0.1 Introduction

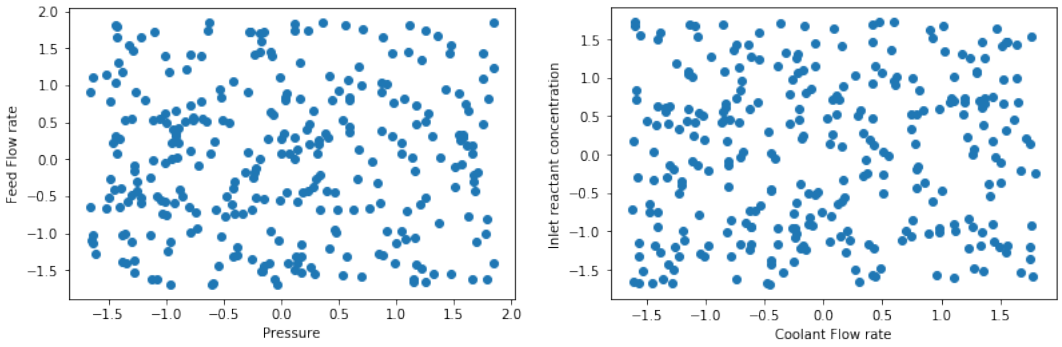
Logistic regression is used when dependant variables are categorical and not quantitative,the dataset in question considers five different independant variables that are based on an experiment conducted and there result categorizes a dependant 'TEST' which has levels as pass or fail and depending on particular condition,a logistic model developed should be able to predict whether reactor will Pass or fail with accuracy and precision.

0.1.1 Statistics of data

The first step before any model creation should be check for different statistical features of the data as follows.

Multicollinearity check

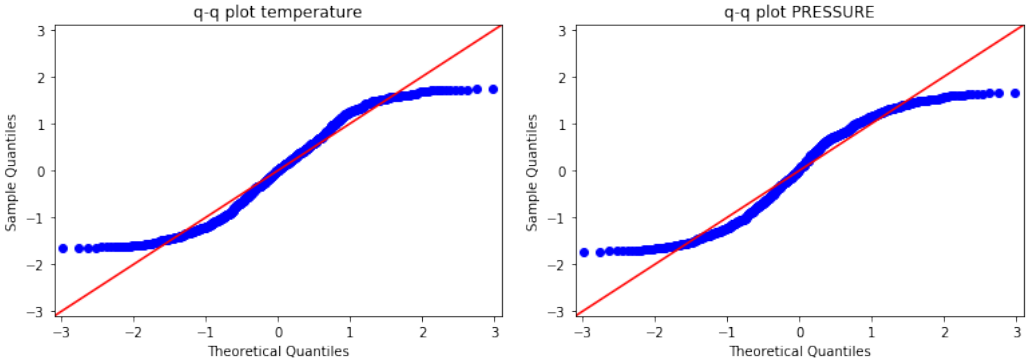
Multicollinearity means checking for relations between variables that we have assumed indpendant in the beginning.



we can see that there is no particular trend between different independant variables for all 10 combinations hence multicollinearity doesnt exist for data.

Distribution of data

Its important to check distribution of each observation parameter as taking mean centred data is always better,distribution is checked with help of quantile-quantile plot and a 45 degree line,if the observation data falls mostly on 45 degree line we can ensure assumed distrubution is correct,by default in python its normal distribution so a 45 degree line will ensure normal distribution.



It can be shown that all 5 independant observation variables follow normal distribution as most of the data falls on 45 degree line of default normal disribution.

0.2 Cost function

The most important thing is to select a cost function that can help to minimize overall cost and helps to give optimum weights/parameters for the function of logistic regression by applying gradient descent algorithm to overall cost function.

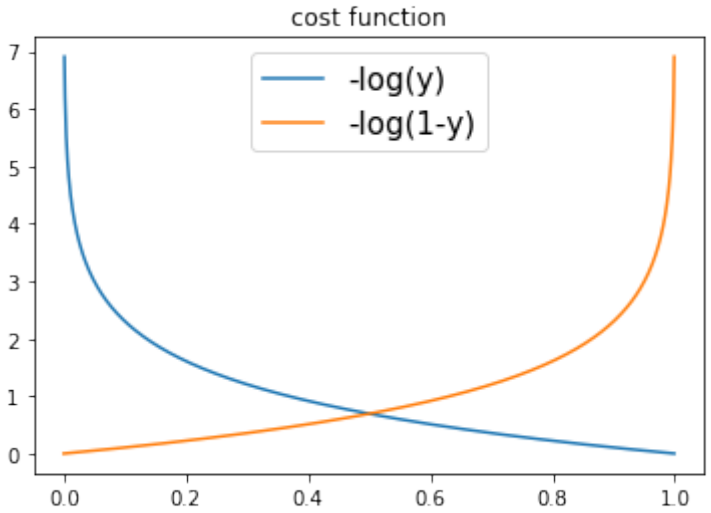
As the output is probability of success($y_i = 1$) OR failure($y_i = 0$) we have to select a cost function that will penalize success $y_i = 1$ when predicted output is $y_i^\alpha = 0$ and penalize failure $y_i = 0$ when predicted output is $y_i^\alpha = 1$.

$$L = \sum_{i=1}^{n} -y_i * log y_i^\alpha - (1 - y_i) * log(1 - y_i^\alpha)$$

$$L = \begin{cases} -log(y^\alpha), \text{if } y_i \text{ is } 1 \\ -log(1 - y^\alpha), \text{if } y_i \text{ is } 0 \end{cases}$$

where n is number of samples,for training purpose we select 700 observations so 'n' is 700 here,if y_i i.e. actual Test output for ith observation in training set is equal to 1,we only get first part of cost function and other part is 0,now depending on y_i^α which is predicted value of Test variable depending on initialization and gradient,if y_i^α is not equal to 1 and is near 0 then value of $-log(y_i^\alpha)$ tends to ∞ which means we get a high penalty for wrong predicted value so on that basis cost function should be minimized to give exact values as actual values.

Similarly when y_i is zero the first term is 0 in Cost funtion L,the cost function is now equal to $-log(1 - y_i^\alpha)$ so when y_i^α is not equal to zero we get value of L near ∞ i.e. again a high penalty for wrong predicted output in such way entire cost function is minimized.



above graph clearly explains the behaviour of cost function as explained.

0.3 Gradient descent

Gradient descent is a minimization algorithm used to decrease the slope(gradient) of cost function based on a parameter called 'learning rate' and initialization of weight parameters,based on learning rate magnitude there is possibility of reaching local minima rather than global

minima and similar things apply to initialization of weights hence both should be optimum at start and multiple initializations are needed to check for actual global minima.

Mathematical interpretation and matrices

Mathematically gradient descent is as follows.

$$\theta_j = \theta_j - \nabla L(y_i^\alpha, \theta)$$

here θ is weight matrix,for current problem it's size is $(5 * 1)$.

here $\nabla L(y_i^\alpha, \theta)$ is gradient matrix that contains gradient of cost function with respect to each weight parameter as shown below.

$$\theta = \begin{Bmatrix} \theta_1 j \\ \theta_2 j \\ \theta_3 j \\ \theta_4 j \\ \theta_5 j \end{Bmatrix}, \nabla L(y_i^\alpha, \theta) = \begin{Bmatrix} \partial L/\partial\theta_1 j \\ \partial L/\partial\theta_2 j \\ \partial L/\partial\theta_3 j \\ \partial L/\partial\theta_4 j \\ \partial L/\partial\theta_5 j \end{Bmatrix}, \theta_0 = \begin{Bmatrix} \theta_0 j \\ \theta_0 j \\ \theta_0 j \\ \theta_0 j \\ \theta_0 j \end{Bmatrix}$$

Sigmoid function for y_i^α

As output of y_i^α should lie in between 0 and 1, as it's probability,the sigmoid is special function whose output lies between 0 and 1.

$$y_i^\alpha = \sigma(z^i) = \frac{1}{1 + \exp(-z^i)}$$

where z_i is equal to:

$$z_i = \theta_0 i + \theta_1 j * X_1^i + \theta_2 j * X_2^i + \theta_3 j * X_3^i + \theta_4 j * X_4^i + \theta_5 j * X_5^i$$

where X^i correspond to independant variables temperature,pressure etc. for that particular i^{th} observation.
Creating a matrix of all X^i for 700 observations.

$$X = \begin{Bmatrix} X_1^1 & X_2^1 & X_3^1 & X_4^1 & X_5^1 \\ X_1^2 & X_2^2 & X_3^2 & X_4^2 & X_5^2 \\ X_1^3 & X_2^3 & X_3^3 & X_4^3 & X_3^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_1^{700} & X_2^{700} & X_3^{700} & X_4^{700} & X_5^{700} \end{Bmatrix}$$

Now creating a Z matrix containing all z^i

$$Z = \begin{Bmatrix} X_1^1 & X_2^1 & X_3^1 & X_4^1 & X_5^1 \\ X_1^2 & X_2^2 & X_3^2 & X_4^2 & X_5^2 \\ X_1^3 & X_2^3 & X_3^3 & X_4^3 & X_3^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_1^{700} & X_2^{700} & X_3^{700} & X_4^{700} & X_5^{700} \end{Bmatrix} \begin{Bmatrix} \theta_1 j \\ \theta_2 j \\ \theta_3 j \\ \theta_4 j \\ \theta_5 j \end{Bmatrix} + \begin{Bmatrix} \theta_0 j \\ \theta_0 j \\ \theta_0 j \\ \theta_0 j \\ \theta_0 j \end{Bmatrix}$$

It should be noted that j subscript represents iteration number in gradient descent while i superscript represents i^{th} observation number.

Now creating matrix of all $\sigma(z^i)$ from Z matrix calculated above also further use this matrix to create error matrix which is difference of $\sigma_(z^i) - y_i$

$$H(X) = \begin{Bmatrix} \sigma_(z^1) \\ \sigma_(z^2) \\ \sigma_(z^3) \\ \vdots \\ \vdots \\ \sigma_(z^{700}) \end{Bmatrix}, error = \begin{Bmatrix} \sigma_(z^1) \\ \sigma_(z^2) \\ \sigma_(z^3) \\ \vdots \\ \vdots \\ \sigma_(z^{700}) \end{Bmatrix} - \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_{700} \end{Bmatrix}$$

It's proved that equation[1]:

$$\partial L/\partial \theta_1 j = \sum_{i=1}^{700} (\sigma(z^i) - y_i) * X_1^i$$

Similarly it applies to other $\frac{\partial L}{\partial \theta_j}$,finally it can be converted into matrix form and entire gradient matrix can be written as:

$$\nabla L(y_i^\alpha, \theta) = X^T * error$$

while

$$\partial L/\partial \theta_0 j = \sum_{i=1}^{700} (\sigma(z^i) - y_i)$$

Finally the Cost function can also be written as matrix equation.

$$L = -Y * \log H(x) - (1 - Y) * \log(1 - H(x))$$

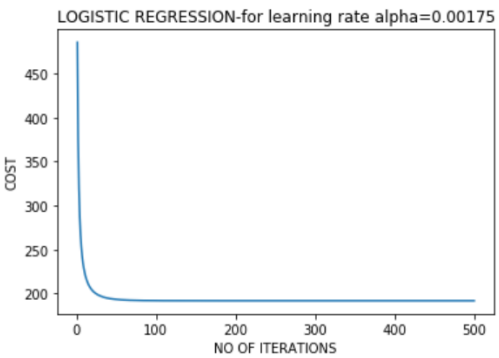
where Y is:

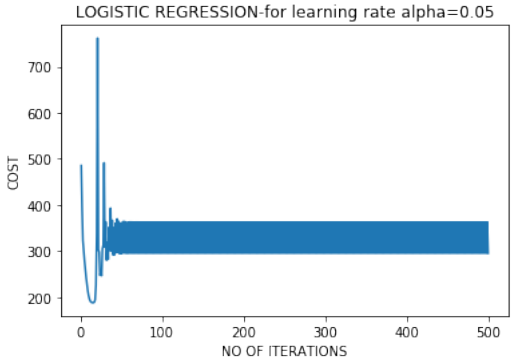
$$Y = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_{700} \end{Bmatrix}$$

0.4 Results and analysis

The learning rate and number of iterations as well as initializations play an important role in finding global minima.Following graphs show effect of smaller and higher learning rate on cost function and global minima while number of iterations are held constant.

Different initializations can also lead to different minimas,this problem is typical as different local minimas can be encountered based on the initialization.

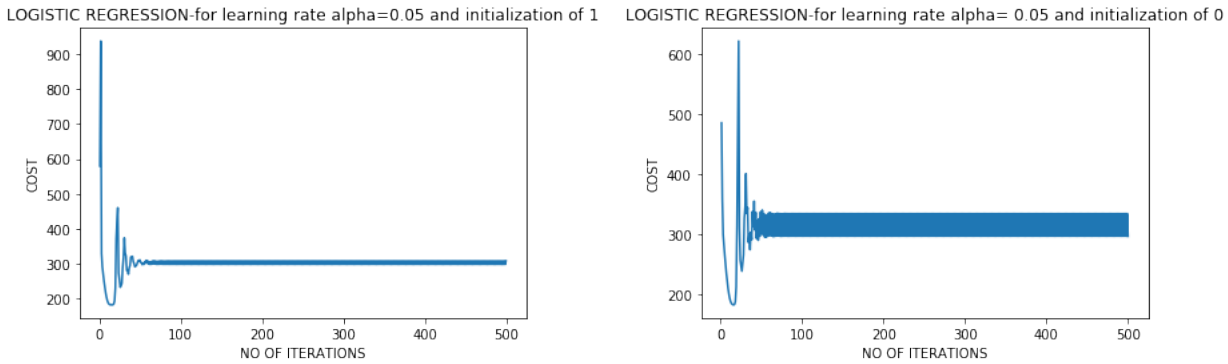




,

LEARNING RATE COMPARISON FOR SAME NO. OF ITERATIONS.

It can be seen that a lower learning rate provided a much lower cost function than relatively high learning rate.

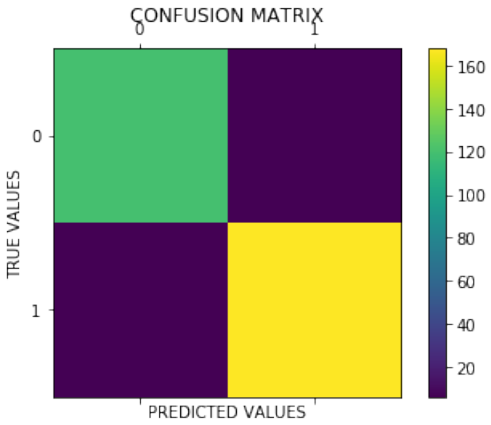


,

EFFECT OF INITIALIZATION ON COST FUNCTION MINIMA.

Confusion matrix and accuracy

Confusion matrix is measure of how much test data is correctly matched with predicted values and hence gives accuracy percentage,different samples result in different accuracy,but overall a good model will not diverge from a particular range of values,based on training sets an accuracy range of (92,98) percentage was achieved.



accuracy score: 96.0

report

	precision	recall	f1-score	support
0	0.95	0.95	0.95	126
1	0.97	0.97	0.97	174
accuracy			0.96	300
macro avg	0.96	0.96	0.96	300
weighted avg	0.96	0.96	0.96	300

,

HIGH PRECISION AND RECALL AT SAME TIME INDICATES GOOD MODEL FIT.

Optimum parameters

Based on optimum weights obtained through gradient descent,we can find independant variables that are important and those that aren't based on confidance intervals.

The equation obtained with 97 percent accuracy is as follow:

$$= 1.15 - 0.09 * X1 - 0.63 * X2 - 0.71 * X3 + 4.04 * X4 - 0.192 * X5$$

From equaton it can be seen that except X_4 i.e. 'coolant flow rate' other variable coefficients are negative and have less magnitude compared to X_4 i.e probability of passing the Test by reactor increases at higher 'coolant flow rate' other parameters just decrease the probability but not by large magnitude.

Significance levels for parameters

Using statsmodels package in python a chart with p-values was obtained that gave following significance levels for each coefficients.

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.1658	0.172	6.790	0.000	0.829	1.502
x1	-0.0972	0.140	-0.694	0.488	-0.372	0.177
x2	-0.6356	0.148	-4.296	0.000	-0.926	-0.346
x3	-0.7198	0.152	-4.737	0.000	-1.018	-0.422
x4	4.0688	0.327	12.459	0.000	3.429	4.709
x5	-0.1937	0.141	-1.371	0.170	-0.470	0.083

SIGNIFICANCE LEVEL CHART

0.5 Conclusion

From significance level chart it can be concluded that 'temperature'(X1) and 'inlet reactant concentration'(X5) have p-value greater than 0.05 which indicates that we can't reject null hypothesis for this coefficients at level of 95 percent confidence interval and they are insignificant for deciding Test to be Pass or Fail,rest all coefficients are significant for the deciding Test to be a Pass or Fail.

0.6 References

[1] Speech and language processing,Stanford university,Daniel jurafsky,James H.Martin,page no.[91-92],2019.