ELL409: Assignment 2

Arundhati Dixit   2016ME10824

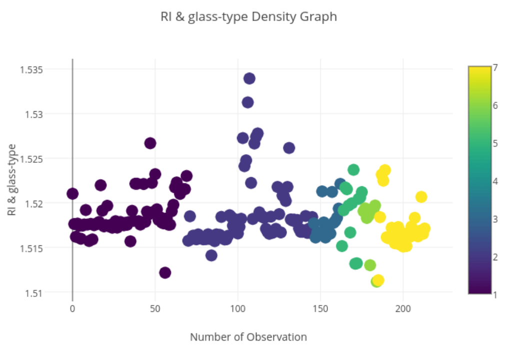
4 March 2020

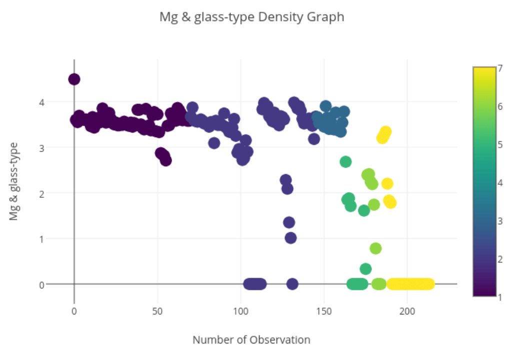
Multinomial Logistic Regression

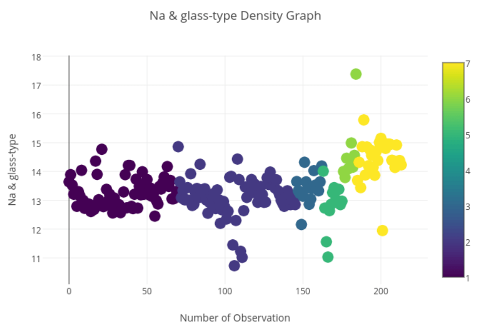
Dataset: https://archive.ics.uci.edu/ml/datasets/Glass+Identification (creator: B. German Central Research Establishment Home Office Forensic Science Service Aldermaston, Reading, Berkshire RG7 4PN)

Description: The dataset consists of 214 points described by 9 attributes. The attributes are RI: refractive index, Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10), Mg: Magnesium, Al: Aluminum, Si: Silicon, K: Potassium, Ca: Calcium, Ba: Barium, and Fe: Iron. Types of glass are 1- building_windows_float_processed, 2- building_windows_non_float_processed, 3- vehicle_windows_float_processed, 4- vehicle_windows_non_float_processed, 5- containers, 6- tableware, and 7- headlamps.

Code: The algorithm used is Multinomial Logistic Regression which is basically Maximum Likelihood Estimation, and is implemented in Python. In case of regression, the attributes are used to predict the value of dependent variable, and the obtained values are thus continuous depending on the value that attributes take. The logistic regression model, on the other hand, uses logistic function to model binary variable and thus class of a point can be predicted with the help of its attributes. Multinomial logistic regression generalises this to a multi class model. The Independence of Irrelevant Alternatives (IIA) assumption belies the model, which is a decent assumption for our dataset.







Plots like those shown above help us understand how each attribute works.

We use the function $softmax(z) = \frac{e^z}{\sum_{i=1}^{n} e^{z_i}}$, wherein the exponential makes all outputs positive, and then the output is normalised. The input is given by $z = xb$, and thus the multinomial logistic regression is characterised by $\hat{y} = softmax(xb)$. Batch sizes can be used, but since the dataset is of just 214 points, I have not used the same.

An accuracy of ~83% is obtained on the training set finally.

Robust Regression

Dataset: https://github.com/vincentarelbundock/Rdatasets/blob/master/csv/robustbase/hbk.csv (creator: Hawkins, Bradu, and Kass (1984), Location of several outliers in multiple regression data using elemental sets)

Description: http://vincentarelbundock.github.io/Rdatasets/datasets.csv has a list of datasets suitable for regression, although the size of datasets is typically small. The dataset called hbk and generated by Hawkins, Bradu, and Kass consists of 75 observations in 4 dimensions (1 response and 3 explanatory variables). It has a mix of outliers with the data created specifically for robust regression.

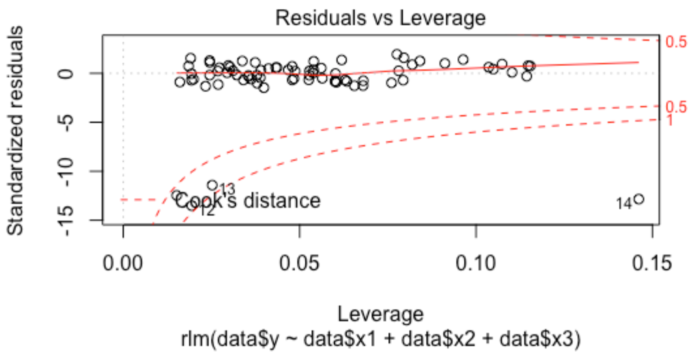
Code: The algorithm used is Robust Regression using Huber's criterion, and is implemented in Python. In case of OLS regression, the attributes are used to predict the value of dependent variable, and the obtained values are thus continuous depending on the value that attributes take. But the model formulated could be sensitive to presence of outliers or points with high leverage. To get rid of this and in order to get more conservative estimates, we can use robust regression and estimate coefficients using iteratively reweighted least squares. The normal equation now looks like:

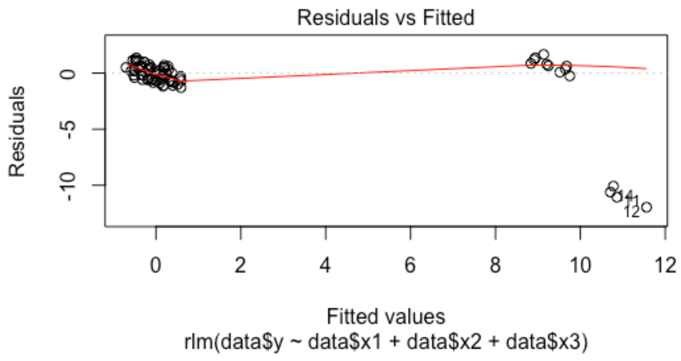
$$\sum_{i=1}^{n} w_i(y_i - x_i'b)x_i' = 0$$

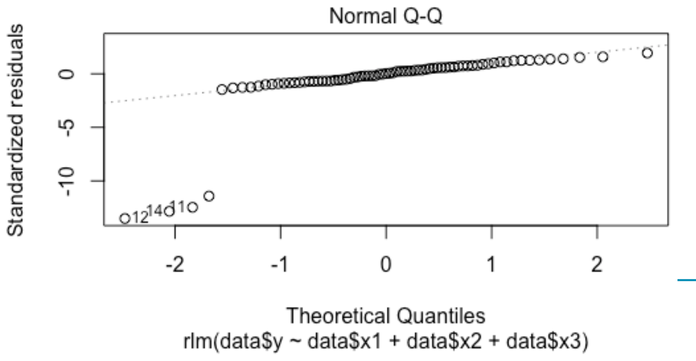
$$\text{and } w(e) = \begin{cases} 1 & \text{if } |e|<k \\ \frac{k}{|e|} & \text{if } |e|>k \end{cases}$$

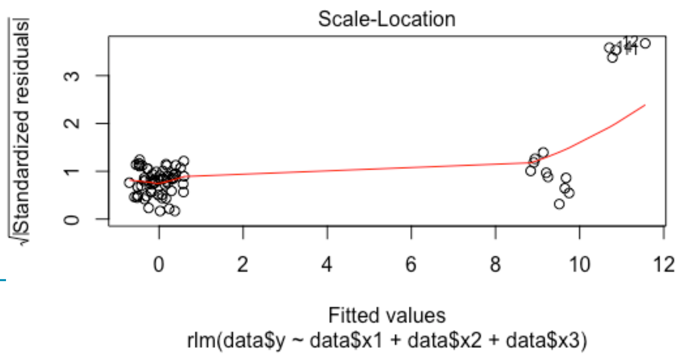
The error minimisation was $||e||^2 = e^T e$ in OLS, and now it is $||we||^2 = e^T w^T we$. The solution becomes $b = [X^T W^T W X]^{-1} X^T W^T W y$. The Newton Raphson update in each iteration in IRLS is $w^{(new)} = w^{(old)} - H^{-1} \nabla E(w)$.

We see that as the absolute residual goes down, the weight goes up. In other words, cases with a large residuals tend to be down-weighted. This weight is always equal to 1 in case of OLS estimates. We use k = 4/m, where m is the number of data points.







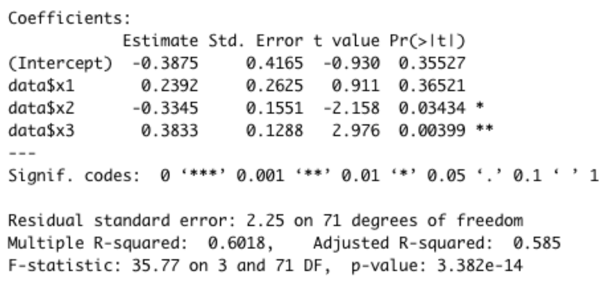


If we plot the residuals and leverage to check the presence of outliers (I have used R to do so, and to check my estimates), it is very evident that a few points are clear outliers and have high leverage, and hence robust regression is suitable.

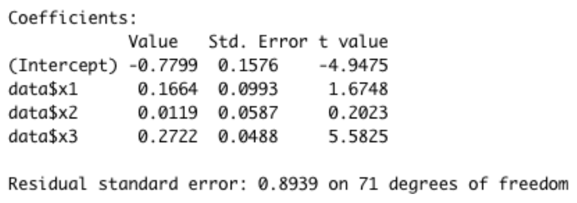
OLS estimates: -0.3875 (intercept)   0.2392 (x1)   -0.3345 (x2)   0.3833 (x3)

IRLS estimates: -0.7799 (intercept)   0.1664 (x1)   0.0119 (x2)   0.2722 (x3)

We see that the residual standard error decreases and the t-value of estimates are also mostly more conservative.



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.3875     0.4165  -0.930  0.35527
data$x1       0.2392     0.2625   0.911  0.36521
data$x2      -0.3345     0.1551  -2.158  0.03434 *
data$x3       0.3833     0.1288   2.976  0.00399 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.25 on 71 degrees of freedom
Multiple R-squared:  0.6018,    Adjusted R-squared:  0.585
F-statistic: 35.77 on 3 and 71 DF,  p-value: 3.382e-14
```



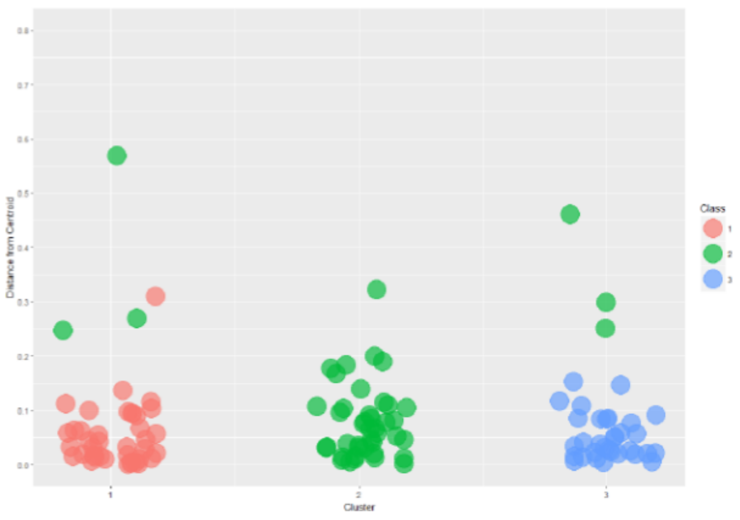
```
Coefficients:
            Value   Std. Error t value
(Intercept) -0.7799  0.1576    -4.9475
data$x1      0.1664  0.0993     1.6748
data$x2      0.0119  0.0587     0.2023
data$x3      0.2722  0.0488     5.5825

Residual standard error: 0.8939 on 71 degrees of freedom
```

Mini-Batch k means Clustering

Dataset: https://archive.ics.uci.edu/ml/datasets/wine (creator: Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.)

Description: The dataset is chemical analysis of wines grown inin Italy, from three different cultivators, and has 178 data points each described by its 13 attributes, that is, chemical compositional content. The attributes are Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines, and Proline. The first column contains cultivator index 1, 2, or 3.

Code: Mini-batch k-means works similarly to the k-means algorithm, just the difference is that in mini-batch k-means the update of centroids is done iteratively on a sample drawn from all observations instead of all observations. This reduces the time required for the algorithm to find convergence with imperative cost in quality. The larger the the size of the batch, the more computationally costly the training process and the better would be the quality.



This is what the clustering looks like for batch size = 15, #number of iterations = 10. For this algorithm, we assign centroids and the clusters are formed depending on the Euclidian distance between the cluster centroid and the point in question. The centroids are updated by gradient descent on batches of data.