Multi Linear Regression Assignment

DasRay, A (s423160) and Puddicombe, J(s439038)

October 2023

Contents

1 Introduction

1.1 Aim

The aim of the work described in this report is to explore the method of gradient descent for the solution of the multi-linear regression problem with respect to optimality and appropriateness for certain data sets.

1.2 Objectives

In order to achieve the aim, we will be carrying out the following tasks:

- to preprocess the data from two sets, ‘advertising’ and ‘auto’, to cleanse them and appropriately transform the data, rendering them optimal for the required investigations.
- to construct an algorithm for gradient descent in Python that takes in values η, n and a data set, and iterates gradient descent n times with learning rate η.
- to investigate the effectiveness and efficiency of varying values of η and n with respect to the data sets, using training and test data, r-squared test and validation using scikit-learn’s implementation of the analytic solution.
- to explore specific modifications of the linear regression equations by implementation of the following:
  - reduction of the dimensionality of the advertising data set
  - introduction of a cross-multiplicative term in the advertising data set
  - exploration of the optimal degree of the regression polynomial with respect to the auto data set
- to validate our model accuracy using functions from standard Scikit-Learn library.

2 Algorithm Development and Validation

2.1 Methodology

2.1.1 Theory

Linear regression is a statistical method that is used to describe the relationship between a dependent variable and one or more independent variable by fitting a linear equation to the data set. When the dependent variable can be expressed as a function of more than one independent variables we call it multi-linear function, and can be expressed as below:

$$y(x_1, x_2, \dots, x_r) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_r x_r \tag{1}$$

where, [7]

- $x_1, x_2, \dots x_r$ are the independent variables or features.
- $y(x_1, x_2, \dots, x_r)$ is the dependent variable or output
- $\theta_0, \theta_1, \theta_2, \dots \theta_r$ are the parameters of the function.

To estimate the approximate values of the parameters such that the function fit the data we employ an iterative optimization algorithm called Gradient Descent.[**?**]

In Gradient Descent we first determine a cost function which is nothing but the square of the difference between predicted values and actual values of y. We then try to minimize the this cost function via a series of iterations.[6] The cost function is given by:

$$J(\theta_0, \dots, \theta_r) = \frac{1}{n}\sum_{i=1}^{n}\left(y(x_{1i}, \dots, x_{ri}) - y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{r}\theta_j x_{ji} + \theta_0 - y_i\right)^2 \tag{2}$$

where [7]

n is the number of observations

x_{ji} are the individual observations of each of the features.

In each iteration we calculate $\nabla J(\theta_0, \dots, \theta_r)$,

$$\nabla J(\theta_0, \dots, \theta_r) = \frac{\partial J(\theta_0, \dots, \theta_r)}{\partial \theta_0}, \dots, \frac{\partial J(\theta_0, \dots, \theta_r)}{\partial \theta_r} \tag{3}$$

where [7]

$$\frac{\partial J(\theta_0, \dots, \theta_r)}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}\left(y(x_{1i}, \dots, x_{ri}) - y_i\right)$$

$$\frac{\partial J(\theta_0, \dots, \theta_r)}{\partial \theta_1} = \frac{2}{n}\sum_{i=1}^{n}\left(y(x_{1i}, \dots, x_{ri}) - y_i\right)x_{1i}$$

$$\frac{\partial J(\theta_0,\ldots,\theta_r)}{\partial\theta_r} = \frac{2}{n}\sum_{i=1}^{n}\left(y(x_{1i},\ldots,x_{ri}) - y_i\right)x_{ri}$$

And then update each parameter as,[7]

$$\theta_j new = \theta_j current - \eta.\frac{\partial J(\theta_0,\ldots,\theta_r)}{\partial\theta_j}$$

Here η is called the learning rate. We iterate for a number of steps and a fixed learning rate such that θ values are optimised in order to minimise the cost function.

In order to establish whether we have achieved a sufficiently accurate model, we calculate the "goodness of fit" of each of the training and test data against the regression function. This value is given by:

$$1 - \frac{\sum_{i=1}^{n}(y_i - y(x_{1i},\ldots,x_{ri}))^2}{n\cdot var(y)} \tag{4}$$

We will consider any value above 0.7 to be acceptable, since such a value would be seen as strong within the fields of finance as well as social science.[1]

2.1.2 Code Architecture

Model Function

In implementing the above calculations, to ensure that certain parts of our code is reusable and for code clarity, we decided to create some nested functions. (This also expressed the SOLID principle of each function doing a single thing.) We started off by initializing a function called **ComputationV2** which takes in a dataframe of our features and target, η, and number of steps and returns θ_j values. It evaluates the equation:

$$\theta_j new = \theta_j current - \eta.\frac{\partial J(\theta_0,\ldots,\theta_r)}{\partial\theta_j}$$

ComputationV2 while evaluating $\frac{\partial J(\theta_0,\ldots,\theta_r)}{\partial\theta_j}$ calls the function **calcDiff** . **calcDiff** evaluates

$$\frac{\partial J(\theta_0,\ldots,\theta_r)}{\partial\theta_r} = \frac{2}{n}\sum_{i=1}^{n}\left(y(x_{1i},\ldots,x_{ri}) - y_i\right)x_{ri}$$

and returns it. To evaluate $y(x_{1i},\ldots,x_{ri})$ **calcDiff** calls another function **estimatedTargetValue** which evaluates the expression and returns it.

For all our iterations we have preferred to use a while loop over a for loop because the while loop runs as long a condition is true, whereas a for loop iterates through an iterator. To use the for loop we would first have to create a list or use a range function, which we realised would be inefficient.

Goodness of Fit Function

To evaluate the Goodness of Fit of one set of θ we have created the **goodnessOfFit** function. This function takes in the array consisting of the features and target and the θ values and returns the Goodness of Fit value as defined above.

(To do this, it reuses the **estimatedTargetValue** function referred to above.)

Function to check Goodness of Fit for particular learning rate and no of steps

We also developed a function called **computationV2Multipoint**, which evaluates the Goodness of Fit for a given learning rate and number of steps. This particular function takes in a data frame of features and target, η, step gap and the number of times the step gap is iterated, and returns another data frame with the θ values and the Goodness of Fit for all the different step intervals. Originally we used computationV2 along with some "shell" code for this purpose, but found that with the large step numbers we were calculating sequentially, it was vital to develop this more efficient algorithm, which operates in O($number\_of\_steps$) rather than O($number\_of\_steps^2$). This helps us evaluate at which number of steps the set of θ values gives the best Goodness of fit. This function calls **estimatedTargetValue** and **goodnessOfFit** to perform as intended.

For specific scenarios we have slight variations in the Function name and body.

2.1.3 Preparing the data sets

In order to validate the Gradient Descent algorithm, we used two data sets, "adv" and "auto". Before using the data sets, we carried out exploratory data analysis on these sets in order to assess their suitability for linear regression and to cleanse them appropriately. Our planned approach is described in the next section.

To explore and cleanse the data, we employed the following methodology, which was broadly as outlined by O'Neill and Schutt [2, pp.34-38], who were basing their approach on the original approach of John Tukey of Bell Labs.

As per the specific assignment questions, we categorised sales (in the advertising data set) and mpg (in the auto data set) as the target (dependent) variables, and all the other variables as feature(dependent).

Impermissible and missing data values

The Advertising Dataset does not have missing Datapoints however the Auto Dataset has them. we have discussed in details as how we chose to deal with the missing data in section 4.1

Transformation/Normalisation of variables We decided on principle not to normalise the features. This was because there would be no negative impact on the results: the derived theta values would accurately represent the linear dependencies, even with unnormalised data. The features are not “competing” against each other in the way that they would be in, say, principal component analysis. Furthermore, no value transformation would be required when using the derived linear model for prediction.

However, we did plan to transform the variables in each of the data sets in order to explore non-linear regression.

Outliers We decided not to remove outliers as these seemed to be valid data points, and some of the outliers with respect to one feature might not be outliers with respect to another.

Plots We planned to generate summary statistics(boxplots) of all the variables, to get a general feel for the distributions, and, in particular, to examine outliers.

We planned to generate pairwise scatter plots of the dependent variables against the corresponding independent variables, to get a feel for the patterns of dependency of the target on the features, before performing the more scientific analysis.

We planned to generate Seaborn correlation heatmaps to determine if there were any features that were highly correlated within a data set, and which therefore could cause issues [3, p.127]. This is due to linear independence of the features being a premise of the analytical solution [4, p.74].

3 Analysis of the Advertising Dataset

3.1 Exploratory Data Analysis and Cleansing: Advertising Data Set

There were found to be no missing values in the advertising data set.

In Figure 1, we see a display of boxplots of the four variables in the data set. There are merely two of outliers in the newspaper plot, and these are not extreme, so they should cause no issues for the proposed model. The TV and radio distributions appear to be almost perfectly symmetrical. The newspaper and sales distributions are skewed towards the low values. This may be because they are samples from a Poisson distribution. A Poisson distribution is produced as the sum of a high number of identical independent Bernoulli variables, each of which has a small chance of a "1" value. This would be expected of the sales variable, since the target population is likely to consist of a large number of people, each of whom is unlikely to purchase; it could also be hypothesised that newspaper advertising arises from a large number of ad-hoc decisions that are rarely affirmative. In contrast, in TV and radio advertising, there may be a fewer number of decisions to be made in favour of an advertising campaign, but a larger probability of an affirmative. This would give rise to binomial distributions, consistent with the symmetricality we see here.

The exact type of distribution each variable displays (specifically the dependent (sales) variable) and whether it is skewed or not, will not per se affect the suitability of modelling by linear regression, if we assume that our dataset is sufficiently large.[5]

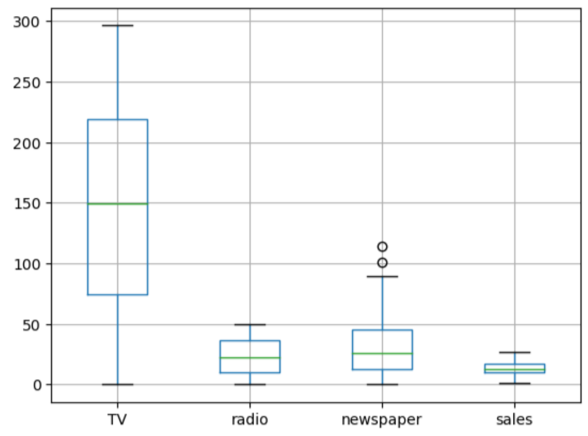


Figure 1: Boxplots of TV, Radio and Newspaper Adspend and Sales

Figures 2, 3 and 4 show pairwise scatterplots of sales against the feature variables.

In Figure 2 we see an interesting horn shape. It seems that sales could be accurately modelled as a linear function of TV advertising. However the gradient

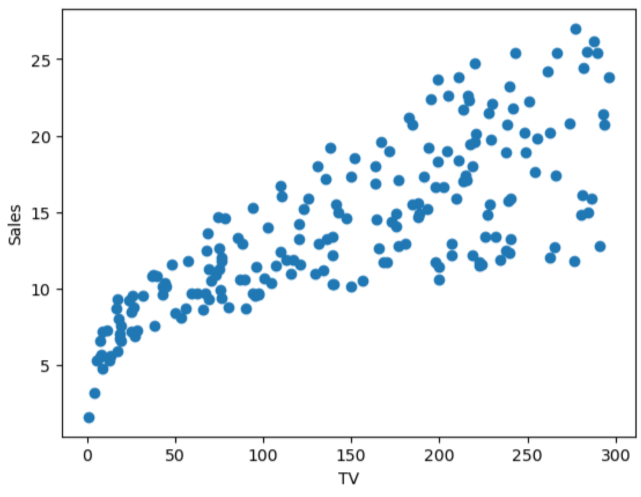


Figure 2: Sales vs. TV

would suddenly decrease at the ”20” value of TV. So it seems that the first few ads or ad campaigns would have a relatively large effect on sales. Alternatively, it could be that the points with associated low values for TV advertising are outliers, or that low ”TV” values combine with a specific pattern of other feature values to give rise to a higher gradient. In any case, it is a small deviation from from the shape of the bulk of the distribution, so should not impact the modelling significantly. It does, though, highlight the need for caution when extrapolating any predictive line of best fit. As it increases in one medium, one should expect advertising to reach a ”saturation” level where further increases do not increase sales.

The vertical distribution pattern of this graph is very even, with linear boundaries that are non parallel. This would be consistent with the existence of an evenly distributed ”amplifying” feature either within or outside the given feature set that is statistically independent of TV advertising. We later examine whether this feature might be radio advertising.

Figure 3 shows a similar horn shape for sales against radio advertising, though there is no change of gradient, and the distribution seems to cluster at the upper edge of the horn shape. If there is an amplifying feature in play here, it may not have an evenly distributed probability distribution or it may be statistically dependent on the radio advertising spend.

Figure 4 shows that newspaper advertising has minimal effect on sales. Again we see the skewing of the distribution consistent with a Poisson pattern.

Figure 5 shows the cross-correlation heatmap for the feature and target variables. Points to note here are:

- TV advertising is pairwise independent of radio and newspaper advertising.

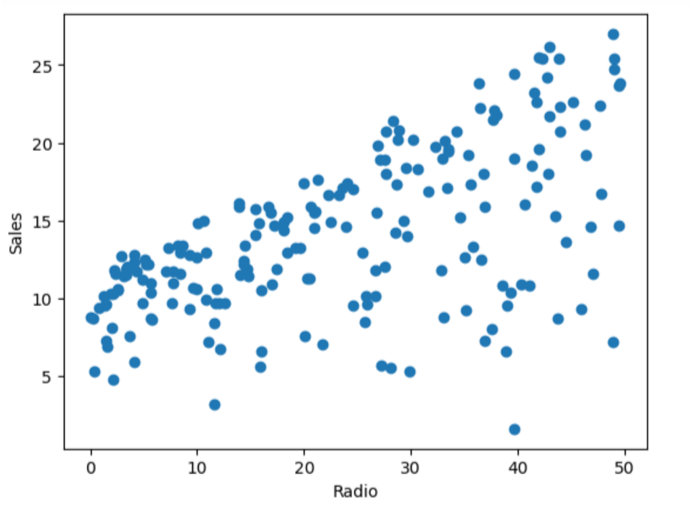


Figure 3: Sales vs. Radio

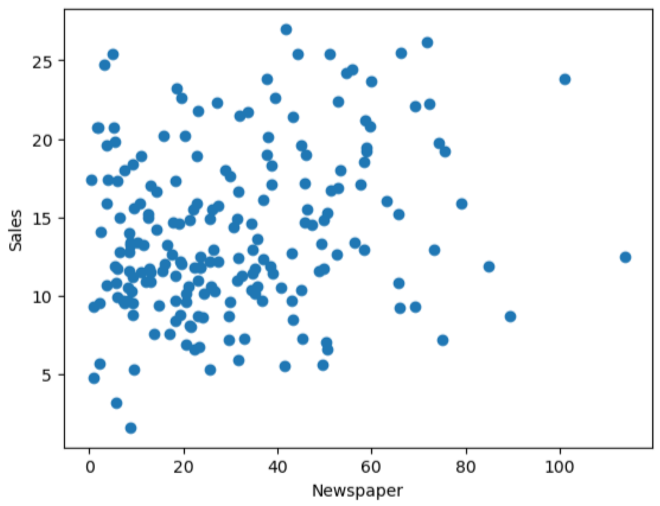


Figure 4: Sales vs. Newspaper

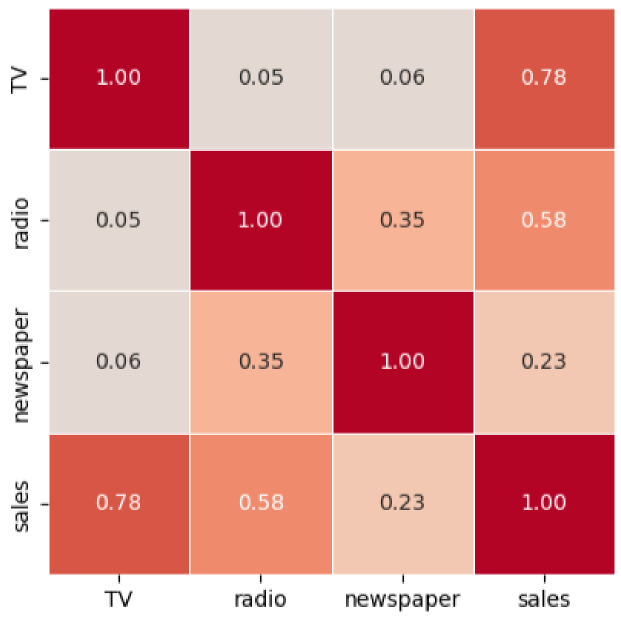


Figure 5: Advertising Variables: Cross-correlation Heatmap

- There is a small but definite positive correlation between radio and newspaper advertising.
- All three features display positive correlation with sales: TV(high), radio(moderate) and newspaper(low).

Having performed this analysis, we have found no adverse indicators for implementing a linear regression model.

3.2 Scenarios

We investigated the correlation between sales and the features of the advertising dataset by modelling the relationship using three different regressors.

The first regressor we chose was:

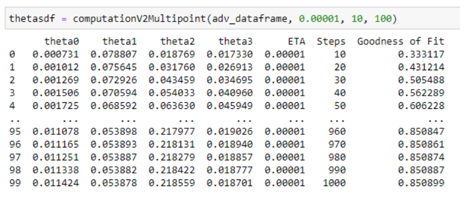
$$sales = \theta_0 + \theta_1 * TV + \theta_2 * radio \tag{5}$$

The second was:

$$sales = \theta_0 + \theta_1 * TV + \theta_2 * radio + \theta_3 * newspaper \tag{6}$$

And the third was:

$$sales = \theta_0 + \theta_1 * TV + \theta_2 * radio + \theta_3 * TV * radio \tag{7}$$



```
thetasdf = computationV2Multipoint(adv_dataframe, 0.00001, 10, 100)
```

```
      theta0    theta1    theta2    theta3      ETA  Steps  Goodness of Fit
0   0.000731  0.078807  0.018769  0.017330  0.00001     10         0.333117
1   0.001012  0.075645  0.031760  0.026913  0.00001     20         0.431214
2   0.001269  0.072926  0.043459  0.034695  0.00001     30         0.505488
3   0.001506  0.070594  0.054033  0.040960  0.00001     40         0.562289
4   0.001725  0.068592  0.063630  0.045949  0.00001     50         0.606228
..       ...       ...       ...       ...      ...    ...              ...
95  0.011078  0.053898  0.217977  0.019026  0.00001    960         0.850847
96  0.011165  0.053893  0.218131  0.018940  0.00001    970         0.850861
97  0.011251  0.053887  0.218279  0.018857  0.00001    980         0.850874
98  0.011338  0.053882  0.218422  0.018777  0.00001    990         0.850887
99  0.011424  0.053878  0.218559  0.018701  0.00001   1000         0.850899
```

Figure 6: Table of Trace of Gradient Descent Procedure (Advertising Data Set)

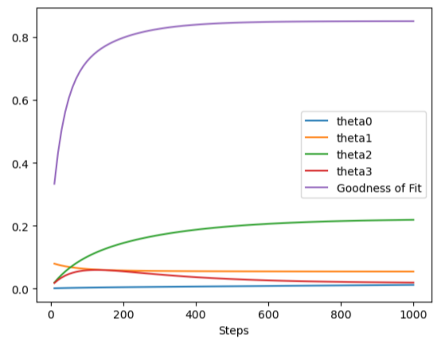


Figure 7: Graph of Trace of Gradient Descent Procedure (Advertising Data Set)

Comparison between the first two regressors would test the hypothesis that newspaper advertising had minimal predictive value for sales. The third regressor would test the hypothesis that there was non-linearity caused by interaction between TV and radio advertising.

We tested the gradient descent algorithm (which had already been validated against sklearn for the auto dataset) for Regressor 2 for the values $\eta = 0.00001$ and steps = 1000, with the results monitored at each 10 steps.

We obtained a trace of the algorithm (Figure 6 and Figure 7).

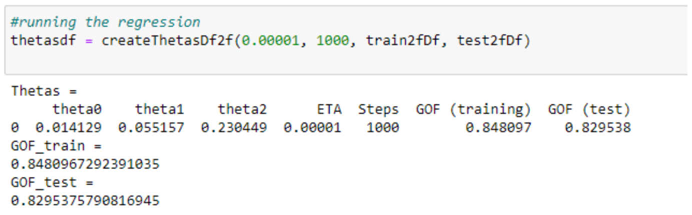
The sequence of differences between the values for the goodness of fit appear to be monotonically decreasing. This does not prove convergence, though it is a necessary condition, but convergence seems to be indicated by the graph (the purple line). Moreover, each of the thetas seems to converge.

For each of the three regression functions, we decided to use an 80%/20% train/test data split to validate the model, following the example of O'Neill and Schutt [2, p.77]. Furthermore, we made the assumption that η=0.00001 and steps = 1000 would be valid for the calculation of all three regressors.

3.3 Results, Analysis and Discussion

3.3.1 Regressor 1

Figure 8 shows the result obtained for Regressor 1. The gradient descent process produced a goodness-of-fit value for the training set of 0.85 and for the test set of 0.83, showing that the linear model with TV and radio as the only features is a good predictor for the theta values we calculated. This can be seen illustrated in Figure 9 and Figure 10, where the regressor plane (in purple) closely shadows both the training and test sets.



```
#running the regression
thetasdf = createThetasDf2f(0.00001, 1000, train2fDf, test2fDf)
```

```
Thetas =
     theta0    theta1    theta2      ETA  Steps  GOF (training)  GOF (test)
0  0.014129  0.055157  0.230449  0.00001   1000        0.848097    0.829538
GOF_train =
0.8480967292391035
GOF_test =
0.8295375790816945
```

Figure 8: Sales modelled by a linear function of TV and radio: regressor parameters table

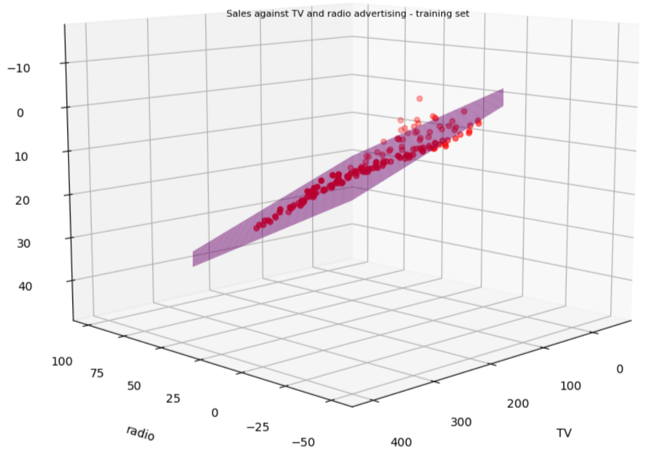


Figure 9: Sales modelled by a linear function of TV and radio: regressor and training set graph

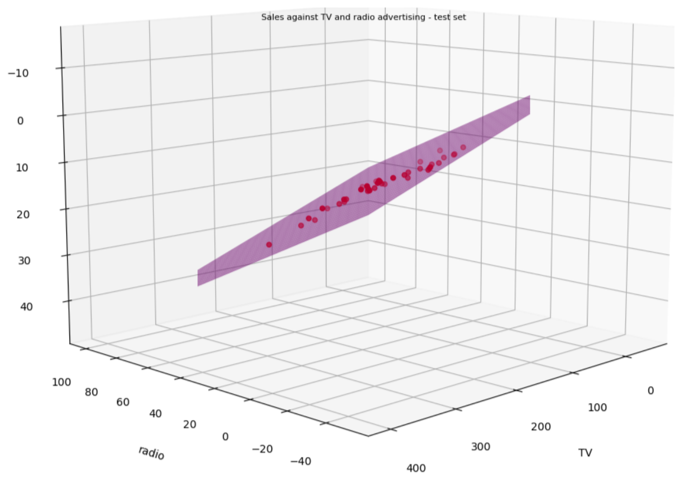
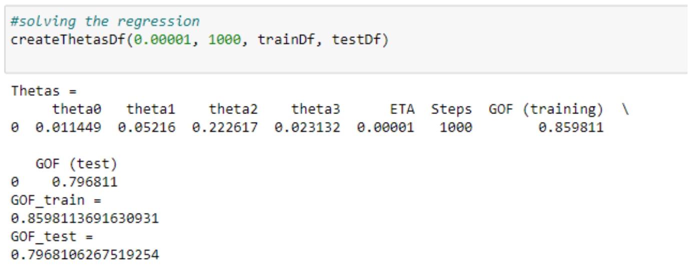


Figure 10: Sales modelled by a linear function of TV and radio: regressor and test set graph

3.3.2 Regressor 2

For Regressor 2, the training and test goodness of fit values are 0.86 and 0.80 respectively(Figure 11). So the training value is only marginally better than for the two-feature case and the test value is markedly worse. Of course, these are only sample values (for the random training and test sets selected), but they do seem to indicate that little value is added to the predictor by including newspaper advertising as a feature.



```
#solving the regression
createThetasDf(0.00001, 1000, trainDf, testDf)
```

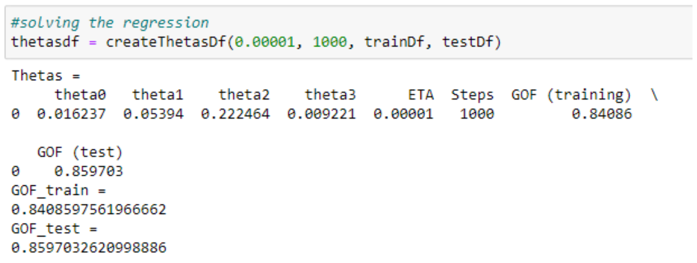
```
Thetas =
     theta0   theta1    theta2    theta3      ETA  Steps  GOF (training)  \
0  0.011449  0.05216  0.222617  0.023132  0.00001   1000        0.859811

   GOF (test)
0    0.796811
GOF_train =
0.8598113691630931
GOF_test =
0.7968106267519254
```

Figure 11: Sales modelled by a linear function of TV, radio and newspaper: regressor parameters table

3.3.3 Regressor 3

Regressor 3 tests if the introduction of a non-linear term into the line of regression improves goodness of fit. The non-linear term was (TV*radio/100), with the scale factor of 1/100 introduced to obviate overflow issues in the calculations. Here the training and test goodness of fit values are 0.84 and 0.86 respectively (Figure 12), which are essentially no better than those for for Regressor 1. Consequently there is no compelling evidence to drive the inclusion of the non-linear term (the product of TV and radio sales) in the prediction function. The θ_3 value of 0.0092 confirms the lack of need for this term.



```
#solving the regression
thetasdf = createThetasDf(0.00001, 1000, trainDf, testDf)
```

```
Thetas =
     theta0   theta1    theta2    theta3      ETA  Steps  GOF (training)  \
0  0.016237  0.05394  0.222464  0.009221  0.00001   1000         0.84086   

   GOF (test)  
0    0.859703  
GOF_train =
0.8408597561966662
GOF_test =
0.8597032620998886
```

Figure 12: Regression with extra TV*radio term: regressor parameters table

Figure 13, Figure 14, Figure 15, Figure 16, Figure 17, Figure 18, Figure 19 and Figure 20 provide further illustration. In each of these graphs, Regressor 3 is seen from a particular angle. In Figure 19 and Figure 20, especially, it can be seen that the data points follow a linear pattern, in contrast to the convoluted shape of the regressor.

3.4 Summary and Conclusion

In this section we have examined the advertising dataset with respect to three regressors. We conclude that predicting sales linearly by TV and radio advertising gives the best result. There is a high goodness of fit that is not significantly increased by the addition of a linear newspaper term nor by a term in 'TV*radio'.

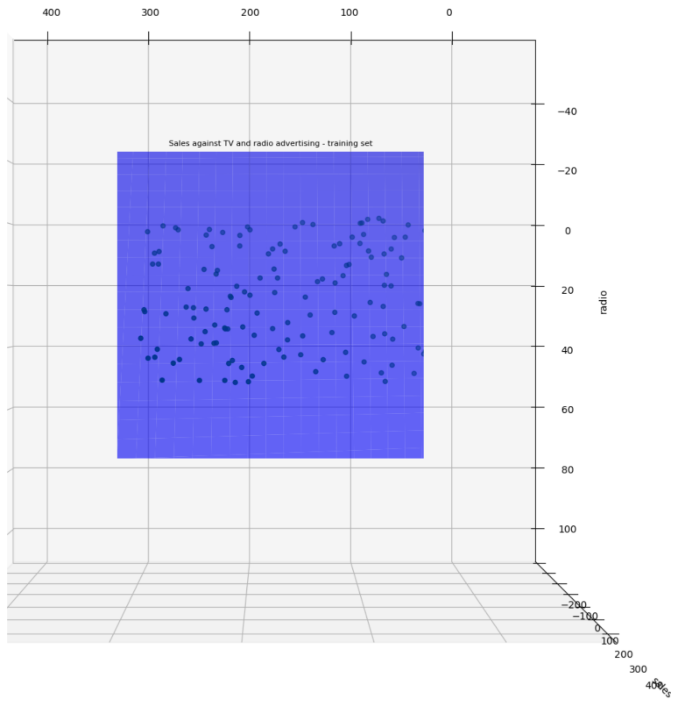


Figure 13: Regression with extra TV*radio term: regressor and training set graph: View 1

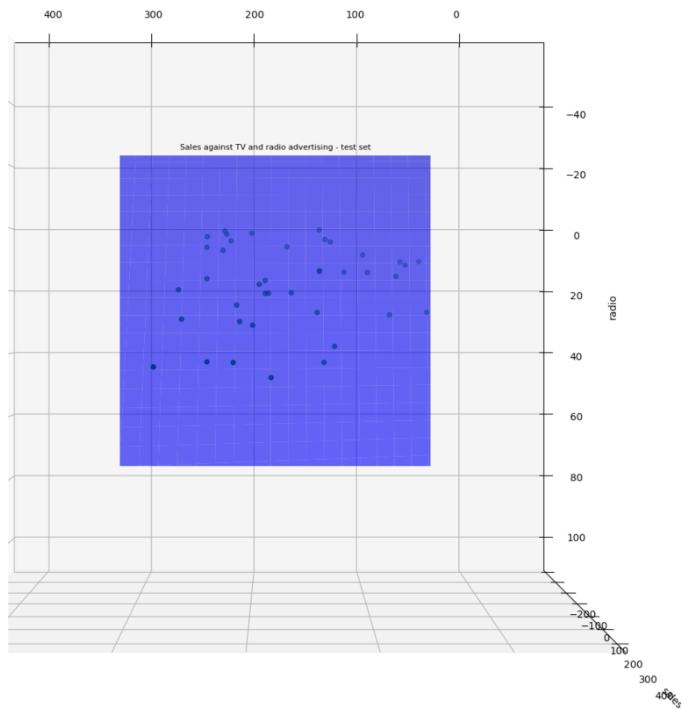


Figure 14: Regression with extra TV*radio term: regressor and test set graph: View 1

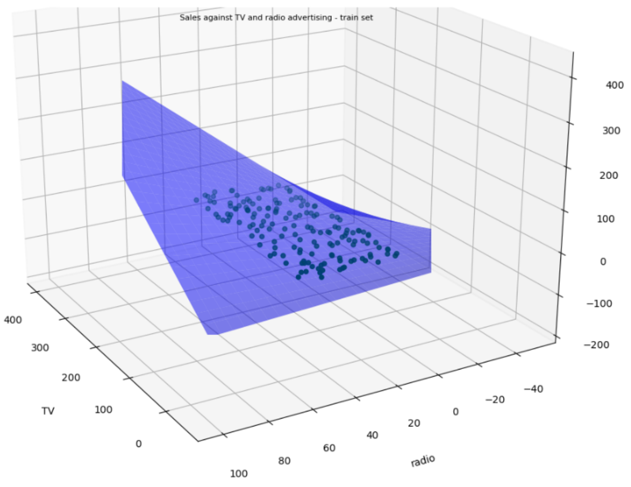


Figure 15: Regression with extra TV*radio term: regressor and training set graph: View 2

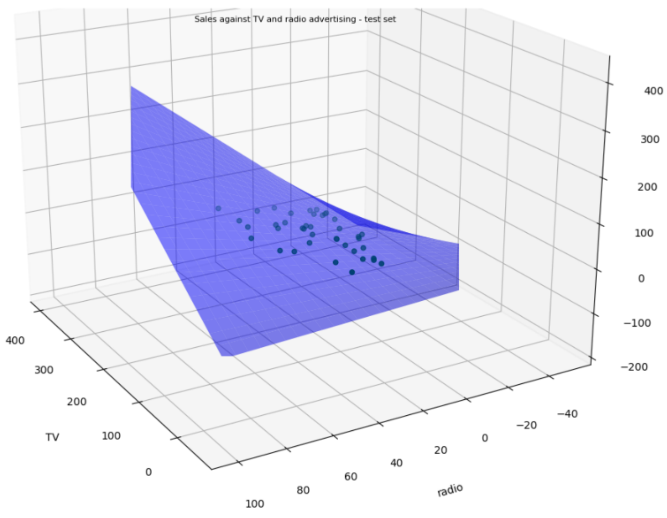


Figure 16: Regression with extra TV*radio term: regressor and test set graph: View 2

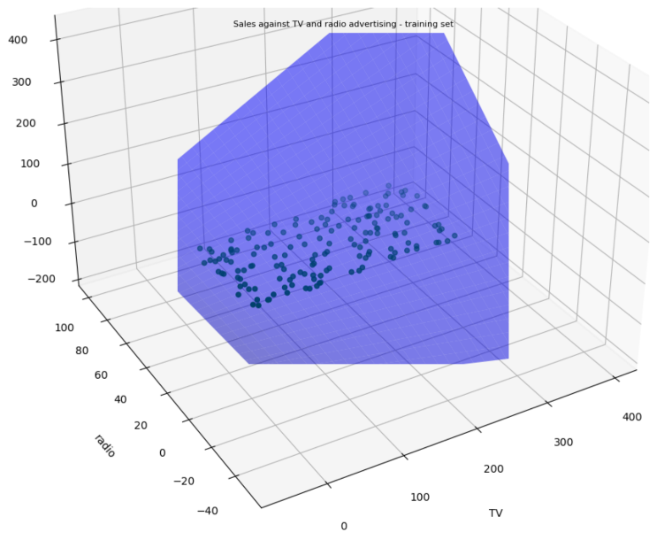


Figure 17: Regression with extra TV*radio term: regressor and training set graph: View 3

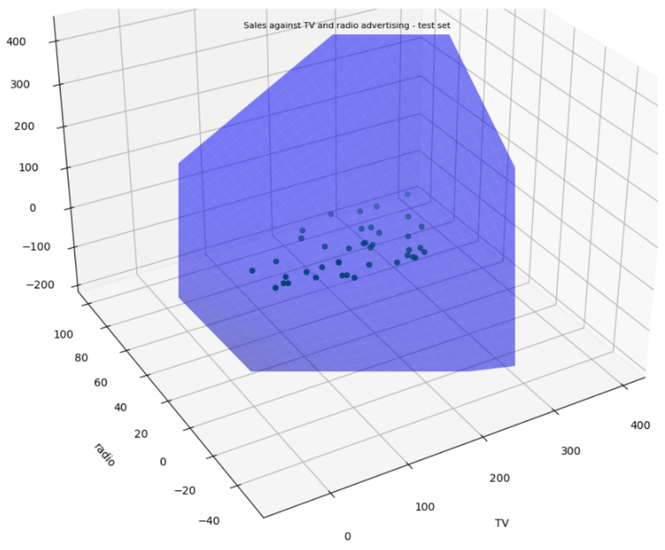


Figure 18: Regression with extra TV*radio term: regressor and test set graph: View 3

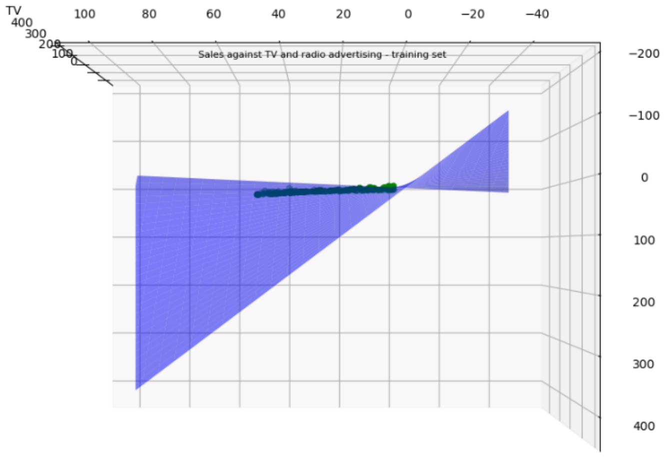


Figure 19: Regression with extra TV*radio term: regressor and training set graph: View 4

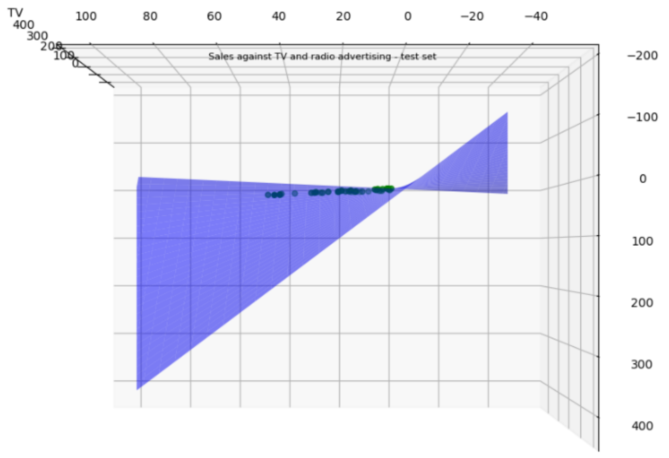


Figure 20: Regression with extra TV*radio term: regressor and test set graph: View 4

4 Analysis of the Auto Dataset

4.1 Exploratory Data Analysis

With the Auto Dataset our main focus of enquiry has been to investigate and establish a relationship between the mpg and horsepower column. We started our exploratory Data Analysis by plotting a scatter-plot between mpg and horsepower.

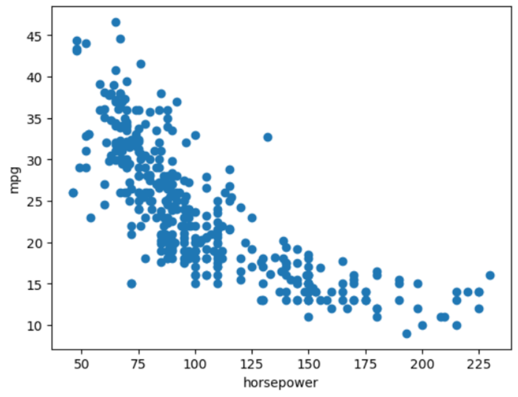


Figure 21: horsepower vs mpg

From Figure 21 it is clear that there is an inverse relationship between mpg and horsepower i.e. as horsepower increases mpg decreases. We are will try to determine if mpg is a linear or quadratic or quintic function of horsepower.

The Auto dataset has 9 columns (mpg, cylinders, displacement, horsepower, weight, acceleration, year, origin, name). Upon investigation we found that the horsepower column has 5 missing data points which were being populated as "?" as a result the whole column was being processed as string type along with the name column.

4.2 Data Processing

We took the following strategies to process and transform data into an appropriate dataFrame for processing.

- Remove columns that are not relevant to our investigation. (cylinders, displacement, weight, acceleration, year, origin, name)
- Remove the 5 missing horsepower data points.
- Convert the data type of all columns to floating point integers
- Normalize the data

Rationale behind removing the 5 missing data points
The 5 data points account for approximately 1% of the the total data-points. We decided that removing the those data-points will not have a significant impact in fitting the curve as opposed to replacing the missing values with the average.
Rationale behind Normalization
While working on the quadratic and degree 5 relationships we realised that the higher order features have a disproportionate impact on determination of the parameters. Additionally it slowed down our algorithm, often working with numbers that were too large to compute, which resulted in faulty calculations. Rationalizing the features and the target helped us compute faster and return more appropriate results.

4.2.1 Required code modification

We have created different pipelines to process each scenario separately. As a result each pipeline has their own distinct variable name and function names even though the variables and functions are identical in functionality.
For example, computationV2Multipoint, computationV2quadratic, computationV2quadratic2, computationV2degree5, computationV2degree52 are exactly the same functions but the function names and some of the variable names has been changed for better readability of the code and to avoid any future confusion inside the program.

4.2.2 Scenarios

1. Examining Linear Relationship.
   For the linear Relationship we tried fitting the data to this equation:
   $$y = \theta_0 + \theta_1 x \tag{8}$$
2. Examining Quadratic Relationship.
   For the Quadratic Relationship we tried fitting the curve to 2 equations:
   $$y = \theta_0 + \theta_1 x + \theta_2 x^2 \tag{9}$$
   and
   $$y = \theta_0 + \theta_1 x^2 \tag{10}$$
3. Examining Degree five Relationship. For the Degree five Relationship we tried fitting the curve to 2 equations:
   $$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 \tag{11}$$
   and
   $$y = \theta_0 + \theta_1 x^5 \tag{12}$$

4.2.3 Training and Testing Data

Training data consists of random 75% of the total processed data sets. Once the training and test data has been split using the select_rows function, the training data is fed into our algorithm, and the θ values are generated. Using the θ values we generate the equation and plot the curves.

To test our model, we take the features in the test data and input in the equation generated by the model. These are the targets predicted by the model. The Mean of Absolute values of the difference between predicted values and Observed values should be good indication of how well the model works.

4.3 Results, Analysis and Discussion

4.3.1 Analysis, Plots and Results

Linear

We have used computationV2Multipoint to compute the θs of (8). The function gives us the change of θs and Goodness of Fit over the number of steps in a DataFrame. η has been taken as 0.1. We checked ran the iterations for 2000 steps and traced the graph (Figure 22) using the dataframe.

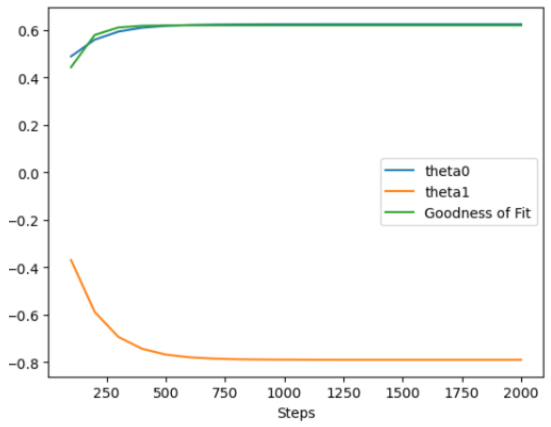


Figure 22: Variation of Goodness of fit with no of steps for fixed eta for Linear Relationship

As per the graph (Figure 22), η maximized at 0.619960 at about 2000 steps and the θ values for that step is 0.624626 and -0.789422. We substitute these values in equation (8) and plot graph [23]. Additionally we have obtained θ values using the scikit-learn SGDRegressor function and used them to plot another line to validate our results.

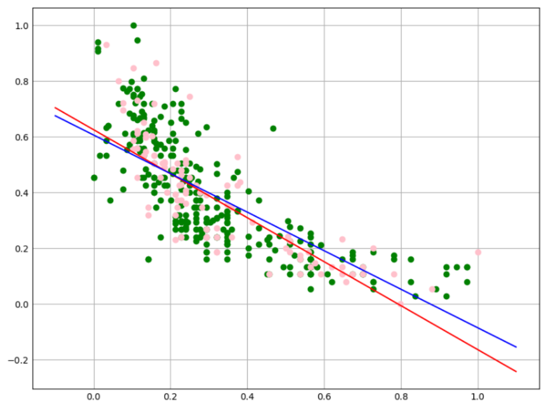


Figure 23: Linear relationship between mpg and horsepower

The red line represents the plot drawn using θ obtained using our algorithm, The blue line represents the plot drawn using θ obtained using Scikit-Learn Library. The Green scatter points represents the Training data point and the pink scatter points represents the Testing data points.

Quadratic
We have used computationV2quadratic and computationV2quadratic2 to compute the θs of 9 and 10. η has been taken as 0.1 and we ran the iterations for 50000 steps and traced graphs [24] and [25] using the dataframe.

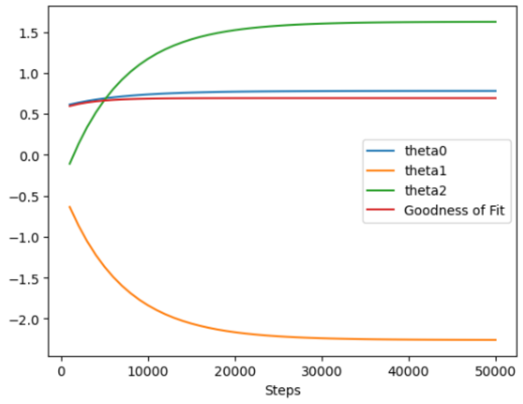


Figure 24: Variation of Goodness of fit with no. of steps for fixed eta for the long form quadratic expression

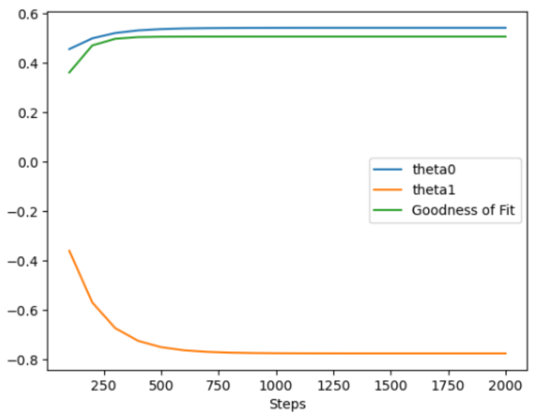


Figure 25: Variation of Goodness of fit with no. of steps for fixed eta for the short form quadratic expression

As observed, the θs obtained from 10, for all practical purposes converged and gave a Goodness of Fit of 0.5067 whereas the θs obtained from 9 converged and gave a Goodness of Fit of 0.6920. Therefore, 9 is a better fit to the dataset and we proceed to evaluate the results of 9 further.

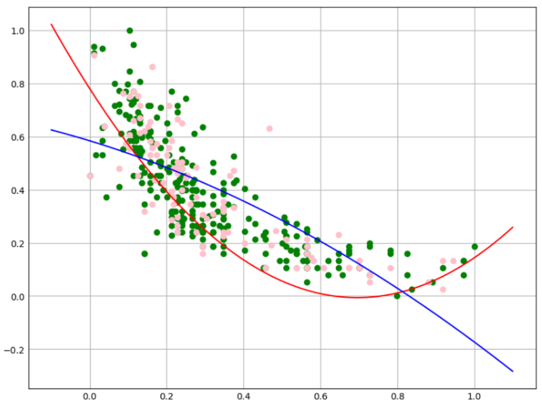


Figure 26: Quadratic relationship between mpg and horsepower

We plot 9 with θ_0 = 0.780413, θ_1 = -2.259908 and θ_2 = 1.623187. In figure [26] red line represents the plot drawn using θ obtained using our algorithm, The blue Line represents the plot drawn using θ obtained using Scikit-Learn Library. The Green scatter points represents the Training data point and the pink scatter points represents the Testing data points.

Degree Five

We have used computationV2degree5 and computationV2degree52 to compute the θs of 11 and 12. η has been taken as 0.1 and we ran the iterations for 50000 steps and traced graphs [27] and [28] using the dataframe.

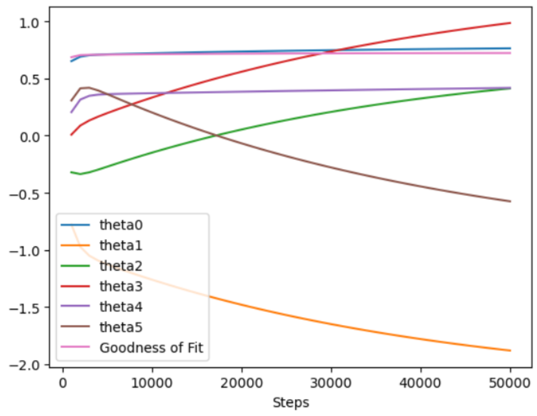


Figure 27: Variation of Goodness of fit with no of steps for fixed eta for the long form degree 5 expression

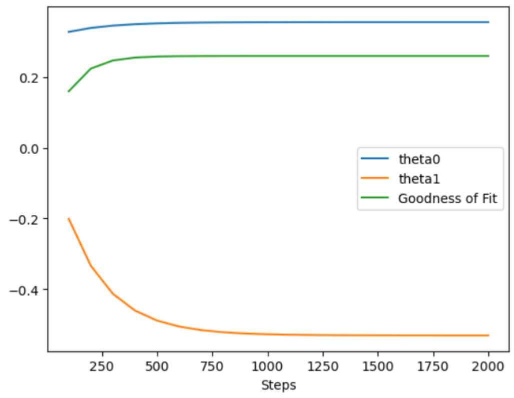


Figure 28: Variation of Goodness of fit with no of steps for fixed eta for the short form degree 5 expression

As observed, the θs obtained from 12, for all practical purposes converged and gave a Goodness of Fit of 0.2592 whereas the θs obtained from 11 converged and gave a Goodness of Fit of 0.7228. Therefore, 11 is a better fit to the dataset and we proceed to evaluate the results of 11 further.

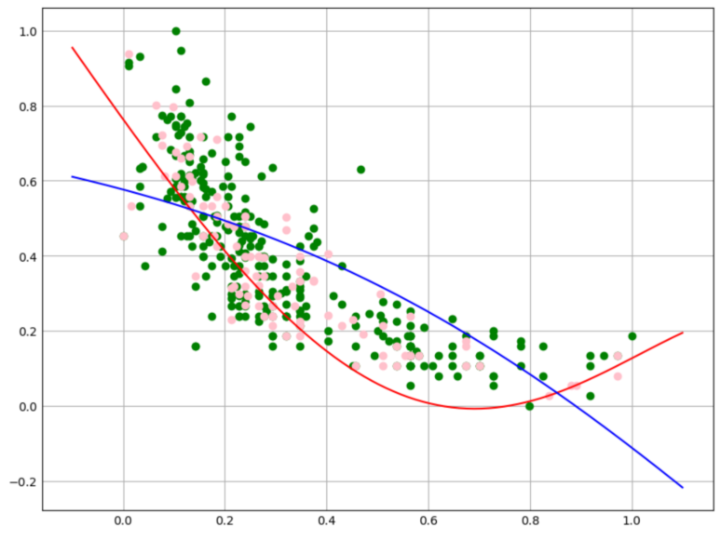


Figure 29: Degree 5 relationship between mpg and horsepower

We plot 11 with $\theta_0 = 0.763627$, $\theta_1 =$ -1.879245, $\theta_2 = 0.412939$, $\theta_3 = 0.985208$, $\theta_4 = 0.417800$, $\theta_5 =$ -0.574219. In figure [29] red line represents the plot drawn using θ obtained using our algorithm, The blue line represents the plot drawn using θ obtained using Scikit-Learn Library.

4.3.2 Discussion and Conclusion

Equation	Goodness of Fit	Mean of Absolute difference between observed and predicted Values
Linear	0.6199 or 62%	0.865
Quadratic	0.6920 or 69.2%	0.128
Degree Five	0.7229 or 72.29	0.096

Table 1: Objective comparison of the curves

Through Visual Inspection, we can conclude that the curves given by Degree Five [11] and quadratic [9] equations fit the data-points more accurately.

However, the Degree Five equation [11] has the best Goodness of Fit value among all the equations. Additionally it has the lower value of Mean of Absolute differences between observed and predicted values among degree five and quadratic equations.

This leads us to conclude that the below expression best fits the Auto Dataset:
$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$
where x is the horsepower and y is the mpg.

References

[1] Fernando J. R-Squared: definition, calculation formula, uses, and limitations[Internet]. New York, Edmonton: Investopedia; 2023 [updated 2023 Apr 08; cited 2023 Oct 17]. Available from: https://www.investopedia.com/terms/r/r-squared.asp

[2] O'Neill, C, Schutt, R. Doing data science: Straight talk from the front line. Sebastopol(CA): O'Reilly Media, Inc.; 2013.

[3] Miles, J, Shevlin, M. Applying regression and correlation: A guide for students and researchers. Los Angeles(CA), London, New Delhi, Singapore: SAGE Publications; 2001.

[4] Montgomery DC, Peck EA, Vining GG. Introduction to linear regression analysis. 3rd ed. New York, Chichester, Weinheim, Brisbane, Singapore, Toronto: John Wiley and Sons, Inc.; 2001.

[5] Li X, Wong W, Lamoureux EL, Wong TY. Are linear regression techniques appropriate for analysis when the dependent (outcome) variable is not normally distributed? Investigative Ophthalmology & Visual Science[Internet]. 2012 May[cited 2023 Oct 21]; 53:3082-3083. Available from: https://iovs.arvojournals.org/article.aspx?articleid=2128171

[6] Wikimedia Foundation, Inc. Gradient descent[Internet]. San Francisco (California): Wikipedia; 2023 Sep 28[cited 2023 Oct 18]. Available from: https://en.wikipedia.org/wiki/Gradient_descent

[7] Peter Sherar & Stuart Barnes. Advanced Java and Advanced Python. Assignemnt Brief[Internet]. 2023 Oct 6 [cited 2023 Oct 20]. Cranfield University, Cranfield. Available from: https://canvas.cranfield.ac.uk/courses/27657/pages/assessment-and-feedback

A Contributions:

A.1 Contributions by Aritra Das Ray

1. Built the first Iteration of the main Algorithm.
2. Built the code for Auto Dataset.
3. Processed, Analysed and Reproted findings on the Auto Data-set.

A.2 Contributions by Johnathan Puddicombe

1. Built the Second and final Iteration of the main Algorithm with significant upgrades.

2. Built the code for Advertising Dataset.
3. Processed, Analysed and Reported findings on the Advertising Data-set.

B Main Function (v0) Code

```
def computation(nparray, eta, steps):
    theta0, theta1, theta2, theta3 = 0,0,0,0
    n=len(nparray)
    a=1
    i=0
    sum0,sum1,sum2,sum3 = 0,0,0,0

    while a<=steps:
        while i<n:
            sum0 = sum0 + (theta0 + theta1*nparray[i][0] + theta2*nparray[i][1]
            sum1 = sum1 + (theta0 + theta1*nparray[i][0] + theta2*nparray[i][1]
            sum2 = sum2 + (theta0 + theta1*nparray[i][0] + theta2*nparray[i][1]
            sum3 = sum3 + (theta0 + theta1*nparray[i][0] + theta2*nparray[i][1]
            i=i+1

        delta_theta0 = (eta*2*sum0)/n
        delta_theta1 = (eta*2*sum1)/n
        delta_theta2 = (eta*2*sum2)/n
        delta_theta3 = (eta*2*sum3)/n

        theta0 = theta0 - delta_theta0
        theta1 = theta1 - delta_theta1
        theta2 = theta2 - delta_theta2
        theta3 = theta3 - delta_theta3

        a=a+1

    return theta0, theta1, theta2, theta3
```

C Main Function (v1) Code

```
def computationV2(featureListTargetDf, eta, numberOfSteps):
    featureListPlusOne = featureListTargetDf['featureListPlusOne']
    numberOfFeaturesPlusOne = len(featureListPlusOne[0])
    thetaValues = [0]*numberOfFeaturesPlusOne
    step = 0
    while step < numberOfSteps:
        theta_iter = 0
        while theta_iter < numberOfFeaturesPlusOne:
            thetaValues[theta_iter] = thetaValues[theta_iter] - eta*calcDiff(fea
            theta_iter = theta_iter+1
        step = step+1
    return(thetaValues)
```

D Main Function (v2) Code

```
def computationV2(featureListTargetDf, eta, numberOfSteps):
    featureListPlusOne = featureListTargetDf['featureListPlusOne']
    numberOfFeaturesPlusOne = len(featureListPlusOne[0])
    thetaValues = [0]*numberOfFeaturesPlusOne
    step = 0
    while step < numberOfSteps:
        theta_iter = 0
        while theta_iter < numberOfFeaturesPlusOne:
            thetaValues[theta_iter] = thetaValues[theta_iter] - eta*calcDiff(fea
            theta_iter = theta_iter+1
        step = step+1
    return(thetaValues)
```

E Helper Functions Code

```
def calcDiff(featureListTargetDf, thetaValues, theta_iter):
    featureListPlusOne = featureListTargetDf['featureListPlusOne']
    target = featureListTargetDf['target']
    numberOfFeaturesPlusOne = len(featureListPlusOne[0])
    numberOfDataPoints = len(featureListTargetDf)
    data_point_iterator = 0
    running_total = 0
    while data_point_iterator < numberOfDataPoints:
        featureListPlusOneValues = featureListPlusOne[data_point_iterator]
        targetValue = target[data_point_iterator]
        featureListPlusOneValue = featureListPlusOneValues[theta_iter]
        discrepancy = (estimatedTargetValue(featureListPlusOneValues, thetaValue
        running_total = running_total + discrepancy
        data_point_iterator = data_point_iterator + 1
    return (2/numberOfDataPoints)*running_total

def estimatedTargetValue(featureListPlusOneValues, thetaValues):
    running_total = 0
    feature_iter = 0
    while feature_iter < len(featureListPlusOneValues):
        running_total = running_total + thetaValues[feature_iter]*featureListPlus
        feature_iter = feature_iter+1
    return running_total
```

E.1 Goodness of Fit Code

```
def goodnessOfFit(featureListTargetDf, thetaValues):
    featureListPlusOne = featureListTargetDf['featureListPlusOne']
    target = featureListTargetDf['target']
    numberOfFeaturesPlusOne = len(featureListPlusOne[0])
    numberOfDataPoints = len(featureListTargetDf)
    sumOfSquares = 0
    data_point_iterator = 0
    while data_point_iterator < numberOfDataPoints:
        featureListPlusOneValues = featureListPlusOne[data_point_iterator]
        targetValue = target[data_point_iterator]
        discrepancy = (estimatedTargetValue(featureListPlusOneValues, thetaValue
        sumOfSquares = sumOfSquares + discrepancy**2
        data_point_iterator = data_point_iterator+1
    gOF = 1 - sumOfSquares/((numberOfDataPoints-1)*(target.var()))
    return gOF
```

E.2 Iteration of Steps Code

```
#Code for generating outputs from various step numbers with constant eta.
#comment
steps = 10
ETA = 0.00001
STEP_GAP = 10
thetas = []
while steps < 1001:
    print(steps)
    result = computationV2(adv_dataframe, ETA, steps)
    GOF = goodnessOfFit(adv_dataframe,result)
    #print(GOF)
    result.append(ETA)
    result.append(steps)
    result.append(GOF)
    thetas.append(result)
    steps = steps + STEP_GAP
thetasValidatedf = pd.DataFrame(data=thetas,columns=['theta0', 'theta1', 'theta2
print(thetasdf)
```

E.3 Iteration of Steps Code (v2)

```
#Combining iteration of steps code and computationV2 code into a
#single function to time-optimise as O($steps$) rather than O($steps^2$)

def computationV2Multipoint(featureListTargetDf, eta, step_gap, iterations):
    featureListPlusOne = featureListTargetDf['featureListPlusOne']
    numberOfFeaturesPlusOne = len(featureListPlusOne[0])
    thetaValues = [0]*numberOfFeaturesPlusOne
    thetasList = []
    iteration = 0
    step = 0
    while iteration < iterations:
        iteration_step_count = 0
        while iteration_step_count < step_gap:
            theta_iter = 0
            while theta_iter < numberOfFeaturesPlusOne:
                thetaValues[theta_iter] = thetaValues[theta_iter] - eta*calcDiff
                theta_iter = theta_iter+1
            step = step+1
            iteration_step_count = iteration_step_count+1
        result = thetaValues.copy()
        GOF = goodnessOfFit(adv_dataframe,result)
        result.append(eta)
        result.append(step)
        result.append(GOF)
        #print("theta values = \n")
        #print(thetaValues)
        thetasList.append(result)
        iteration = iteration+1
    thetasdf = pd.DataFrame(data=thetasList,columns=['theta0', 'theta1', 'theta2
    print(thetasdf)
    return(thetasdf)
```

E.4 Test/Train Data Select Code

```
import math
import numpy as np

#select_rows takes a numpy array and a percentage (between zero and one)
#and returns the training and test sets as numpy arrays.
```

```
#The test set size will be that percentage of the original array.

def select_rows(array, percentage):
    size = len(array)
    rows_to_select = int(math.floor(size*percentage))
    np.random.shuffle(array)
    test_rows = array[:rows_to_select]
    tr = rows_to_select - size
    train_rows = array[tr:]
    return train_rows, test_rows
```

E.5 Validation Using scikit-learn

```
#generating thetas using SciKit Learn

from sklearn.linear_model import SGDRegressor
from sklearn.metrics import r2_score

X = degree5_auto[['hp_normalized', 'hp2_normalized', 'hp3_normalized', 'hp4_norm
y = degree5_auto[['mpg_normalized']].to_numpy().ravel()

Model = SGDRegressor(eta0 = 0.1 , max_iter = 50000)

Model.fit(X,y)

theta_sk_0 = Model.intercept_[0]
theta_sk_1, theta_sk_2, theta_sk_3, theta_sk_4, theta_sk_5 = Model.coef_

sklearn_theta_model = [theta_sk_0, theta_sk_1, theta_sk_2, theta_sk_3, theta_sk_

print(sklearn_theta_model)

#fitnes
y_pred = Model.predict(X)

r2 = r2_score(y, y_pred)

print(r2)
```