In this project, you will be implement multiple linear regression and apply it to a dataset. This assignment follows the specification in the “Project Guidelines” document on Canvas.

I. General Information

Your project is responsible for implementing multiple linear regression. Your project uses a dataset that has one target variable and multiple input features. Your report should cover the relevant steps in the “Project Workflow”, reporting what you did and the results. You are responsible for implementing your own regression functions, with the exception of basic numerical libraries.

II. Dataset

This project will use the Auto-Mpg dataset from the UCI Machine Learning repository. The dataset
file is named auto-mpg.csv. The data includes the
following features: MPG (Continuous), Cylinders (Multivariate Categorical), Displacement (Continuous),
Horsepower (Continuous), Weight (Continuous), Acceleration (Continuous), Model Year (Multivariate Categorical), Origin (Multivariate Categorical), and Car Name (String).

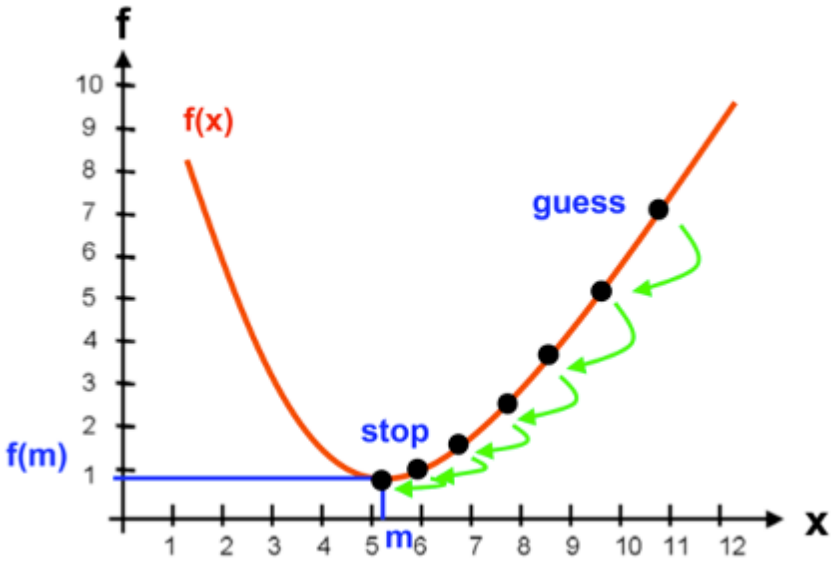

Figure 1: A visualization for the Gradient Descent approach.

III. Implementation Requirements

You should implement multiple linear regression with gradient descent. The Gradient Descent algorithm is an iterative approach to calculating coefficient “guesses” for a best-fit regression line.

Your implementation cannot make use of machine learning libraries that perform regression for you (e.g. sklearn). You can, however, use them for supplemental activities (e.g. normalization).

IV. Implementation Recommendations

The Python library NumPy may be of substantial use. NumPy allows you to store data in matrices
(see the np.array() function) and apply matrix operations to them. Consider a similar scenario in which we seek to implement linear regression for predicting performance.
Should you choose to make use of NumPy, you should be aware of the dot function that facilitates dot product calculations and the np.sum function, which facilitates summation. An example can be found here: https://pythonexamples.org/python-numpy-dot-product/.

IV. Evaluation Requirements

Using your implementation, you should evaluate two regression models:

- A regression model that uses the provided dataset as is.
- A regression model that uses a "standardized" version of the dataset.

"Standardization" is the process of (1) normalizing continuous features and (2) standardizing these features' mean to zero and variance to 1. As implementing standardization is not the goal of this project, I am permitting you to use the MinMaxScaler and StandardScaler classes in the sklearn.preprocessing library to perform standardization.

The R_2 Metric

The most common way to evaluate regression models is through the R_2 metric, which measures how much of your data's variance is captured by your regression model. Generally speaking, your model's R_2 value should be higher because you want your model to explain as much of your data's variance as possible (i.e., has the best "fit"). In other words, the best-performing model is the one that yields the highest R_2. Alongside the R_2 metric, you should report the final coefficients in your models. To better understand your coefficients, it may also help to present information about how "cost" (i.e., $Y_{pred} - Y$) changes across iterations for each model. (e.g., At what point do your iterations start having little change in your cost?).