**Multiple Linear Regression Model and Its Variants as Solutions for Regression Problems in Machine Learning – Part II**

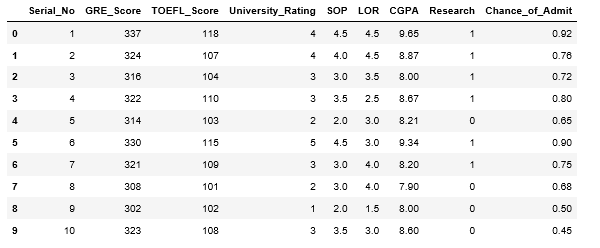
*By Veena S. Parvathi, Business Intelligence & Analytics Group*

**Round up of Part I**

In the first installment of this series, I introduced to you the concept of regression and the method of multiple linear regression for solving regression problems.

In this episode, you will be exposed to the method of Ridge regression, which has multiple linear regression at its foundation level, on which some amount of fine tuning of model parameters is performed for the betterment of the model.

This question is easier to answer through a demonstrative example than by a long description extending to multiple paragraphs. Take a look at the sample data shown below.



This is a screenshot taken from a python jupyter notebook view of the dataset obtained using **pandas** library. The data is arranged in rows and columns. Every column is one variable or data field and every row is one record. Throughout this blog, I will be adhering to terminologies aligned with machine learning (ML) as much as possible.

Here’s a question for you. Given a chance to pick a data field whose values are strongly related to values in one/more of the remaining fields, what will be your choice? Well, you got it right – “Chance of admit” and this is going to be the target data field, which is called as ***label.*** Now you have to pick one(more) field(s) from the remaining set, which you feel can be mapped to the label. The set of fields you selected are the ***features*.** Features are nothing but predictor variables for label. Mind you, the predictors should not be interrelated in any way. So the rule of thumb here is

* Predictors – Independent variables; each predictor is independent of the others.
* Target – Dependent variable

In a regression problem what you do is find a mathematical relationship between the features and the label. Once this relationship is established, then the value of the label can be predicted for any given set of values of features. Label will always be a continuous variable of numeric type. In this example I have considered all the feature values as of type numeric, but in general they can be of type ordinal/nominal/interval as well. A detailed discussion about data type of feature variables is not intended to be a part of this blog.

There are several open source ML algorithms available to handle regression problems, starting from multiple linear regression to random forest to complex neural networks. They all work in different ways. The fundamental difference is, whilst linear regression mines for any linear relationship between the label and the features, other algorithms look out for non-linear relationships as well.

My sole focus in this blog will be on multiple linear regression (and its variants in subsequent episodes of this blog) as it is the simplest and therefore easiest to understand; it is also a good starting point for someone who wants a jump start in ML.

**Multiple Linear Regression**

To start with, I am picking GRE Score, University Rating, SOP, and Research as features. There’s no concrete reason as to why I picked these fields, I just picked it. For this set of features, I can write the equation for a regression model as

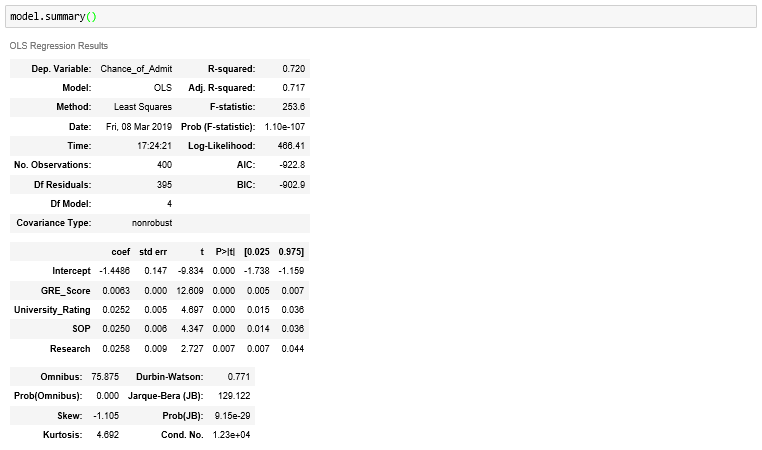
Chance\_of\_Admit = (a x GRE\_Score) + (b x University\_Rating) + (c x SOP) + (d x Research) + e

Where a, b, c, d and e are the model parameters for the regression model generated. Now the question is how will you get the values of a, b, c, d and e. You feed the data for both features and label into your ML algorithm and allow it to find the best values for the model parameters.

An example of how to perform this using python is shown below. To run this code you have to import the library **statmodels.formula.api**

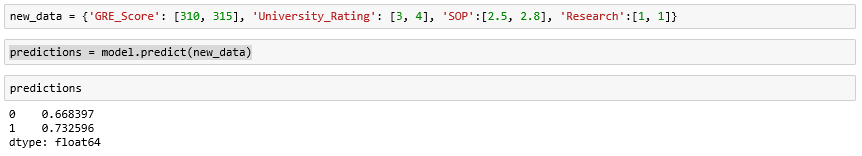


Note that the best values for the model parameters are obtained by running the algorithm by calling the function **ols()**. The same can be displayed along with several other statistics by calling the function **summary()**



The statistics shown in the above screenshot also contains information about the goodness of the model and relevance of each feature to the model.

Now that you have the model created, you can use it to get the predictions from the model for the label. To run the model with some new data, I have created a dataset new\_data as shown below. Predictions from the model are obtained using **predict()** function



**Remarks**

In this blog I have not talked about data types and how to handle data of type other than numeric. The blog also does not cover the topic of data preparation which can take you a long way in getting very good model performance. Details of the model statistics and improving the quality of the model through feature engineering are two more things that can go into the “Missing List”.

I omitted these topics intentionally for fear of deviating too much from what I really want to put in here. But stay tuned, these will be covered in upcoming blogs.

**Acknowledgements**

Dataset used for illustration purpose is obtained from Kaggle community. The link to the dataset is <https://www.kaggle.com/mohansacharya/graduate-admissions>