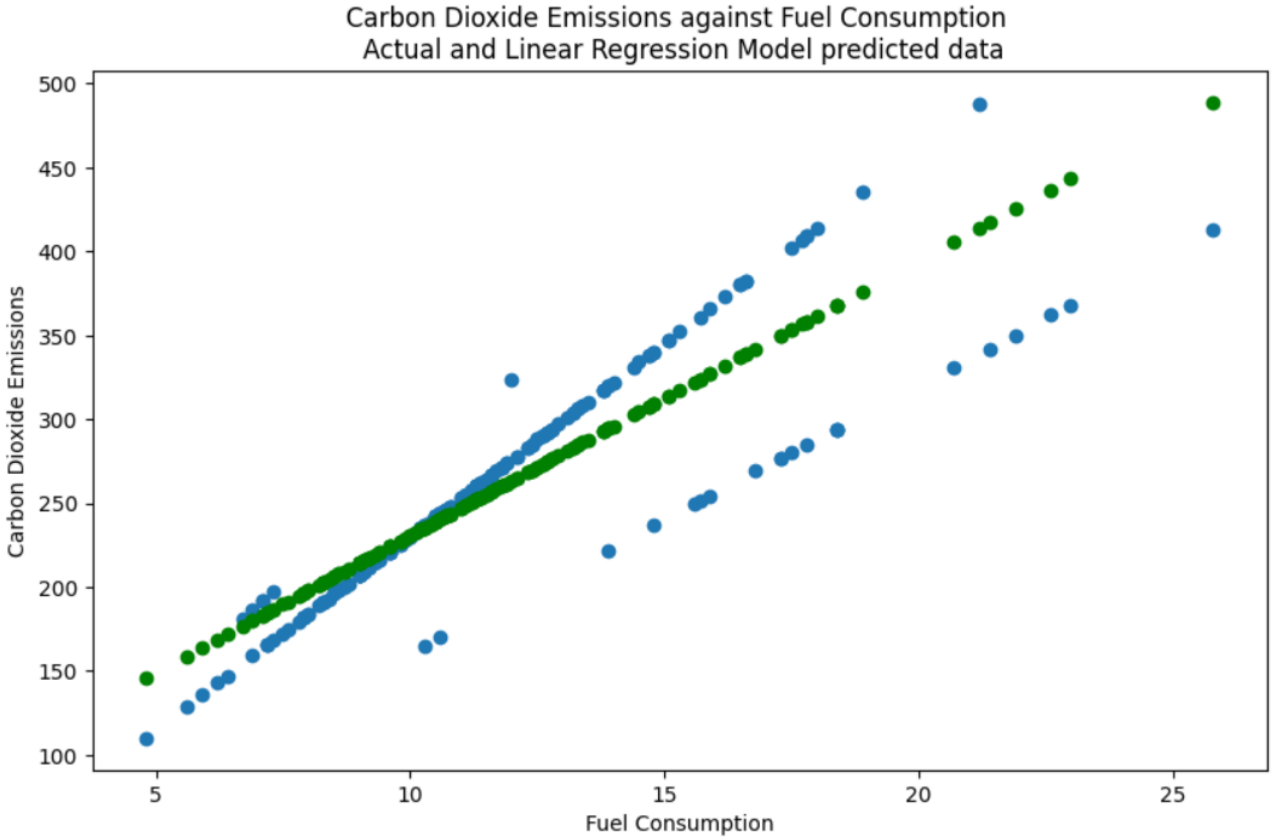


Actual and Predicted Carbon Dioxide Emissions against Fuel Consumption
Multiple Linear Regression Model
Carbon Dioxide Emissions
Fuel Consumption
450
400
350
300
250
200
150
100
5
10
15
20
25

Single Linear Regression: Carbon Dioxide Emissions against Fuel Consumption

By means of contrast with plotting the relationship between Carbon Dioxide Emissions and a three variable Multiple Linear Regression model, the relationship between Carbon Dioxide Emissions and Fuel Consumption was investigated using a Single Linear Regression (SLR) model - using the same method used in the lesson for prediction of the relationship between Carbon Dioxide Emissions and Engine Size.

As shown below, the actual Carbon Dioxide Emissions are shown in blue, the SLR model predictions are shown in green




```
# Step 8: Evaluate the Model
SLR_evaluator_rmse = RegressionEvaluator(labelCol="CO2EMISSIONS", predictionCol="prediction", metricName="rmse")
SLR_evaluator_mae = RegressionEvaluator(labelCol="CO2EMISSIONS", predictionCol="prediction", metricName="mae")
SLR_evaluator_r2 = RegressionEvaluator(labelCol="CO2EMISSIONS", predictionCol="prediction", metricName="r2")

SLR_rmse = SLR_evaluator_rmse.evaluate(FuelConsumption_SimpleLinearRegression_prediction)
SLR_mae = SLR_evaluator_mae.evaluate(FuelConsumption_SimpleLinearRegression_prediction)
SLR_r2 = SLR_evaluator_r2.evaluate(FuelConsumption_SimpleLinearRegression_prediction)

print(f"SLR Prediction Root Mean Squared Error (RMSE): {SLR_rmse}")
print(f"SLR Prediction Mean Absolute Error (MAE): {SLR_mae}")
print(f"SLR Prediction R² (R-squared): {SLR_r2}")
```
✓ 0.2s

```
SLR Prediction Root Mean Squared Error (RMSE): 32.004924934574966
SLR Prediction Mean Absolute Error (MAE): 22.620097411565805
SLR Prediction R² (R-squared): 0.7712903347752389
```

Comparison between evaluation of Multiple and Single Linear Regression model predictions

We can compare Error and R^2 values for the Multiple and Single Linear Regression model predictions against their original data, to identify that with the smallest errors

Linear Regression Model type Evaluation metric	Single	Multiple	Improvement of Model Accuracy
Root Mean Squared Error	32.0	24.0	25%
Absolute Error	22.6	17.8	21%
R^2 (R-squared)	0.77	0.87	13%

By comparing the errors, the Multiple Linear Regression model predictions are, on average, roughly 20-25% more accurate than those of the Single Linear Regression model