Linear Regression :

Data set: **Automobile**

Target value to be predicted: **Price**

Task: We have to predict the price of the auto mobile.

Approach: Linear regression.

Metric: Root mean squared error (RMSE).

Data Preprocessing:

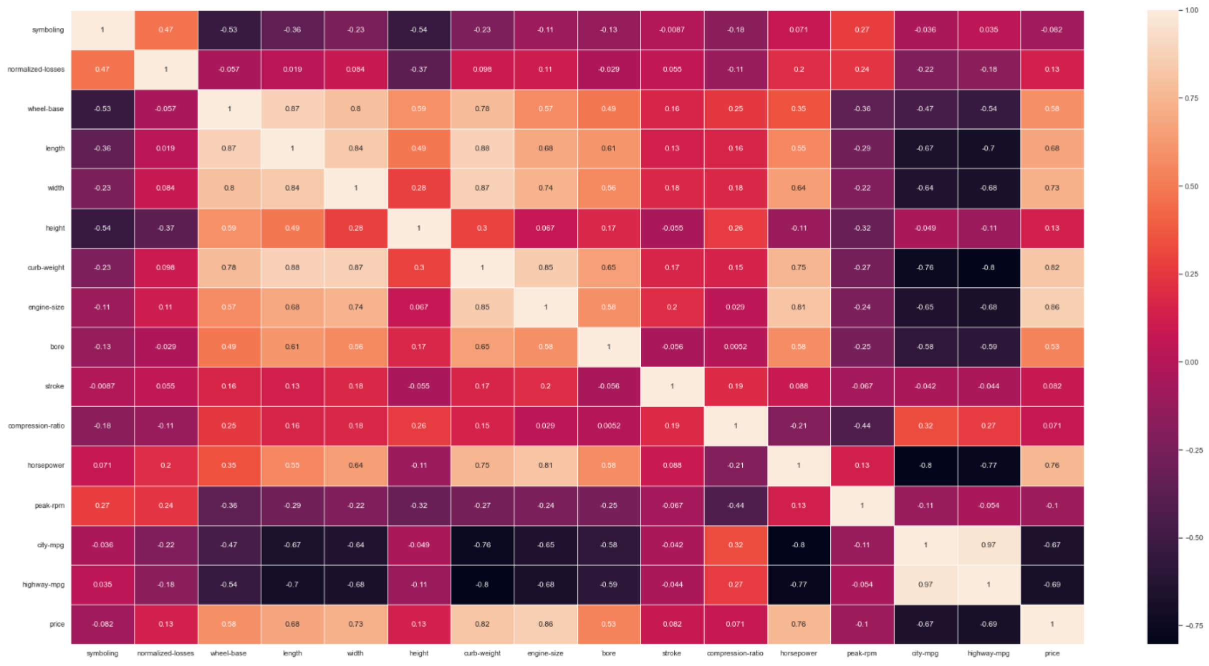
Data preprocessing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data.

Analysis of data :

- Imported the data set using pandas
- Checked for the type of data: many numerical data were object type
- Found out there is ? for NaN
- There were few textual data that needed to be encoded

Pre-processing steps:

- Conversion of some columns' datatype from object type to numeric value (numeric value represented as string)
- Replacing NaN with mean value of data
- Found correlation of features, for which feature correlation is maximum (for univariate model)
- A correlation matrix is a table showing correlation coefficients between variables.
  Heatmap is drawn for visualizing the correlation of features with target



the features with maximum correlation are : engine-size and curb-weight with correlation value of 0.86 and 0.82 respectively.

- Encoding the data to numeric value is done. Few features like make , body-style and number of door where replaced using label encoding , rest features were encoded using one-hot encoding as there where bias on the values.
- min max normalization is used because features are of different scales.

<u>Code Approaches & Test Results</u> :

Closed-Form Solution:

Normal Equation is the Closed-form solution for the Linear Regression algorithm which means that we can obtain the optimal parameters by just using a formula that includes a few matrix

multiplications and inversions. To calculate theta, we take the partial derivative of the MSE loss

function (equation 2) with respect to theta and set it equal to zero. Then, do a little bit of linear algebra to get the value of theta.

Gradient Descent Solution:

Gradient Descent is the process of minimizing a function by following the gradients of the cost function. This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value.

- The data is divided into training and testing set with 80% and 20% split respectively with shuffling .

Following models are implemented –

- **Univariate Linear regression using the inbuilt function to compare built model**

  MSE is: 0.012841620378245079
  RMSE is: 0.1133208735328363

- **Univariate Linear regression using Closed form from scratch**
  Model is implemented based on the following equation

  In matrix form,

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix},$$

  so

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}.$$

  a being the intercept and b being the slope for Y_pred=a+bx
  then after the prediction value is found , find the root mean square error using ypred and yactual in test data

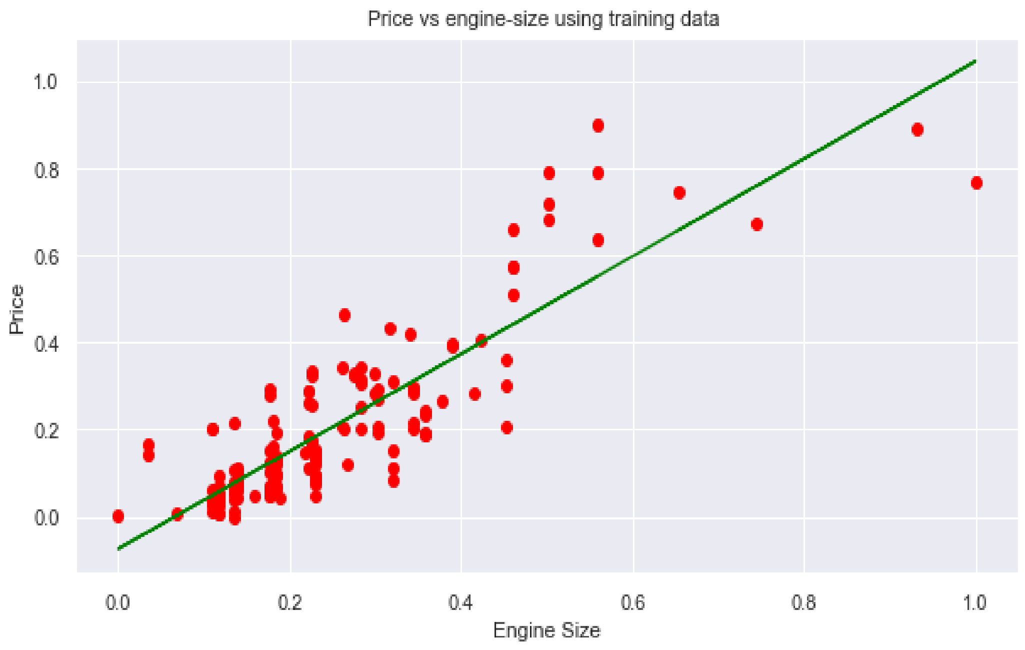
With the given data

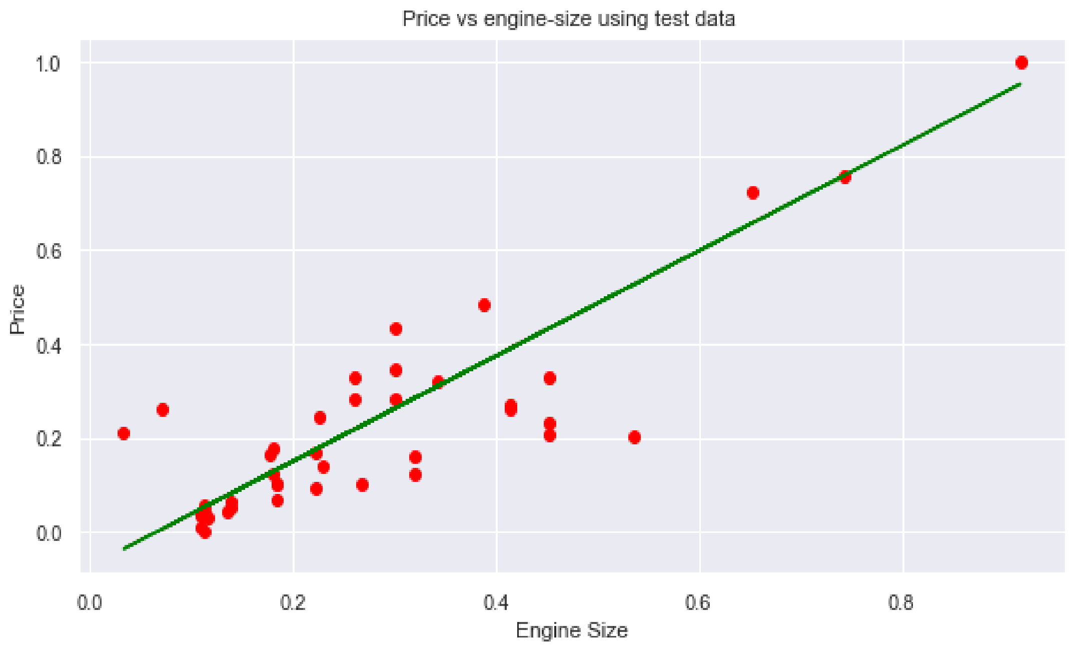
$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Final output of the model :

MSE is: 0.012841620378245079

RMSE is: 0.1133208735328363





- **Univariate Linear regression using Gradient descent form scratch**
  -by trial and error found learning rate to be a=0.009 and iterations=12000
  -used the partial derivative formula as

$$D_m = \frac{-2}{n} \sum_{i=0}^{n} x_i(y_i - \bar{y}_i) \qquad m = m - L \times D_m$$

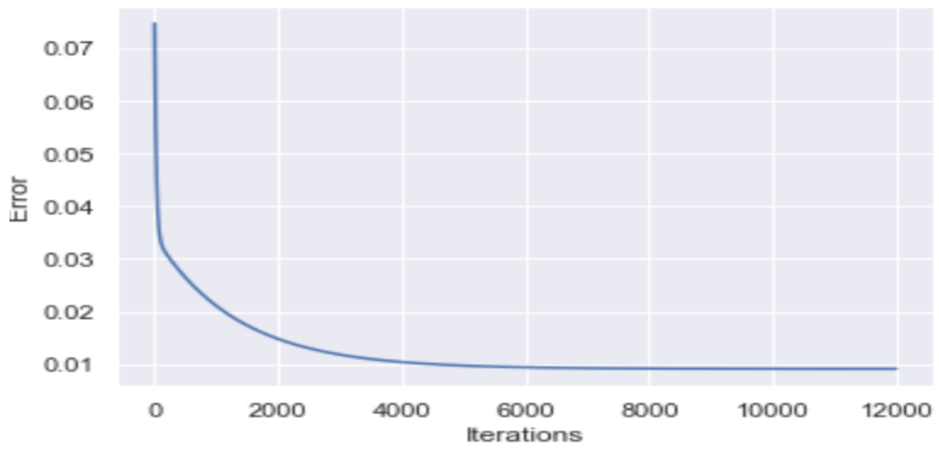
$$D_c = \frac{-2}{n} \sum_{i=0}^{n} (y_i - \bar{y}_i) \qquad c = c - L \times D_c$$

  Update the above equations in each iteration till you get desired slope and intercept
  Once parameters are found then the prediction value is found.Finally find the root mean square error using ypred and yactual in test data

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

  Final output of the model :
  The Mean Square Error(RMSE) is: 0.012701828970142576
  The Root Mean Square Error(RMSE) is: 0.11270239114651727

  Convergence graph for gradient descent :-
  Alpha:0.009
  Number of iterations : 12000



- **Multivariate Linear Regression using inbuilt function to compare built model**
  Features used for better RMSE value are :'width', 'height','length', 'curb-weight', 'engine-size', 'horsepower','normalized-losses','bore','wheel-base','fuel-system_mpfi', 'drive-wheels_rwd' , 'num-of-cylinders_twelve','num-of-cylinders_six','num-of-cylinders_eight','engine-type_ohcv'

  MSE is: 0.026984294648760725
  RMSE is: 0.164026687880233

- **Multivariate Linear Regression using closed form :**

  We have the Y_pred ~ WX
  Where X is feature ,to find the parameters W OR theta vector we need to compute

$$\theta = (X^T X)^{-1} X^T \vec{y}.$$

  Here there is problem being faced that is of pseudo-inverse we cannot say that $(X^T X)^{-1}$ is always invertible if all the features are taken into consideration.
  Solution for this was found by dropping some of the features with negative or very small correlation value with target

  Final output of the model :
  MSE is: 0.026984294648760485
  RMSE is: 0.1642689704379999

- **Multivariate Linear regresssion using gradient descent:**

  Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} [\underset{\substack{\uparrow \\ \text{Predicted Value}}}{h_\Theta(x_i)} - \underset{\substack{\uparrow \\ \text{True Value}}}{y_i}]^2$$

  Gradient Descent

$$\boxed{\Theta_j = \Theta_j - \underset{\substack{\uparrow \\ \text{Learning Rate}}}{\alpha} \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)}$$

  Now,

$$\begin{aligned} \frac{\partial}{\partial \Theta} J_\Theta &= \frac{\partial}{\partial \Theta} \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y]^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x_i) - y) \frac{\partial}{\partial \Theta_j} (\Theta x_i - y) \\ &= \frac{1}{m} (h_\Theta(x_i) - y) x_i \end{aligned}$$

  Therefore,

$$\boxed{\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(h_\Theta(x_i) - y) x_i]}$$

Final output of the model :

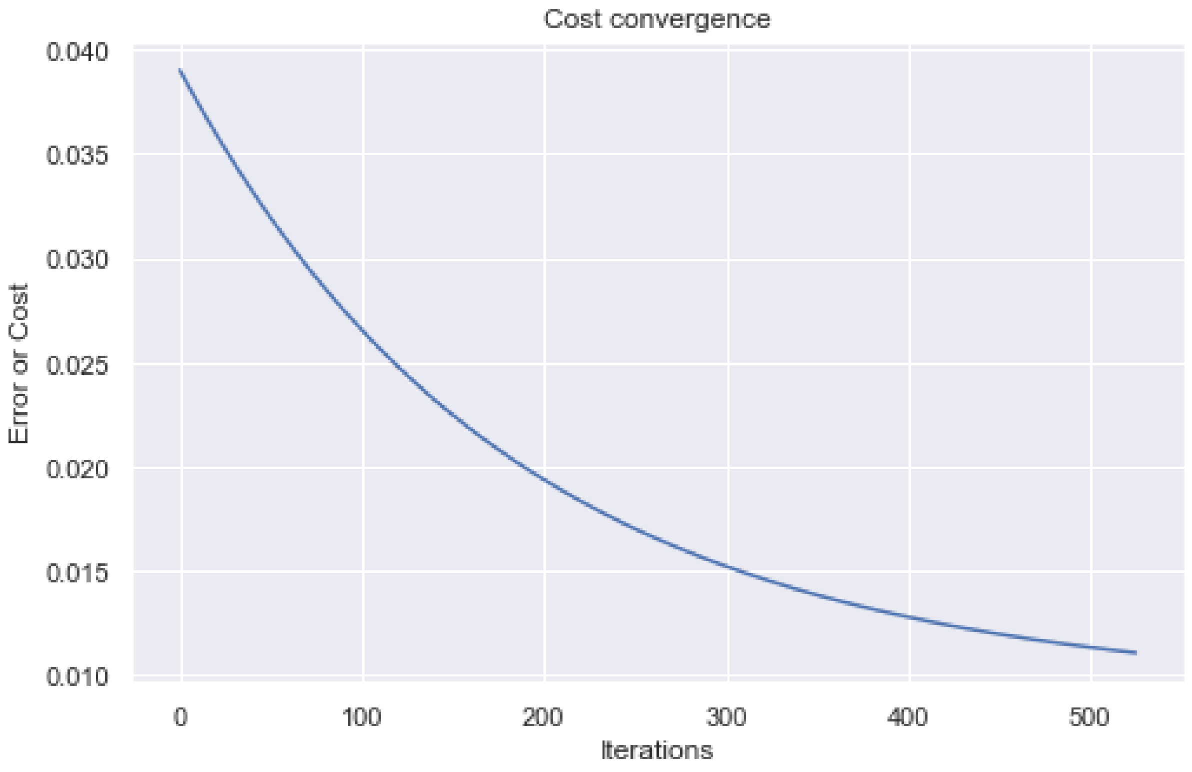
MSE is: 0.02690475433695938

RMSE is: 0.164026687880233

Cost convergence graph is as below:

For alpha=0.0009

Number of iteration =526



RMSE values of Univariate Linear regression :

Inbuilt model	Closed form	Gradient descent
0.1133208735328363	**0.1133208735328363**	**0.11270239114651727**

RMSE values of Multivariate Linear regression :

Inbuilt model	Closed form	Gradient descent
0. 164026687880233	**0.1642689704379999**	**0.164026687880233**

<u>Logistic Regression :</u>

Data set: **ionosphere data set**

Target value to be predicted: **35th column in data set which tells ionosphere is good or bad**

Approach: Logistic regression.

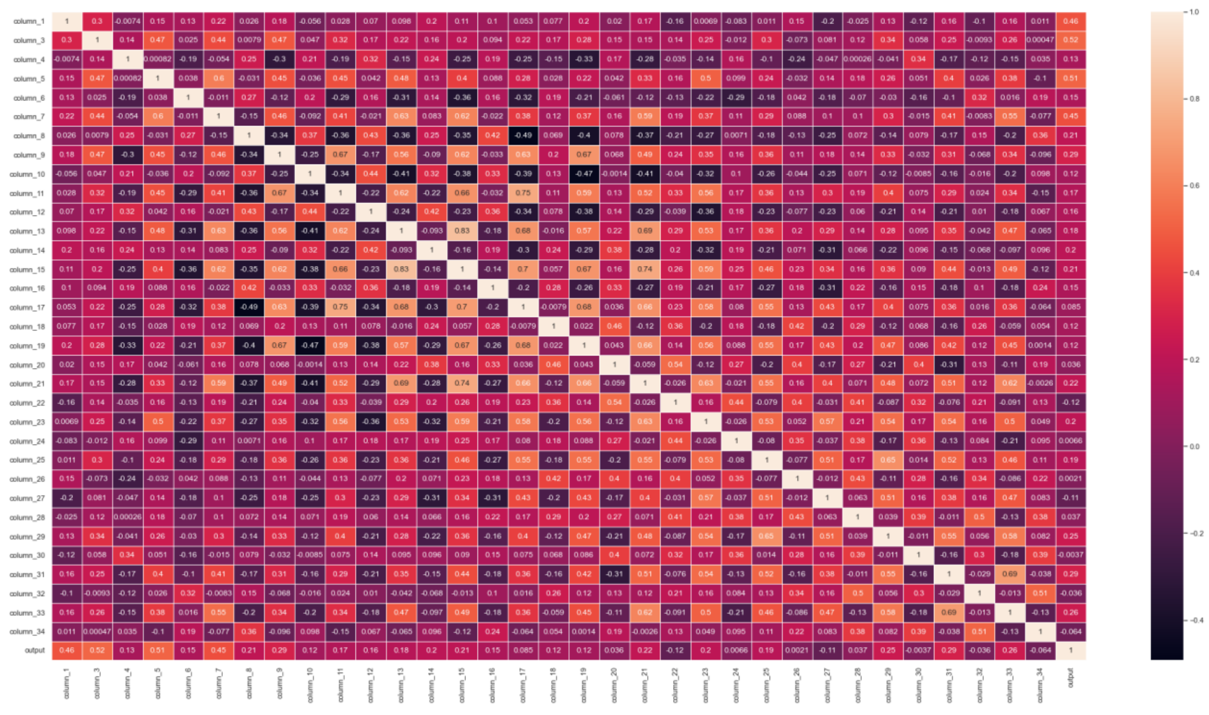
Metric: Accuracy

<u>Analysis of data :</u>

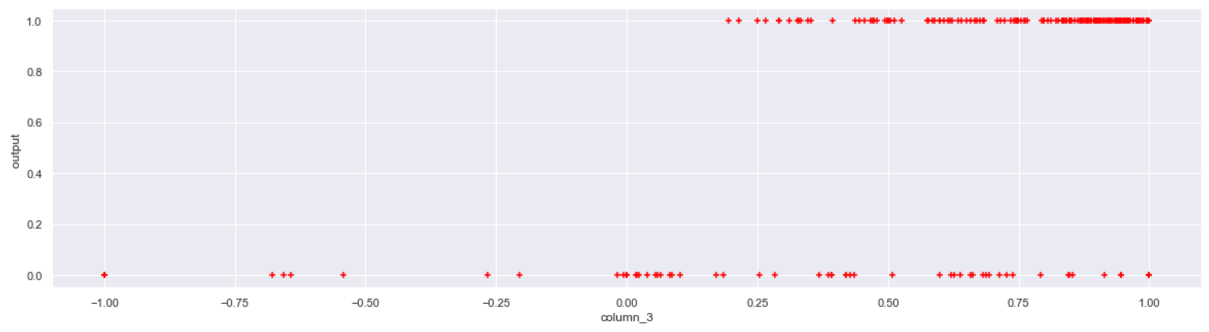
- Imported the data set using pandas
- Checked for the type of data: all where numerical except the 35th column
- No NaN were found .
- The 35th column was textual needed to be encoded
- No headers were found
- Duplicate row was found

<u>Pre-processing steps:</u>

- Added the headers as column_i for ith column.
- Column_2 was entirely null .So dropped it
- Found correlation of features, for which feature correlation is maximum (for univariate model) as column_3 , column_5 and column_7 with highest correlation of column_3
- A correlation matrix is a table showing correlation coefficients between variables.
  Heatmap is drawn for visualizing the correlation of features with target



- Scatter plot is drawn to visualize the data



<u>Train-Test Splitting:</u> We have taken 80% of the data for training and the rest

20% for testing.

<u>Code Approaches & Test Results</u> :

Logistic regression is the same as linear regression except that linear regression can give output as any value, but logistic regression can only give values between 0 and 1 as the probability of a certain class label.

So till now, we have used y = m*x+c line for linear regression. But here will use the sigmoid function for that line which is

Sigm(x) = 1 / 1+ e^(-x)

,here x will be m*x + c.

We can extend this formula for multivariate as well by Y=W.T * X.

For multivariate logistic regression it will be 1 / 1+(- W.T * X).

We are trying to get the best W that fits the sigmoid function so we can here use the maximum likelihood probability function for all the values of X.

$$\ln L(p) = \sum_{i=1}^{N} [y_i \ln p + (1 - y_i) \ln(1 - p)]$$

Unregularised JVal (Summation)

$$\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \ \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \ \log(1 - h_\theta(x^{(i)}))]$$

Unregularised JVal (Vectorised)

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1 - y)^T \log(1 - h)\right)$$

Following models were implemented :

- Univariate Logistic regression using sklearn
  Feature used :column_3

  Accuracy on test set by sklearn model : **88.57142857142857 %**

- Univariate Logistic regression using gradient descent
  Feature used :column_3
  Accuracy on test set by the univariate logistic regression model **: 81.42857142857143 %**

- Multivariate Logistic regression using sklearn

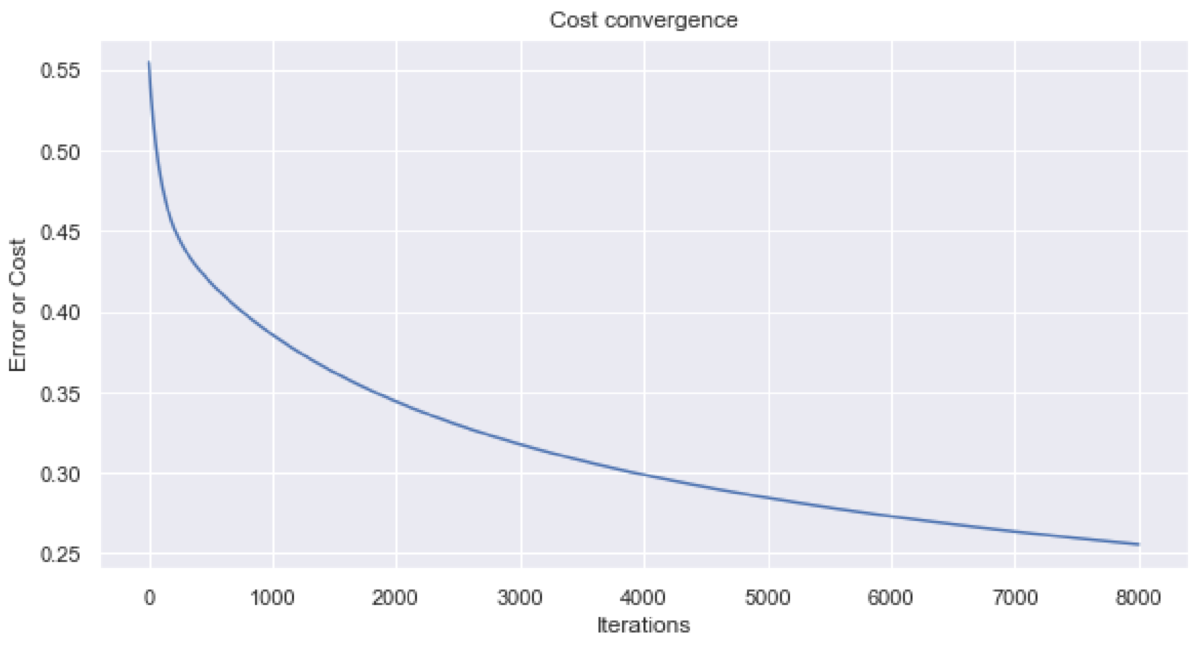
  Accuracy on test set by sklearn model : **85.71428571428571 %**

- Multivariate Logistic regression using gradient descent model

  Accuracy on test set by Multivariate logistic regression model with all features : **81.42857 142857143 %**

  Accuracy on test set by Multivariate logistic regression model with some features : **82.8 5714285714286 %**

  Alpha: 0.004
  Number of iteration : 8000
  Convergence graph is given below



Univariate using Sklearn	Univariate using model
88.57142857142857 %	**81.42857142857143 %**

Multivariate using Sklearn	Multivariate using model with few features	Multivariate using model with all features
85.71428571428571 %	**82.85714285714286 %**	**81.42857142857143 %**

Naïve Bayes Approach :

Data set: **ionosphere data set**

Target value to be predicted: **35th column in data set which tells ionosphere is good or bad**

Approach: Naïve bayes theorem .

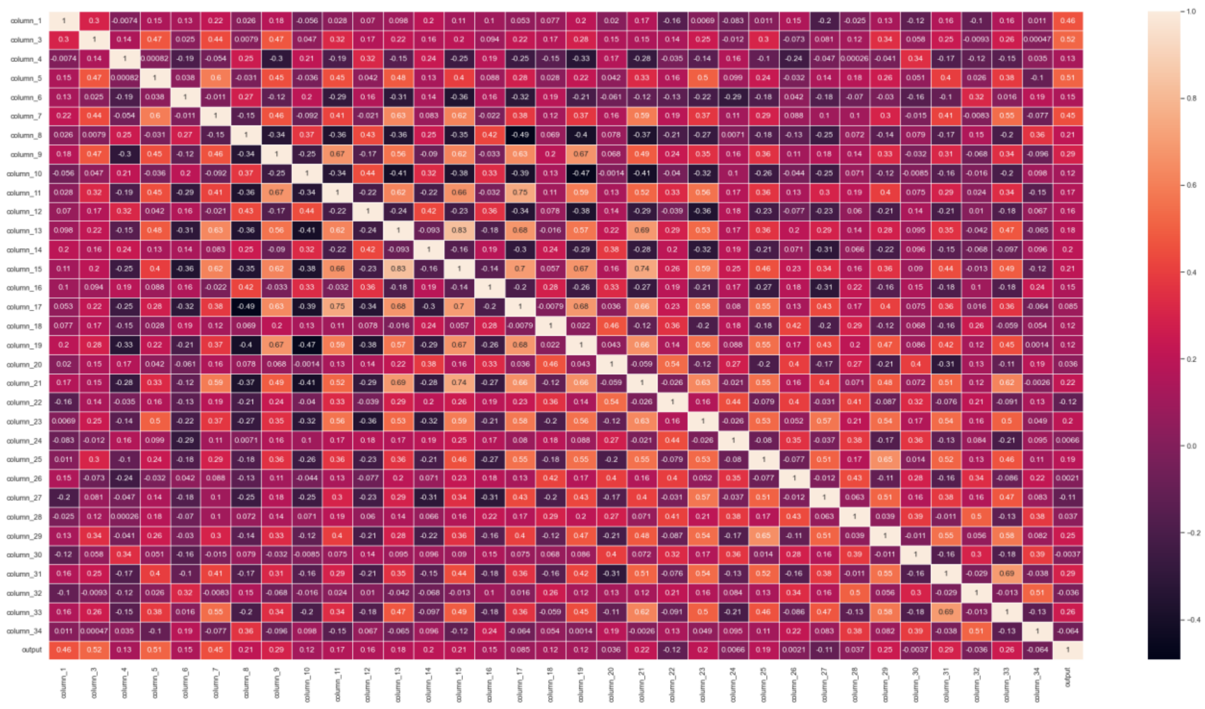
Metric: Accuracy

Analysis of data :

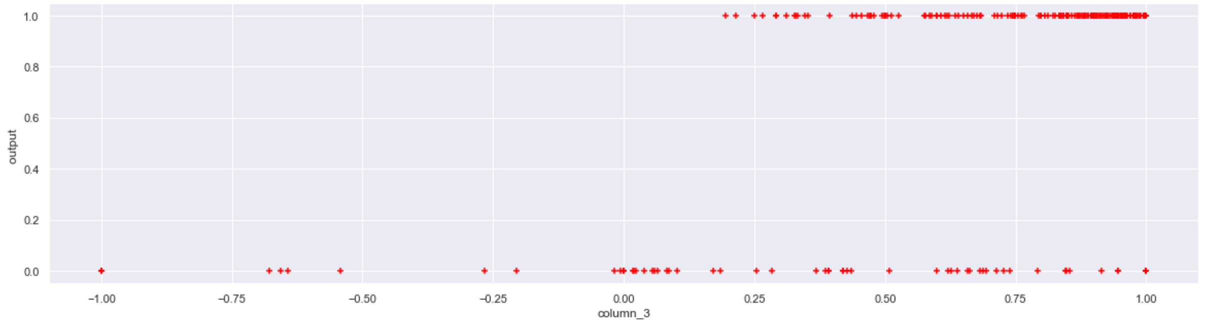
- Imported the data set using pandas
- Checked for the type of data: all where numerical except the 35th column
- No NaN were found .
- The 35th column was textual needed to be encoded
- No headers were found
- Duplicate row was found

Pre-processing steps:

- Added the headers as column_i for ith column.
- Column_2 was entirely null .So dropped it
- Found correlation of features, for which feature correlation is maximum (for univariate model) as column_3 , column_5 and column_7 with highest correlation of column_3
- A correlation matrix is a table showing correlation coefficients between variables.
  Heatmap is drawn for visualizing the correlation of features with target



- Scatter plot is drawn to visualize the data



<u>Train-Test Splitting:</u> We have taken 80% of the data for training and the rest

20% for testing.

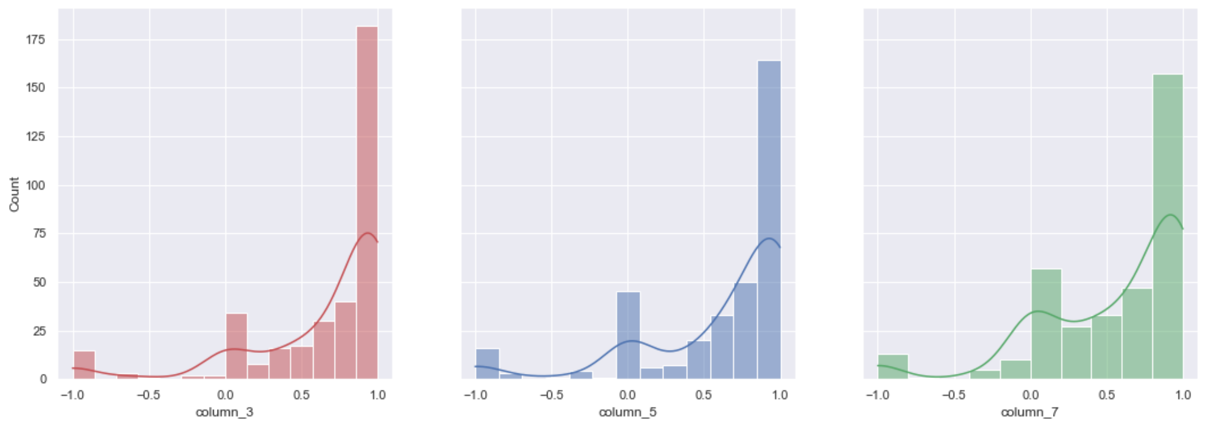
<u>Code Approaches & Test Results</u> :

Naive Bayes Classifier is a very popular supervised machine learning algorithm based on Bayes' theorem. It is simple but very powerful algorithm which works well with large datasets and sparse matrices, like pre-processed text data which creates thousands of vectors depending on the number of words in a dictionary. It works really well with text data projects like sentiment data analysis, performs good with document categorization projects, and also it is great in predicting categorical data in projects such as email spam classification.

$$\overset{\text{Posterior}}{P(A|B)} = \frac{\overset{\text{Likelihood}}{P(B|A)}\,\overset{\text{Prior}}{P(A)}}{\underset{\text{Normalizing constant}}{P(B)}}$$

$$P(B) = \sum_{Y} P(B|A)P(A)$$

<u>Guassian of the columns most correlated to target value are given below:</u>



<u>Accuracy of the model is given below :</u>

Multivariate Model using sklearn	Multivariate using naïve bayes theorem for all features	Multivariate using naïve bayes theorem for some features
84.28571428571429	68.57142857142857 %	68.57142857142857 %

Univariate using Sklearn	Univariate using Bayes theorem
84.28571428571429	88.57142857142857 %