Assignment: 1

Linear regression

CO_2 Emission Prediction Model

Instructor: Mr. Bisrat (Msc.)

Addis Ababa Institute of Technology

School of Information Technology and Engineering

Software Stream

Name	ID Number
Lidiya Mamo	UGR/2485/14
Nathan Mesfin	UGR/0534/14

Introduction

Global warming is the largest threat to our climate. Vehicles are the greatest contributors to this threat with 71.7% of CO_2 being emitted as a result of fuel vehicles. If there is a mechanism for city administrators to identify which cars release more CO_2 to the atmosphere when driven, they can effectively ban vehicles of high emission from their streets significantly reducing the city's total emission. Thanks to linear regression, we have built such a model that predicts how much CO_2 (in g/km) a car releases based on specifications provided and throughout this report, we examine our process from data collection and analysis to training and finally testing the performance of the model. We also test the model against the 5 assumptions made for all linear regression models.

Part I

Data collection, analysis and preparation

Our Dataset was sourced from *Kaggle*. It held data on over 2000 vehicle models from Canada. Upon initial examining, it contained 7385 entries and 12 columns. There were no *NaN* values in the dataset.

Column	Non-Null Count	Data-type
Make	7385 *non-null*	object
Model	7385 *non-null*	object
Vehicle Class	7385 *non-null*	object
Engine Size(L)	7385 *non-null*	float64
Cylinders	7385 *non-null*	Int64
Transmission	7385 *non-null*	object
Fuel Type	7385 *non-null*	object
Fuel Consumption City (L/100 km)	7385 *non-null*	float64
Fuel Consumption Hwy (L/100 km)	7385 *non-null*	float64
Fuel Consumption Comb (L/100 km)	7385 *non-null*	float64
Fuel Consumption Comb (mpg)	7385 *non-null*	Int64
CO2 Emissions(g/km)	7385 *non-null*	Int64

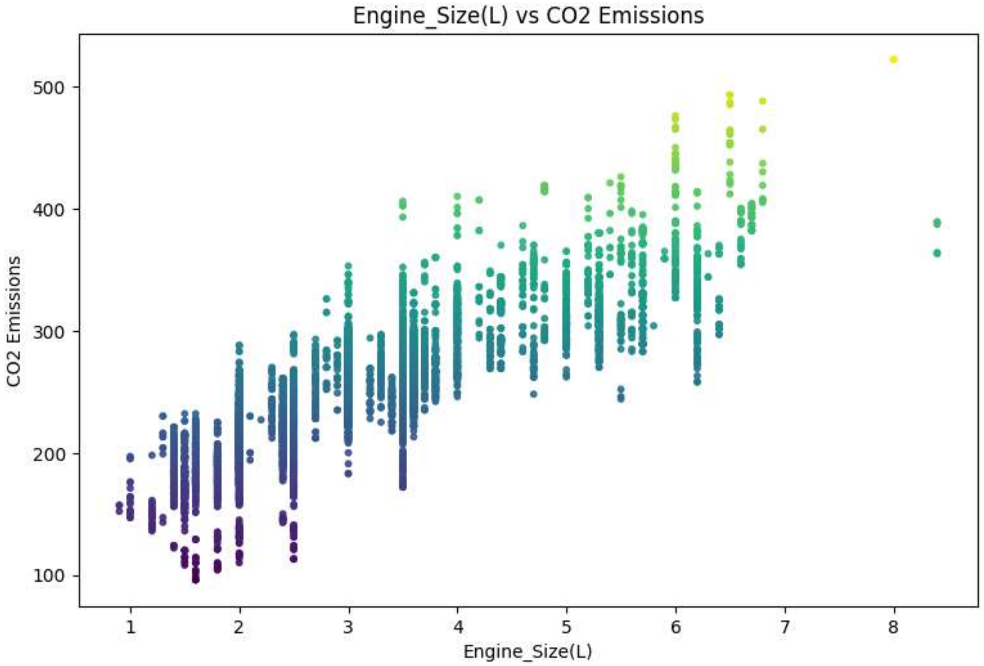
From the 12 columns and possible features for our model, we selected the following to be our independent variables:

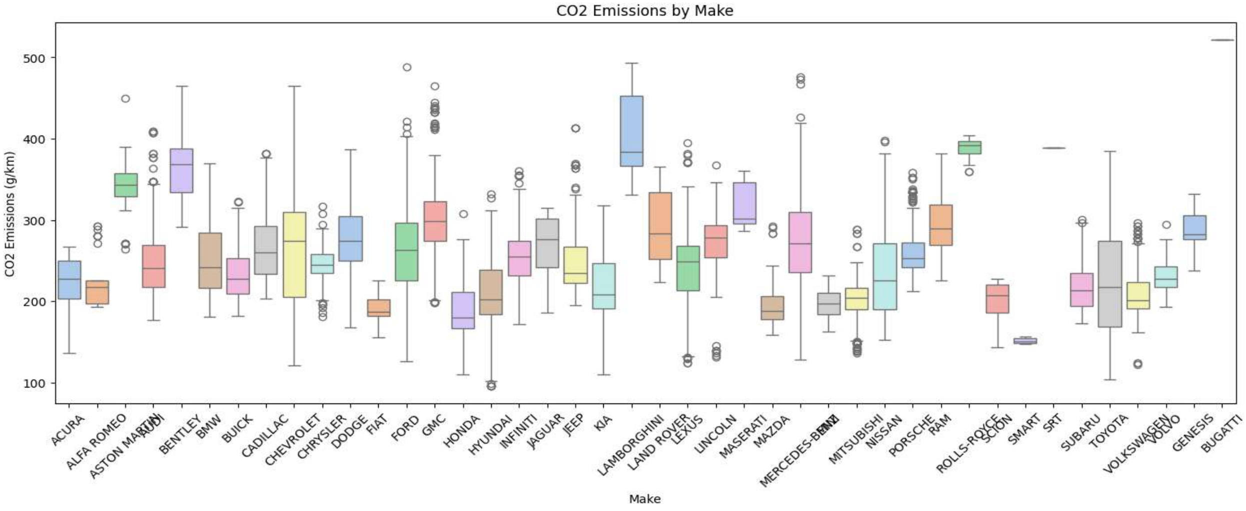
- Make
- Vehicle Class
- Engine Size(L)
- Transmission

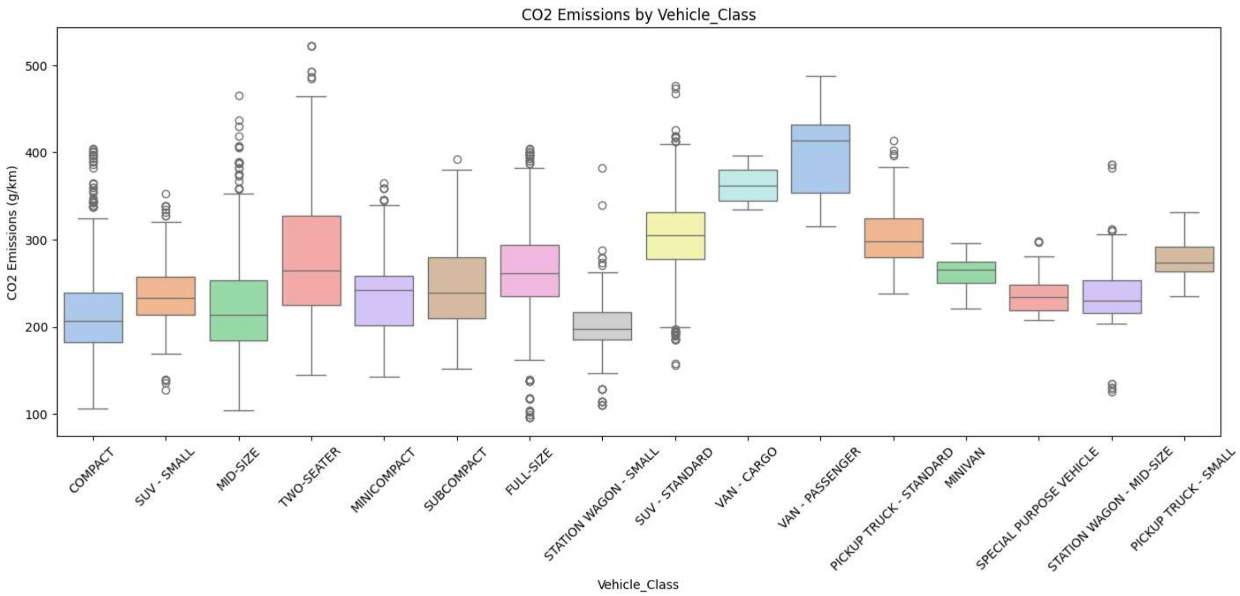
And of course, the “CO_2 Emissions(g/km)” column holds our target variable.

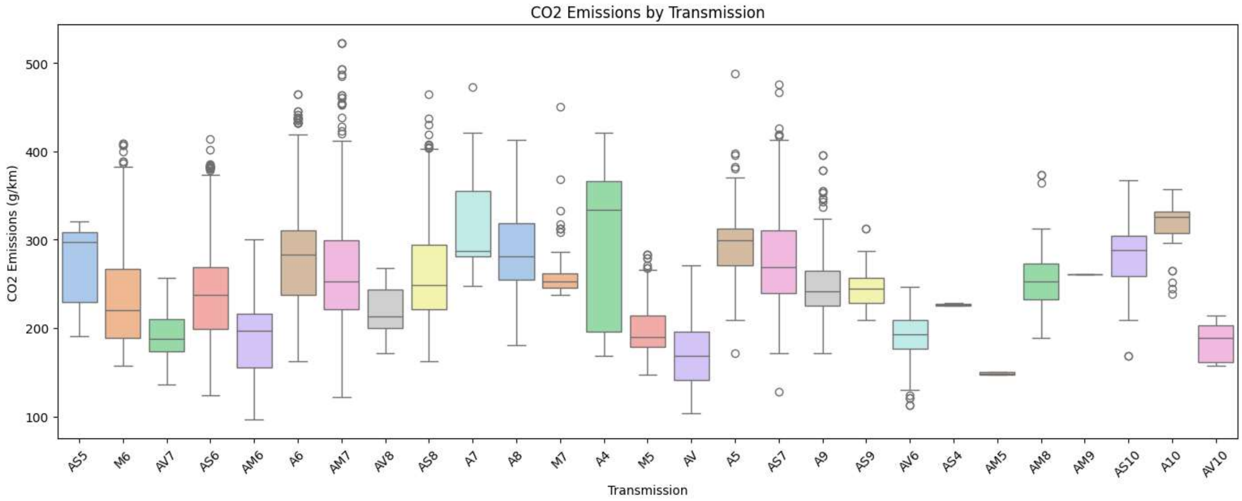
Our rationale behind choosing these columns came due to the high multicollinearity exhibited between the *Engine Size(L), Fuel Consumption City (L/100 km), Fuel Consumption Hwy (L/100 km), Fuel Consumption Comb (L/100 km), Fuel Consumption Comb (mpg)* and *Fuel Type* columns. We explore this further when we discuss the 5 assumptions in a later section. As for the model column, we found that it was a categorical field with over 2000 variations and was not found convenient to be for our purposes.

To have a better understanding of our selected features, we visualized the individual relationship between the independent variables and the target. The following are visualizations for *Make*, *Vehicle Class*, *Engine Size(L)* and *Transmission* versus the target *CO_2 Emissions(g/km)*.









Out of the 4 selected variables, 3 are categorical. If we want to make use of these variables for a linear regression model, we would have to convert them to numerical values. There are various ways in which this can be achieved. One such method being **one-hot-encoding** where we give each category within a field its own column and for every entry we set only one element to one and the rest to zero. Another method is **label encoding** where we assign each category within a field its own integer value without creating a separate column. With 46 categories in the Make field, 16 different vehicle classes and 27 fuel transmission categories we chose label encoding for our model, since one hot encoding will result in 85 columns making the data less manageable and more challenging to test against our assumptions.

Once the data was filtered and prepared in this manner, it was split into 3 sets - training, testing and validation. We used the 60-20-20 split which meant 60% will be used for training 20% will be used for testing and the remaining 20% will be used for validation.

This concludes this section; so far, we have filtered, analyzed and prepared the data. This lays out the ground work for training the model.

Part II

Training the model

Mathematical framework for Model training

As this section includes mathematical equations, please refer to the table below for definitions of variables and symbols used in this context.

Symbol	**Description**
$C(x)$	Actual CO2 emissions
$\hat{C}$ (x)	Predicted CO2 emissions
θ_0	Bias term
$\theta_1,\theta_2,\theta_3, \theta_4$	Model parameters (weights or coefficients)
X_1,X_2,X_3,X_4	Features (independent variables)
$J(\theta)$	Loss (cost) function
α	Learning rate (used in gradient descent)

The goal of this model is to predict CO_2 emissions and more specifically it is to find the weights associated with each independent variable. The mathematical representation of our model is as follows:

$$\hat{C}(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + \theta_4 X_4$$

Cost Function:

To measure how well our model's prediction $\hat{C}(x)$ matches the actual CO_2 emissions $C(x)$, we use the Mean Squared Error as the cost function. Mathematically the cost function is defined as:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{C}(x) - C(x))^{2}$$

Gradient descent:

To reduce the gap between $\hat{C}(x)$ and $C(x)$, we use gradient descent. The update is done iteratively until we arrive at optimal values for the weights and the bias term.

Here is how the weights and our bias term are updated:

$$\theta_0 = \theta_0 - \alpha \cdot \frac{\partial J}{\partial \theta}$$

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial J}{\partial \theta} \quad \text{for each j=1,2,3,4}$$

$\frac{\partial J}{\partial \theta}$ represents the partial derivative of the cost function with respect to θ. It can be calculated using:

$$\frac{\partial J}{\partial \theta} = \frac{2}{m} \sum_{i=1}^{m} (\hat{C}(x) - C(x)) \quad \text{for } \theta=\theta_0 \text{ and}$$

$$\frac{\partial J}{\partial \theta} = \frac{2}{m} \sum_{i=1}^{m} (\hat{C}(x) - C(x))\, X_{ji} \quad \text{for } \theta=\theta_j\text{, where } X_{ji} \text{ is}$$

the value of independent variable X_j for the ith training sample.

Tuning hyper parameters and setting initial values

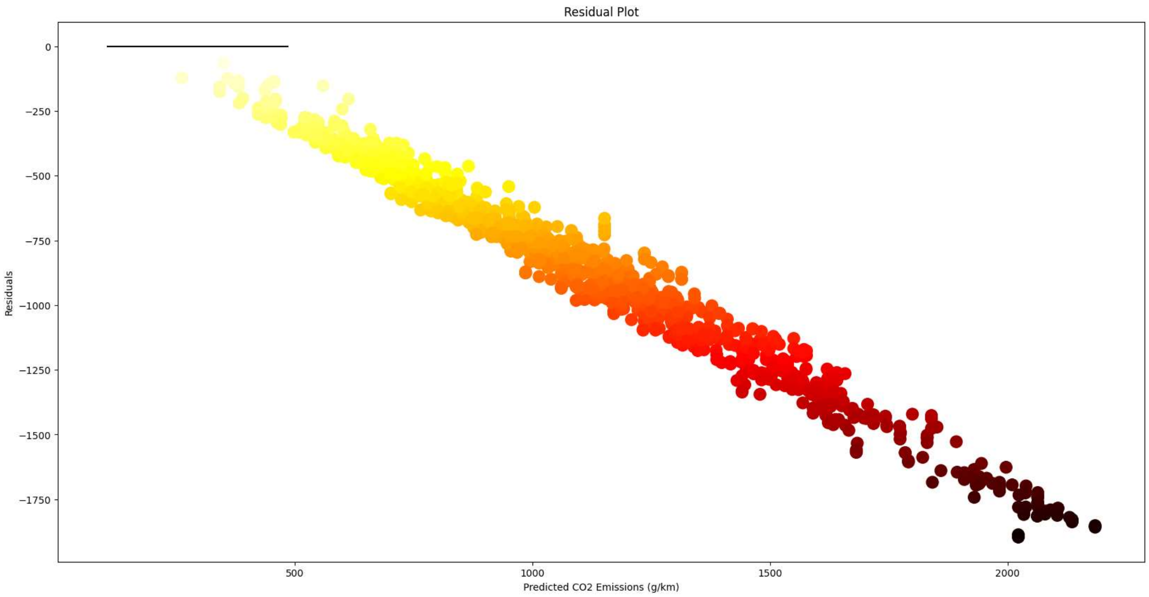
We have utilized two hyper-parameters for this implementation. The first being the learning rate alpha (α) and the other being the number of epochs. The learning rate will determine how quickly gradient descent finds us the best coefficients for the *Make, Vehicle Class, Engine Size(L)* and *Transmission* variables. And since we will be using batch gradient descent, the number of epochs will determine how many times our training set will pass through the model.

When choosing values for α and the number of epochs, we aimed to avoid these pitfalls:

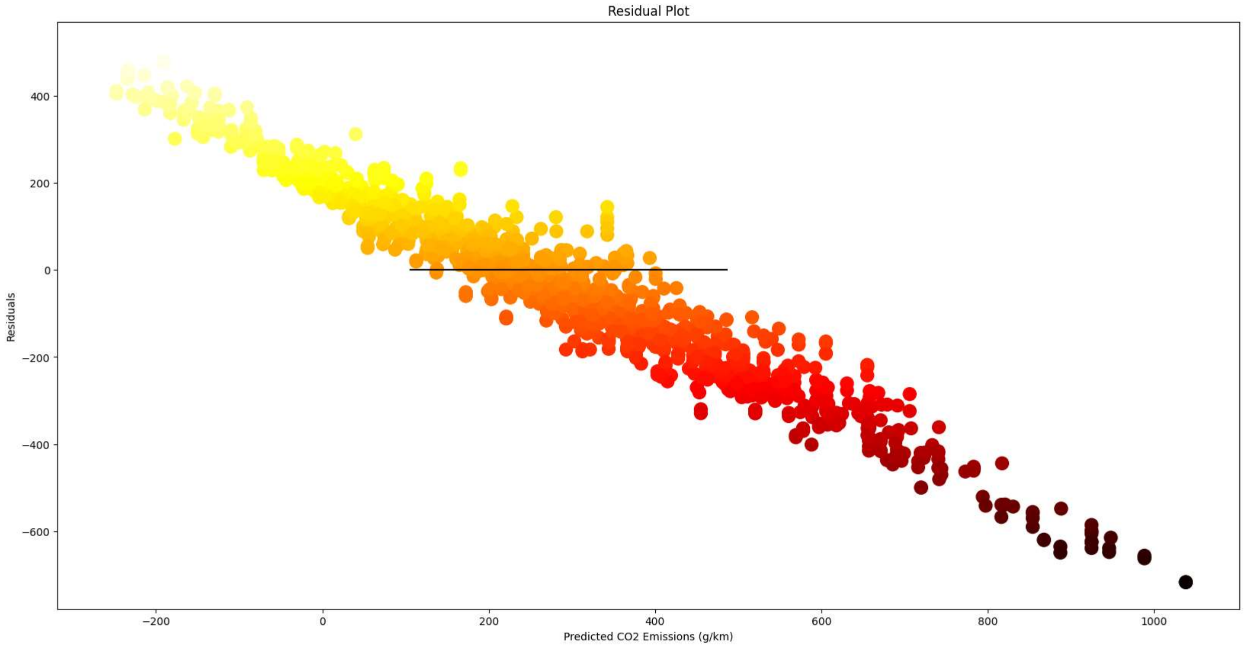
1. Too many epochs: This may result in a model that over-fits the data and poorly captures the underlying patterns. It will also take considerably increase training time without necessarily improving the model
2. Too few epochs: This may result in and under-fitting problem where we don't give the model enough opportunity to learn the relationships in the data once again resulting in poor performance.
3. Too high alpha: This results in large steps for each updates causing gradient descent to diverge instead of converge and to overshoot the parameters.
4. Too Low alpha: This will make the training process computationally expensive with every update of the parameters being unnecessarily small.

Here, we will also add that the **weights** were initialized at a random value of **60.0** and the **bias** term was initialized at **zero**. To arrive at optimal hyper-parameters for our model, we applied a trial-and-error approach. Where we would tune the hyper-parameters until a satisfactory performance is reached. Below we will show the residual plots for 5 different combinations of epochs and learning rates. We experimented with a lot more but we will limit ourselves to 5 for the sake of brevity. An ideal residual plot is one in which the residuals are evenly distributed around the mean or the horizontal line on the plot.

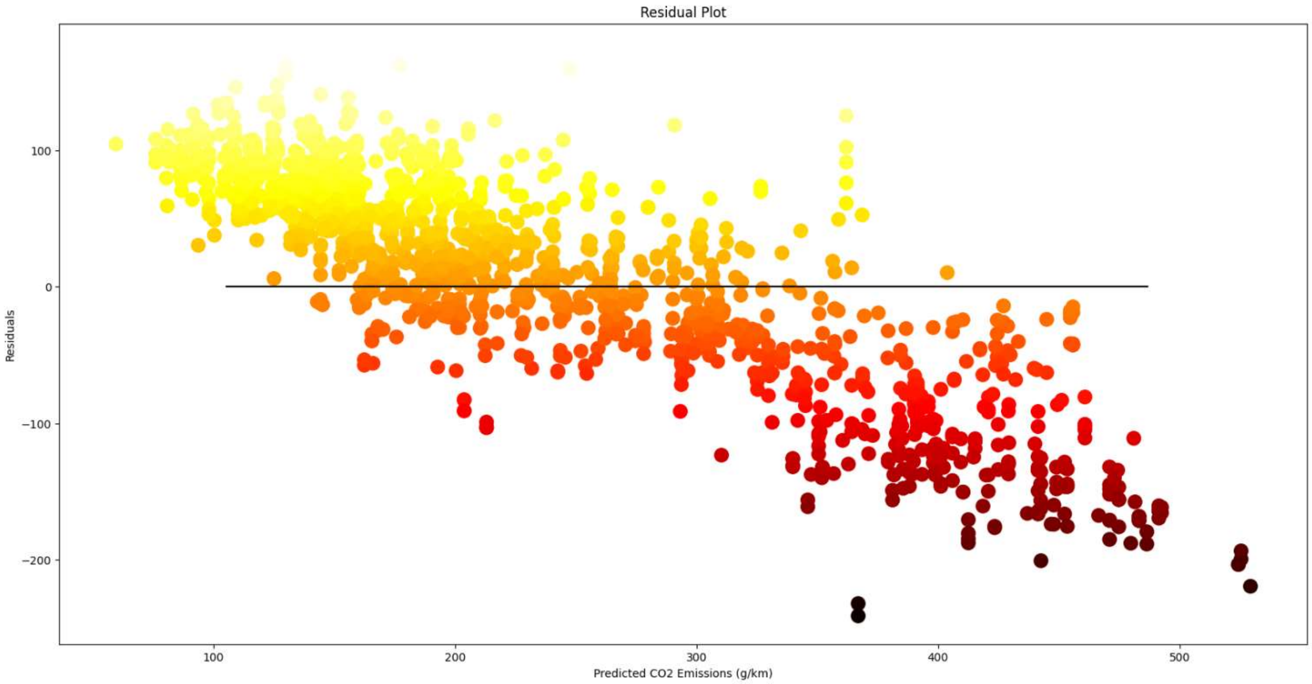
❖ **When epoch = 1 and α = 0.01:** This would be an example of a terrible model. All of the residuals are below the mean.



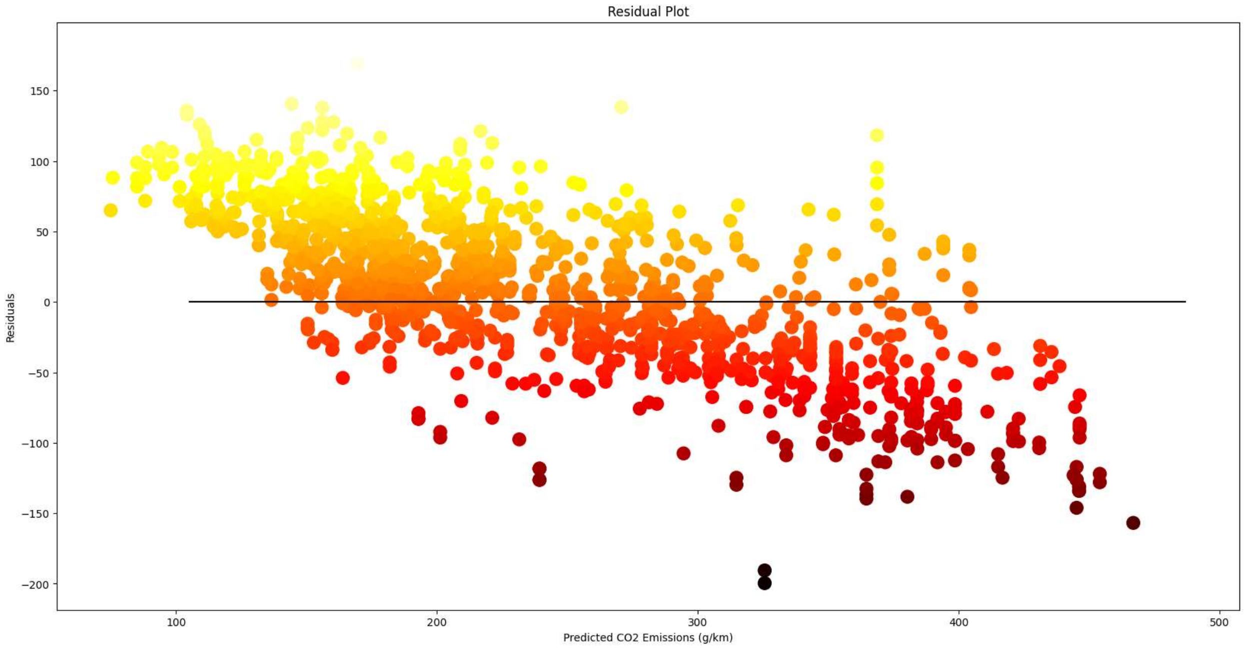
- **When epoch = 10 and α = 0.001:** This too, although it's better than the first one, is a terrible fit for our dataset.



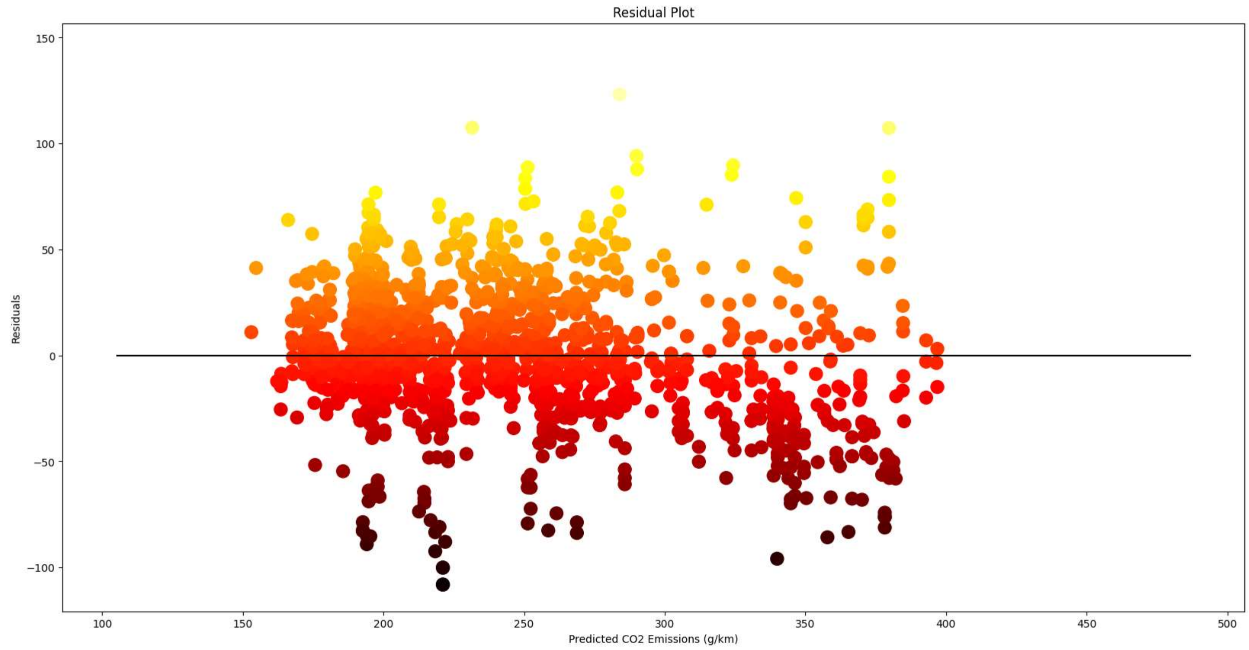
- **When epoch = 50 and α=0.01:** This shows significant progress, but it is still not a good fit.



- **When epoch = 100 and α = 0.001:**



- **When epoch = 10000 and α = 0.00001:** Although it's a steep increase from the previous combinations we examined, this gave us the best performance. Although there are some outliers, most of the residuals are more or less evenly distributed around the mean.



For the last model we have also provided a predicted versus actual value table below:

Actual CO2 Emissions (g/km)	Predicted CO2 Emissions (g/km)
269	304.960985
255	208.874317
197	187.289228
273	284.366263
179	196.319506
177	179.890805
294	282.944858
135	218.423210

Part III

Evaluating the model’s performance

In this section, we will evaluate our model using three distinct performance metrics and explain the significance of each metric's values in assessing our model's performance.

In our python implementation of the model, we have defined methods (functions) to handle these operations but here, we will present a mathematical view of each metric.

1. **Mean Absolute Error (MAE):** in the context of our model, MAE helps us determine the average deviation of our emission predictions from the true values in the testing set.
   We calculated the mean absolute error using:

$$\text{MAE}=\frac{1}{m}\sum_{i=1}^{m}|\hat{\text{C}}(\text{x})-\text{C}(\text{x})|$$

For our best model we had an MAE of approximately **23.715 g/km.** This means that on average, the predictions made by our model are either 23.715 g/km higher or lower than the true value. The following table shows the MAE for the 5 models we observed in the previous section.

Hyper parameters	MAE(g/km)
epoch = 1 and α = 0.01	**848.499**
epoch = 10 and α = 0.001	**173.412**
epoch = 50 and α=0.01	**63.614**
epoch = 100 and α = 0.001	**46.798**
epoch = 10,000 and α = 0.00001	**23.715**

2. **Root mean squared error (RMSE):** The other metric we used to test the accuracy of the predictions is the RMSE. We used this metric in addition to the mean absolute error because it emphasizes larger errors by squaring them, making substantial deviations more apparent.

We calculated the mean squared error for the CO_2 emission prediction by using:

$$\text{RMSE}=\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\hat{C}(x)-C(x)\right)\ 2}$$

Our model scored a RMSE value of **31.18g/km** indicating that on average, the model's predictions are off by about 31.18g/km units from the actual values.

Hyper parameters	RMSE (g/km)
epoch = 1 and α = 0.01	**923.799**
epoch = 10 and α = 0.001	**217.293**
epoch = 50 and α=0.01	**76.879**
epoch = 100 and α = 0.001	**57.435**
epoch = 10,000 and α = 0.00001	**31.18**

3. **Coefficient of determination (R^2):** This metric helps us determine what percentage of our independent variables explains our dependent (target) variable. In other words, how well is CO_2 emission described by our features *Make, Transmission, Engine size (L)* and *Vehicle class.* Although Coefficient of determination is not a reliable mechanism to determine a model's performance, it certainly offers valuable insight when used alongside other performance metrics. It is calculated by:

$$R^2 = 1 - \frac{SSres}{SStot}$$

SSres (residual sum of squares) is calculated by summing up the squared differences between the actual and predicted values as follows:

$$\text{SSres} = \sum_{i=1}^{m} (\hat{C}(x) - C(x))^{2}$$

SStot (Total sum of squares) is calculated by summing up the squares of the differences between actual values and their meaning in the following manner:

$$\text{SStot} = \sum_{i=1}^{m} (\bar{c}(x) - C(x))^{2}$$

where $\bar{c}$ represents the meaning of actual values

We obtained an R^2 value of approximately **0.708,** indicating that about 70% of the variability in CO_2 emissions can be explained by our selected features.

Part IV

Ensuring that the 5 assumptions are satisfied

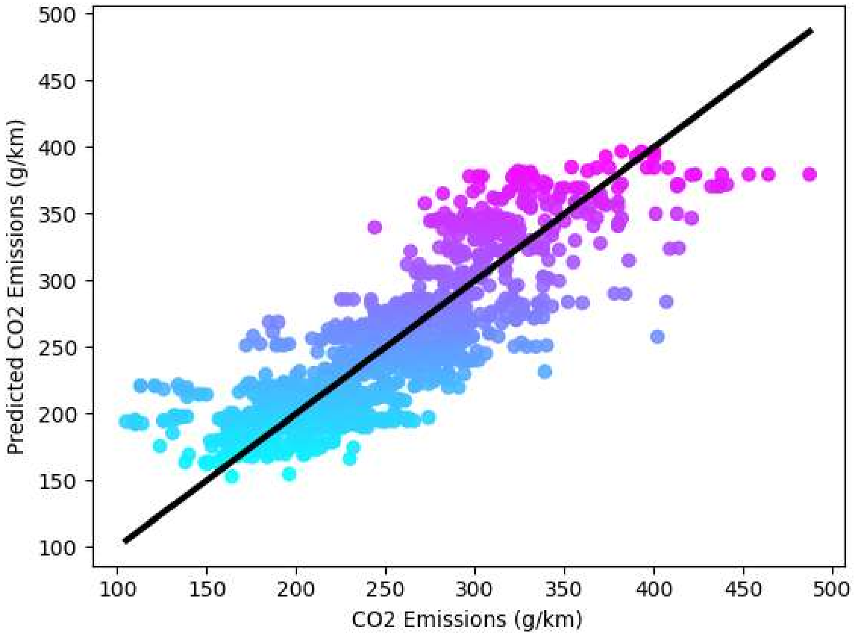
This final section evaluates the model's adherence to the five critical assumptions necessary for a reliable regression model. These are linearity, homoscedasticity, no multicollinearity, Normality of errors and no autocorrelation of errors. Beyond the residual plot, which provides significant insights into these assumptions, we will also introduce numerical statistical methods and additional graphical methods to determine whether these assumptions are met by the model.

1. Linearity

To say that our model is linear there needs to be a linear relationship between our parameters and our target. This is an assumption that we established early on when we laid the mathematical foundation for our model. To revise, we defined the model as:

$$\hat{C}(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 + \theta_4 X_4$$

None of the parameters are non-linear. Moreover, the model can be represented by a straight line as shown below:



2. Homoscedasticity

The Breusch-Pagan test is used here to detect heteroscedasticity. Heteroscedasticity is a phenomenon in which the variance of the residuals is not constant across all levels of independent variables.

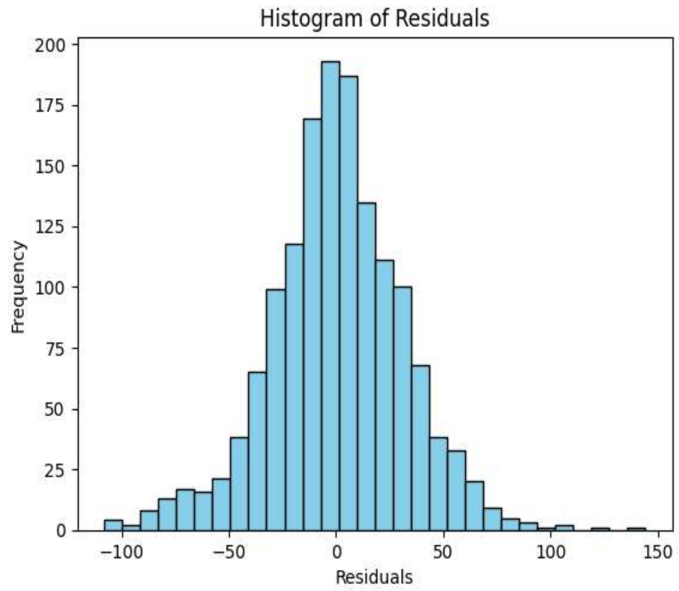
The test will provide a “p” value that we can interpret to determine whether this assumption has been met. In our implementation, we used the “stats models” library to calculate this p value. And this report we won’t delve into the complex math behind this test. While we won’t go into the intricate mathematics of the test, we will focus on how to interpret the value and present the results we obtained for our model.

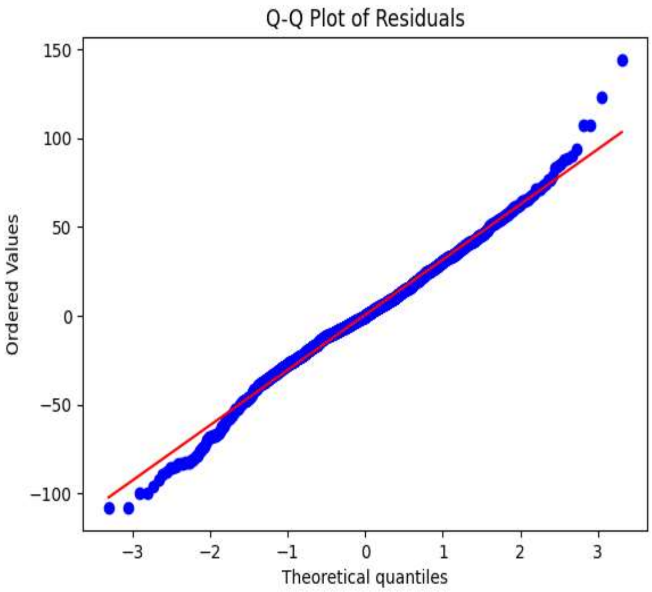
In the Breusch-Pagan test, a *p* value of above 0.05 indicates that homoscedasticity has been fulfilled and a *p* value of 0.05 or below indicates that there is a heteroscedastic distribution or that the errors do not have a constant variance.

Our model scored a Breusch-Pagan p-value of **2.742366481029623 x 10** $^{-33}$ indicating significant heteroscedasticity. Observing the plot presented in the linearity section, we can say that for large and small values the model will perform quite poorly resulting in sub-optimal accuracy.

3. Normality of Errors

A histogram and Q-Q plot are used to assess the normality of the errors. The histogram indicates that the residuals follow an approximately Gaussian distribution. And on the other side, the residuals align closely along a straight line in the Q-Q plot, which suggests that they are normally distributed.





4. No Multicollinearity

As stated earlier in our report, we eliminated certain features due to multicollinearity. To check for multicollinearity, we calculated the variance inflation factor (VIF) for each feature. The aim is to have a variance inflation factor of less that 5.This test helped us determine which ones were multicollinear. Features such as *Engine Size(L), Fuel Consumption City (L/100 km), Fuel Consumption Hwy (L/100 km), Fuel Consumption Comb (L/100 km), Fuel Consumption Comb (mpg), and Fuel Type* were removed because of high VIF values - some even reaching 2000.

After removing most of these columns, we still observed significant multicollinearity between *Engine Size and Fuel Consumption Comb (mpg),* which prompted us to remove Fuel Consumption Comb (mpg) from the features.

For the remaining 4 features the multicollinearity scores, as indicated by VIF, are as follows:

Feature	**VIF**
Make	2.862573
Vehicle Class	3.263873
Engine Size (L)	4.870681
Transmission:	2.992383

All these values are below 5, indicating that multicollinearity is no longer a concern.

5. No Autocorrelation of Errors

Although this assumption is primarily suited for time series data, we tested our model against it. In order to assess the presence of positive and negative autocorrelation among the model's residuals, we used the Durbin-Watson test. The values from the Durbin-Watson statistics range from 0 to 4, where a value around 2 (typically between 1.7 and 2.3) indicates no autocorrelation.

The Durbin-Watson statistic test yielded a value of approximately 1.98, which is around 2. This result suggests that our model's residuals show no significant autocorrelation, or that the errors are randomly distributed. This outcome supports the validity of our model's assumptions and suggests that the regression predictions are reliable in terms of residual independence.

References

- Dataset: https://www.kaggle.com/datasets/debajyotipodder/co2-emission-by-vehicles/data
- Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. New York: Springer
- Murphy, K. P. (2012). Machine learning: a probabilistic perspective