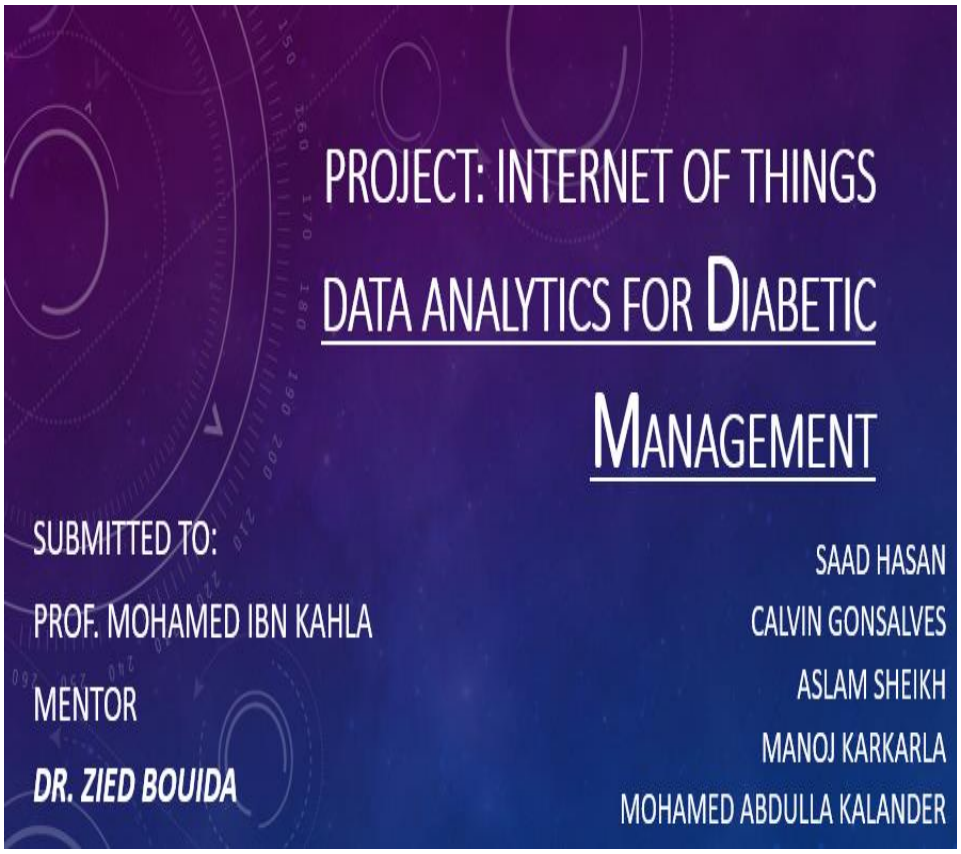


PROJECT: INTERNET OF THINGS

DATA ANALYTICS FOR DIABETIC MANAGEMENT

SUBMITTED TO:

PROF. MOHAMED IBN KAHLA

MENTOR

DR. ZIED BOUIDA

SAAD HASAN

CALVIN GONSALVES

ASLAM SHEIKH

MANOJ KARKARLA

MOHAMED ABDULLA KALANDER

DATA ANALYTICS FOR DIABETES MANAGEMENT

ADVANCED TOPICS IN COMPUTER COMMUNICATIONS: SYSC-5801

COURSE INSTRUCTOR: PROF. MOHAMED IBN KAHLA
PROJECT MENTOR: DR. ZIED

Data Analytics for Diabetes Management

S. Hasan, C. Gonsalves, A. Sheikh, M. Karkarla, M. A. Kalander

***Abstract*- Estimation of future glucose concentration in the blood is essential for the patient in terms of Diabetes management. However, individual patients may be unable to monitor their blood glucose level regularly because of all their daily work interference. The predicted glucose level can be used for detecting hypoglycemic or hyperglycemic conditions for adjusting the insulin injections or insulin intake of automated pumps. In this paper, a novel prediction system which uses a feed forward neural network and data obtained from the UCI machine learning repository, is proposed to predict the future blood glucose values based on the input attributes. The result of the proposed technique is evaluated and compared relative to that obtained from a Support Vector Regression (SVR) model. Our results indicate that, the proposed technique is better than SVR model for initial prediction values.**

I. INTRODUCTION

Diabetes is an almost household disease affecting millions of people in the world which results in most of the complexity of retinopathy, nephropathy, peripheral neuropathy, and blindness. Diabetes disease is mainly classified into Type 1 and Type 2. Whereas Type 1 occurs due to the inability of the β-cell of the pancreas to produce insulin (A hormone that regulates the metabolism of carbohydrate, fats and protein and absorption of glucose from blood). On the other hand, Type 2 mainly results due to a condition in which the endocrine system resist insulin. The functionality of exogenous insulin treatment for Type 1 diabetes is important to regulate Blood Glucose (BG) concentration. In the treatment of people with diabetes mellitus or hypoglycemia, monitor the glucose level is a most important task. To achieve such goal glucose meter is a significantly important device. A glucose meter is a medical device for determining the approximate concentration of glucose in the blood. Patient must put the drop of his/her blood onto a glucose strip that by injecting it into the glucose meter gives the glucose count. However, there has been a lot of research done to calculate glucose level more accurately and continuously. Advanced continuous glucose monitor(CGM) systems have become more popular. A disposable sensor placed under the skin that can also transmit and a reader that receives and displays the measurements are the main consists of a typical CGM system. However, as these techniques only measure the blood glucose level, there is not much warning or precaution for patient to know what the future values will be. Therefore, there has been an increasing research interest in the field of Diabetes Management to predict the Future Blood Glucose levels of atleast 2-3 hours in advance so as to keep the Diabetic patient well informed. Depending on these future predictions, the patient can control his diet as well as adjustments of the insulin pumps. The paper will be focused on the Type 1 Diabetes which is an Insulin Dependent Diabetes condition (IDD) and will also have an effect on the prediction results.

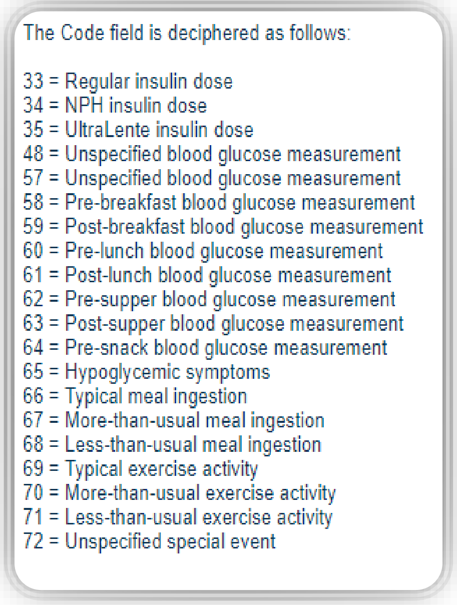
II. DATA PREPARATION

Data Preparation is the process of collecting, cleaning, and consolidating data into one file or data table, primarily for use in analysis. It is one of the most important and often time-consuming aspects of data mining. In fact, it is estimated that data preparation usually takes 50-70% of a project's time and effort. Devoting adequate energy to the earlier business understanding and data understanding phases can minimize this overhead, but you still need to expend a good amount of effort preparing and packaging the data for mining. Data preparation typically involves the following tasks:

• Merging data sets and/or records
• Selecting a sample subset of data
• Aggregating records
• Deriving new attributes
• Sorting the data for modeling
• Removing or replacing blank or missing values
• Splitting into training and test data sets

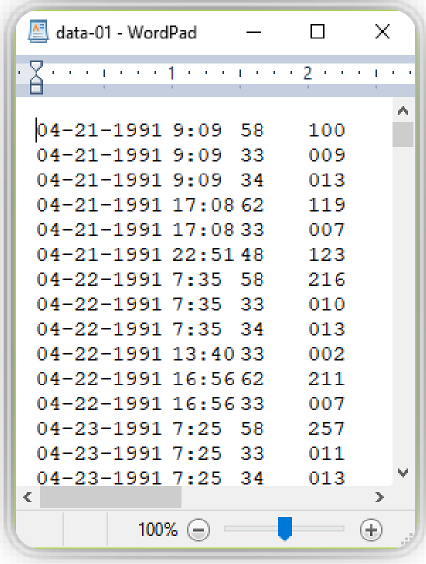
Data preparation process in our project:

The data set for our project is taken from UCI (university of California, Irvine) machine learning repository. There are 70 files each representing different patient's data. Each diabetes file contains of four fields per record. Each field is separated by tab and each record is separated by a newline. The four fields are Date in MM-DD-YYYY format, Time in XX: YY format, code and value. The code is deciphered as various insulin, glucose and exercise measurements as shown in the following Figure 1.



The Code field is deciphered as follows:

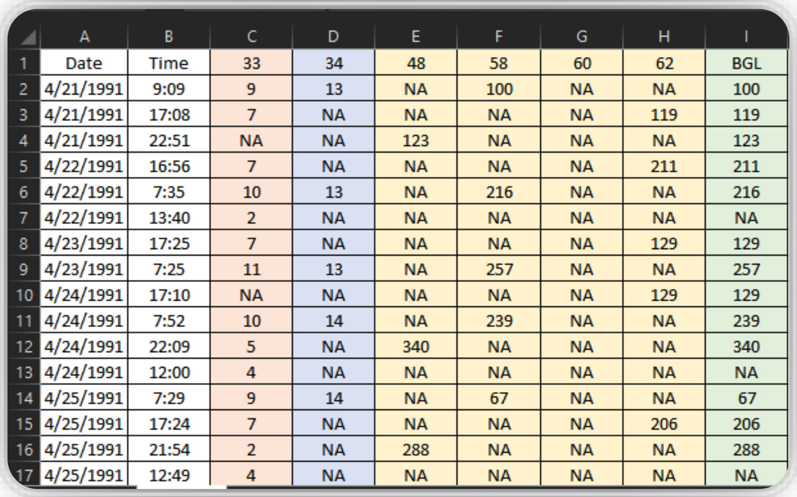
33 = Regular insulin dose
34 = NPH insulin dose
35 = UltraLente insulin dose
48 = Unspecified blood glucose measurement
57 = Unspecified blood glucose measurement
58 = Pre-breakfast blood glucose measurement
59 = Post-breakfast blood glucose measurement
60 = Pre-lunch blood glucose measurement
61 = Post-lunch blood glucose measurement
62 = Pre-supper blood glucose measurement
63 = Post-supper blood glucose measurement
64 = Pre-snack blood glucose measurement
65 = Hypoglycemic symptoms
66 = Typical meal ingestion
67 = More-than-usual meal ingestion
68 = Less-than-usual meal ingestion
69 = Typical exercise activity
70 = More-than-usual exercise activity
71 = Less-than-usual exercise activity
72 = Unspecified special event



```
04-21-1991 9:09  58    100
04-21-1991 9:09  33    009
04-21-1991 9:09  34    013
04-21-1991 17:08 62    119
04-21-1991 17:08 33    007
04-21-1991 22:51 48    123
04-22-1991 7:35  58    216
04-22-1991 7:35  33    010
04-22-1991 7:35  34    013
04-22-1991 13:40 33    002
04-22-1991 16:56 62    211
04-22-1991 16:56 33    007
04-23-1991 7:25  58    257
04-23-1991 7:25  33    011
04-23-1991 7:25  34    013
```

Figure 1. Raw Dataset acquired from the UCI online repository

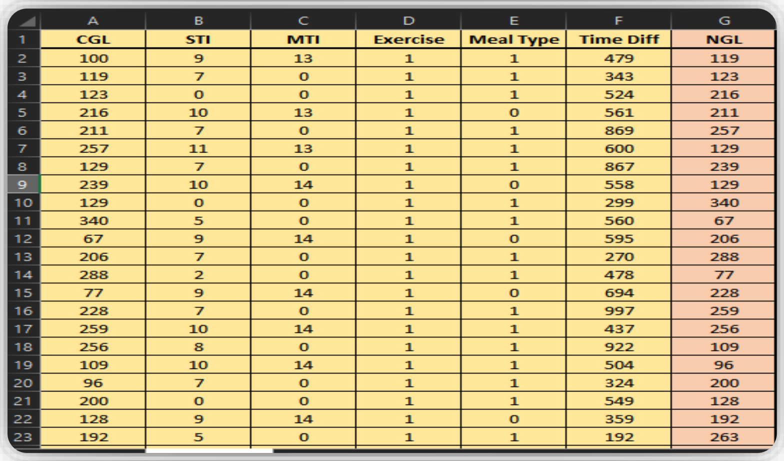
Initially, the above dataset is converted to a format which is needed for Regression algorithms but the results are not much satisfactory. This format is achieved by making the date, time and each measurement code as separate column. BGL column is created to store the glucose measurement made during that time of a day separately so that calculation can be made easier. The format is shown in the following sample Figure 2.



Date	Time	33	34	48	58	60	62	BGL
4/21/1991	9:09	9	13	NA	100	NA	NA	100
4/21/1991	17:08	7	NA	NA	NA	NA	119	119
4/21/1991	22:51	NA	NA	123	NA	NA	NA	123
4/22/1991	16:56	7	NA	NA	NA	NA	211	211
4/22/1991	7:35	10	13	NA	216	NA	NA	216
4/22/1991	13:40	2	NA	NA	NA	NA	NA	NA
4/23/1991	17:25	7	NA	NA	NA	NA	129	129
4/23/1991	7:25	11	13	NA	257	NA	NA	257
4/24/1991	17:10	NA	NA	NA	NA	NA	129	129
4/24/1991	7:52	10	14	NA	239	NA	NA	239
4/24/1991	22:09	5	NA	340	NA	NA	NA	340
4/24/1991	12:00	4	NA	NA	NA	NA	NA	NA
4/25/1991	7:29	9	14	NA	67	NA	NA	67
4/25/1991	17:24	7	NA	NA	NA	NA	206	206
4/25/1991	21:54	2	NA	288	NA	NA	NA	288
4/25/1991	12:49	4	NA	NA	NA	NA	NA	NA

Figure 2. Processed Dataset for Regression Algorithm

As the SVR Algorithm didn't give much acceptable results, we switched onto prediction using Neural Networks. Hence, we converted the data to the format which contains current glucose level (CGL), short term insulin (STI), mid-term insulin (MTI), exercise, meal type, time difference in minutes between two measurements and next glucose level (NGL). The exercise value is 1 if the person has done some physical exercise (1 means yes, 0 means no). The meal type represents whether the patient has a meal or not (1 means yes, 0 means no). NGL is nothing but next records current glucose level. The format is shown in the following sample data figure.



CGL	STI	MTI	Exercise	Meal Type	Time Diff	NGL
100	9	13	1	1	479	119
119	7	0	1	1	343	123
123	0	0	1	1	524	216
216	10	13	1	0	561	211
211	7	0	1	1	869	257
257	11	13	1	1	600	129
129	7	0	1	1	867	239
239	10	14	1	0	558	129
129	0	0	1	1	299	340
340	5	0	1	1	560	67
67	9	14	1	0	595	206
206	7	0	1	1	270	288
288	2	0	1	1	478	77
77	9	14	1	0	694	228
228	7	0	1	1	997	259
259	10	14	1	1	437	256
256	8	0	1	1	922	109
109	10	14	1	1	504	96
96	7	0	1	1	324	200
200	0	0	1	1	549	128
128	9	14	1	0	359	192
192	5	0	1	1	192	263

Figure 3. Processed Dataset for Neural Network Algorithm

We performed two important Data pre-processing steps:

1. Data Reduction - Removed the rows which had a 0 value for the next glucose level (NGL).
2. Data Normalization – Used the min-max normalization technique to normalize values to the 0-1 range

We have used PHP scripting language to code the conversion format. This automation helps us easily convert all the diabetes files to the above required format. We made 60% of the sample as training sample and remaining 40% as test sample. The PHP code for data preparation can be seen in the index of this report.

III. DATA VISUALIZATION

After looking at the rows and colums of a dataset to understand the parameters we are to operate on, the next step is to visualize the dataset to see the patterns in the data or the distribution fo various parameters across the dataset. In this project, we are using R programming language for the visualization and R has extensive tools to visualize the data. First, we have used hist() which part of base R and its default option yields a histogram based on the number of times a record falls into each of the bins on which the histogram is based.

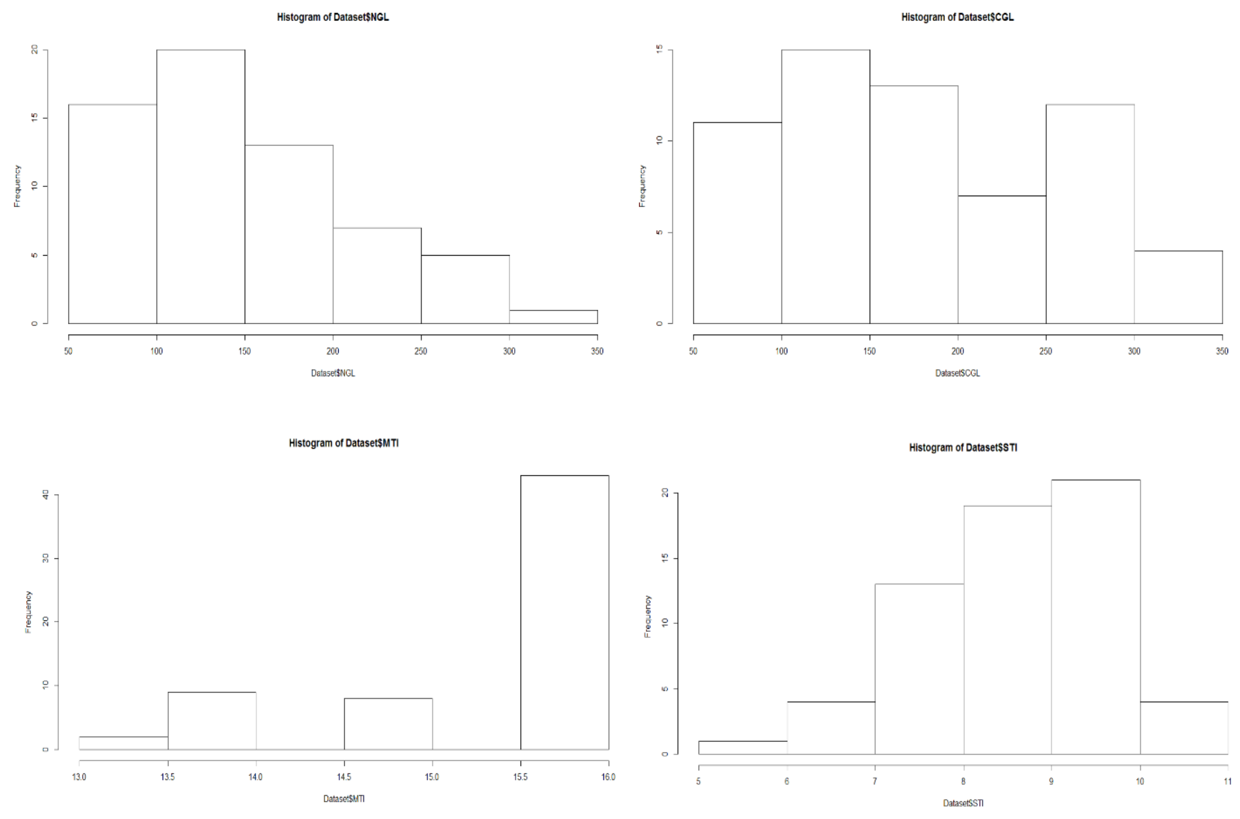


Figure 4. Distribution of various paramters across dataset of patient 1

These histograms give you the frequency of various parameters, so this will give you a basic idea of the distribution and see if the data is clustered along few principle points or uniformly or unequal distribution across the dataset for various parameters.
There are various visualization tools in R but, these visualizations are sufficient to know the dataset we are working on and we will now apply Logistic Regression.
Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable. The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.

Error Residuals vs fitted values: The dotted line at y=0 indicates our fit line. Any point on fit line obviously has zero residual. Points above have positive residuals and points below have negative residuals. The red line is the smoothed high order polynomial curve to give us an idea of pattern of residual movement. In our case we can see that our residuals have non-logarithmic pattern that means we got a better model.

Normal Q-Q Plot: The Normal Q-Q plot is used to check if our residuals follow Normal distribution or not. The residuals are normally distributed if the points follow the dotted line closely; In this case residual points follow the dotted line closely except for observation #349 So our model residuals have passed the test of Normality.

Scale-Location Plot: Scale location plot indicates spread of points across predicted values range One of the assumptions for Regression is Homoscedasticity i.e. variance should be reasonably equal across the predictor range. A horizontal red line is ideal and would indicate that residuals have uniform variance across the range. As residuals spread wider from each other the red spread line goes up; In our case the data is non-Homoscedastic i.e. our data is not uniformly not distributed along the line, Hence not predicting the right value always.
This doesn't mean our training is not accurate but need more variables and constraints to train our algorithm

Residuals vs Leverage Plot: Before attacking the plot, we must know what Influence and what leverage is. Let's understand them first. Influence: The Influence of an observation can be thought of in terms of how much the predicted scores would change if the observation is excluded. Cook's Distance is a pretty good measure of influence of an observation. Leverage: The leverage of an observation is based on how much the observation's value on the predictor variable differs from the mean of the predictor variable. The more the leverage of an observation, the greater potential that point has in terms of influence.

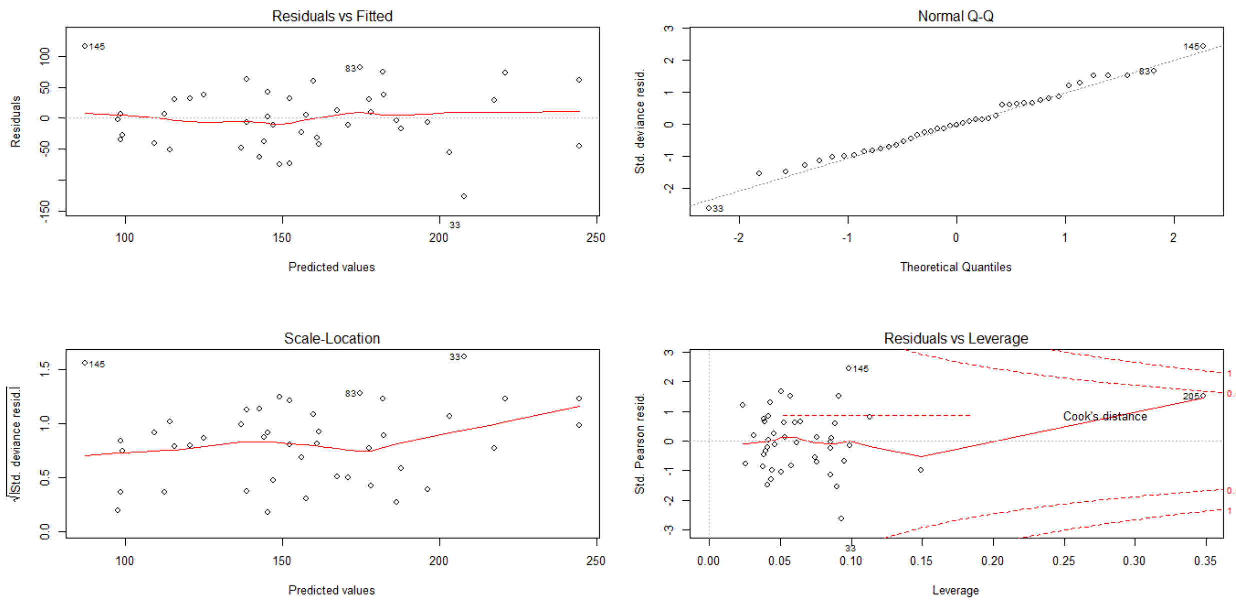


Figure 5. Logistic regression for visualizing and analyzing the datasets

In the plot below the dotted red lines are cook's distance and the areas of interest for us are the ones outside dotted line on top right corner or bottom right corner. If any point falls in that region, we say that the observation has high leverage or potential for influencing our model is higher if we exclude that point. It's not always the case though that all outliers will have high leverage or vice versa.

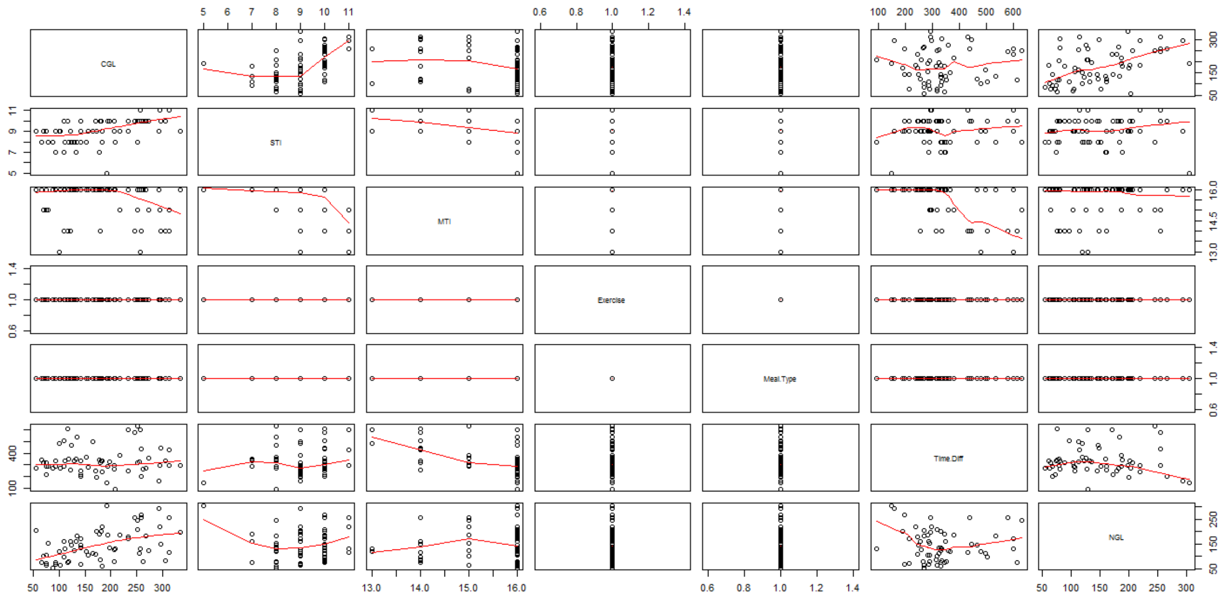


Figure 6. Cluster of Distributions depicting Cook's distance

IV. Data Prediction Using Artificial Neural Networks

Artificial neural networks (ANN) systems are intelligent computing systems that resemble the biological neural networks in human brains. An ANN consists the networks of connected units or nodes called artificial neurons [3]. In an ANN, a typical artificial neuron receives the signal, process it according to activation function used to program it to send to the next artificial neurons connected to it via a connection between two consecutive nodes.

One of the most popular ANN paradigms is the feed-forward neural network (FNN) and the associated back-propagation (BP) training algorithm. Feedforward Neural Networks are the type of artificial neural networks where the connections between do not form a cycle. Feedforward neural networks were the first type of artificial neural network invented and are simpler than their counterpart, recurrent neural networks. They are called feedforward because information only travels forward in the network (no loops), first through the input nodes, then through the hidden nodes (if present), and finally through the output nodes.

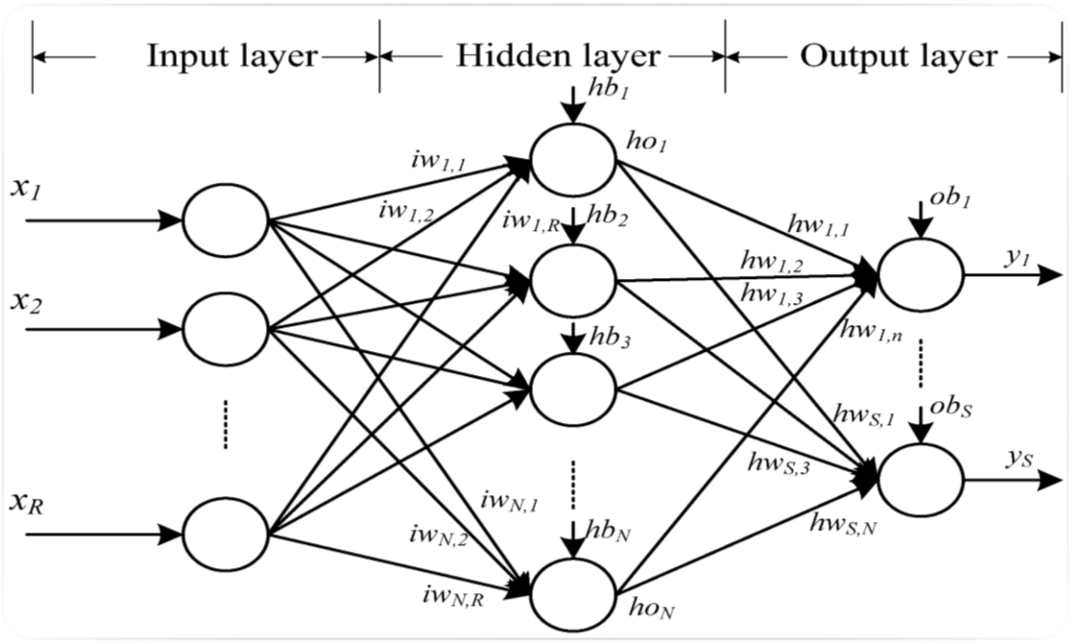


Figure 7. A feedforward Neural Network with information flowing left to right.

Feedforward neural networks are primarily used for supervised learning in cases where the data to be learned is neither sequential nor time-dependent. there are no feedback connections or loops in the network. It has an input layer, an output layer, and a hidden layer. In general, there can be multiple hidden layers. Each node in the layer is a Neuron, which can be thought of as the basic processing unit of a Neural Network.

In a Neural network, An Artificial Neuron (AN) or a perceptron is the basic unit that does all the decision taking the task in very small level. A schematic diagram of a neuron is given below. An AN takes inputs with their corresponding weights(w) and adds them. After summation it uses an activation function to normalize the sum. There are weights called bias(b) associated with each input of a neuron. These are the parameters which the network has to learn during the training phase.

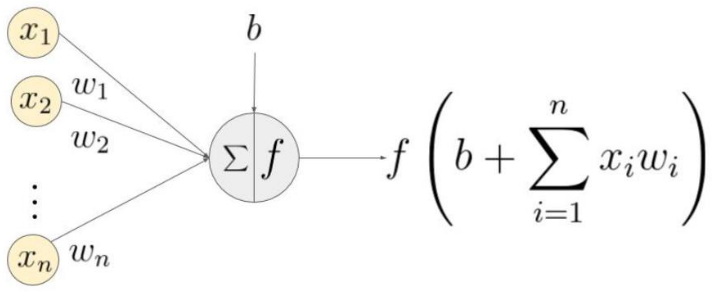


Figure 8. An example of a neuron showing the input ($x_1 – x_n$), their corresponding weight ($w_1 – w_n$), a bias (b) and the activation function *f* [2].

A. Activation Functions

In a neuron, all the decision about the output is being taken by the activation function. It also helps the neuron learns Linear or Non-linear decision boundaries [1]. Additionally, because of its cascading effect it limits the output values of neurons to become very large after several layers by normalizing the output of the neuron. There are three most widely used activation functions

1) Sigmoid
2) Tanh
3) Rectified Linear Unit(ReLU)

FNNs are usually organized into something called layers. Generally, a typical artificial neural network composed of an input layer, zero or few hidden layers, and an output layer. While there are no hidden layers in a single-layer perceptron. On the other hand, there is at least one hidden layer of multiple perceptions in an FNNs. Each layer importance in an FNN in described below

a) Input Layer

   This is the initial layer of a neural network used to supply the input data or features to the network.

b) Output Layer

   This is the final layer which gives out the predictions. For a regression problem, where the output is not a predefined category, we can simply use a linear unit.

c) Hidden layer

   A feedforward network applies a series of functions to the input. By having multiple hidden layers, we can compute complex functions by cascading simpler functions. The number of hidden layers is termed as the depth of the neural network.

B. Working of Artificial feed forward network

The input nodes simply pass on the input vectors x_i. The nodes in the hidden layer and output layer are processing units. Each processing node has an activation function which is commonly chosen to be the sigmoid function, where is a constant controlling the slope of the function. The net input to a processing unit j is given by

$$f(\mathrm{x}) = b + \sum \mathrm{xi}\ w_i \qquad (1)$$

where x_i's are the outputs from the previous layer, w_i is the weight (connection strength) of the link connecting unit i to unit j, and *b* is the bias, which determines the location of the sigmoid function on the x axis.

A feed-forward neural network works by training the network with known examples. A random sample $(x_p$ $y_p)$ is drawn from the training set and x_p is fed into the network through the input layer. The network computes an output vector op based on the hidden layer output. o_p is compared against the training target y_p. A performance criterion function is defined based on the difference between op and y_p. A commonly used criterion function is the sum of squared error (SSE) function

$$C(w, b) = \frac{1}{n} \sum_{i=1}^{n} ||y_i - f(x_i, w, b)||_2, \tag{2}$$

The error computed from the output layer is backpropagated through the network, and weights (w_i) are modified according to their contribution to the error function. where is called learning rate, which determines the step size of the weight updating.

Backpropagation is one of the most common ways to train an ANN. It is an example of a supervised training model, in which example answers are provided for the network to conform to, as opposed to unsupervised learning, in which a network learns to classify inputs into different categories without being trained explicitly. The basic idea is that we define an error function that tells us how much each of the output neurons differs from their intended value, then send this error signal backwards through the network so each weight has an idea of just how wrong they are. Then we use calculus to compute how best to individually adjust each weight to improve the output. By setting the initial weights randomly and then running this backpropagation algorithm several times, local minima can be discovered in the network's error.

V. Implementation

The prediction algorithm proposed in this paper uses datasets of individual patients only as each patient would have a different lifestyle from the rest and thus would also affect the prediction outcome. We used 5 attributes as inputs to the neural network i.e. Current Glucose Levels, Short Term Insulin, Mid- term Insulin, Meal Type and Time Difference between the measured glucose values. Fig. # shows the architecture of the ANN predictor having an input layer of 5 nodes, 2 hidden layers of 4 and 2 nodes respectively and the output layer with a single node. The output of the ANN is next glucose level or predicted glucose level. After much trial and testing, the preferred properties of the ANN were set as follows: learning rate set to 0.001, activation function of the hidden layers was Hyperbolic tangent function and 3 number of iterations were enough to get the appropriate results. As the number of samples varied from patient to patient, the partition of the training dataset and test dataset was taken to be 80 -85% and 20 – 15 % respectively. For example, a dataset consisting of 400 samples, 320 samples were used as training set to train the Neural Network model. After this, the rest 80 samples were tested using this model to get the predictions and check the correctness. Also, normalization was also done in the data pre-processing as it helps the ANN to learn better

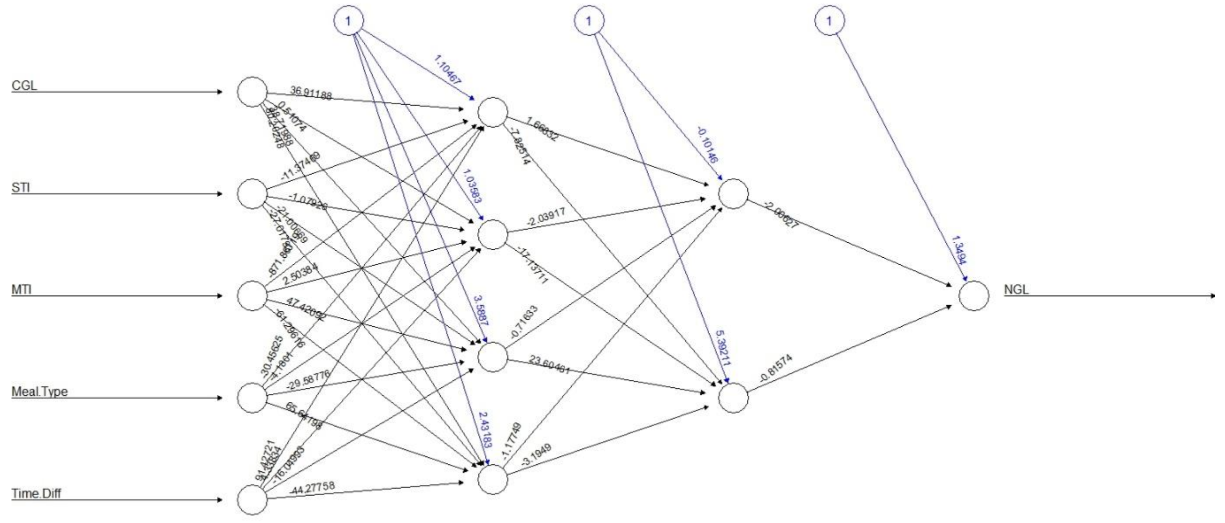


Figure 9. Trained Neural Network Architecture

The Figure 9. shows the architecture of the trained Neural Network model together with the weights of each link as well as the biases for the hidden layers and the output layer. The output of the NN architecture is NGL which is the next or predicted glucose level.

VI. RESULTS & COMPARISON

Many network architectures for the ANN are tested to optimize the prediction output. An architecture of 5-4-2-1 was found to give the best results. Accuracy of the Predicted results was measured by using the evaluation metric MSE given as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2 \qquad (3)$$

Where, N is the number of predictions, $\hat{x}$ is the predicted output and x is the actual blood glucose value.

The Table 1. illustrates the prediction comparison between the proposed method with Neural Networks and the SVR Regression Model. It can be seen from the table that there is a significant decrease in the MSE value using our proposed method.

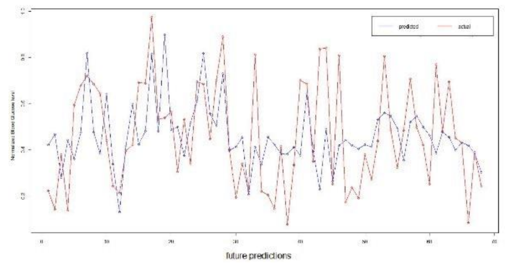
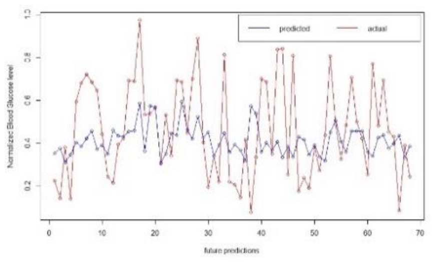
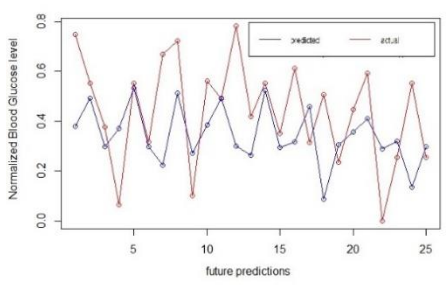
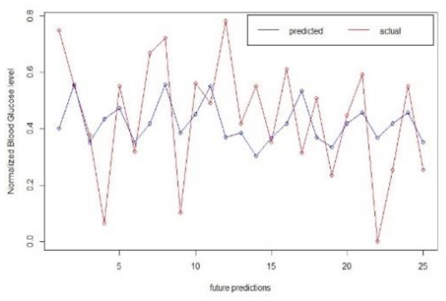
Table 1. MSE comparison between proposed method and SVR algorithm

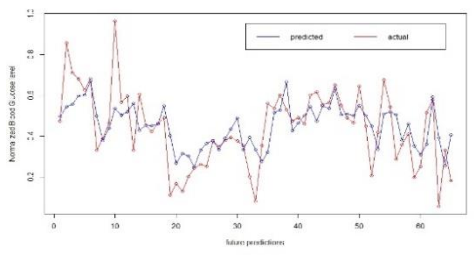
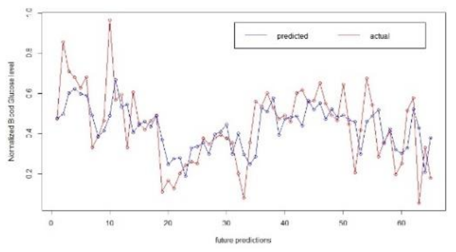
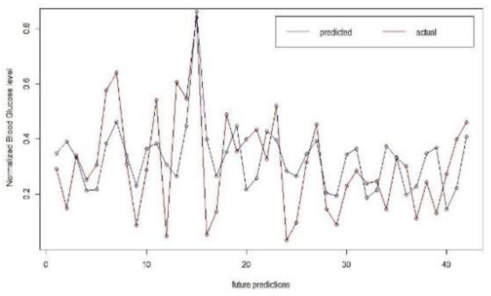
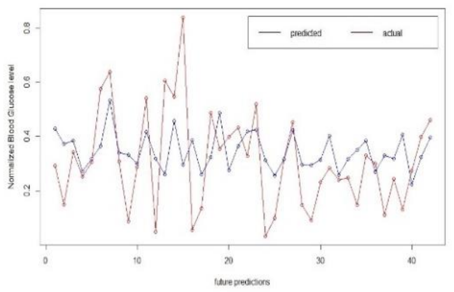
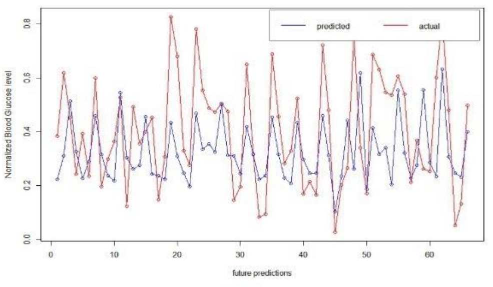
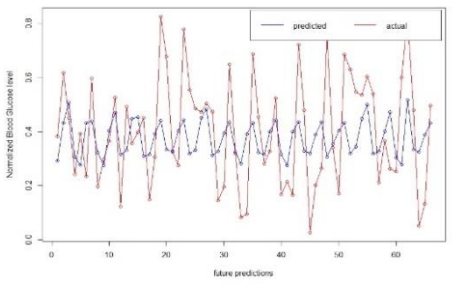
Patient Number	Proposed Neural Network algorithm	SVR Algorithm	% Decrease in MSE
1	0.027	0.035	22.85%
7	0.034	0.036	2.77%
29	0.0119	0.024	50.41%
41	0.029	0.031	6.45%
67	0.020	0.023	13.04%

Each graph in Table 2. is a Normalized Blood Glucose level vs Future Prediction of different patients taken at random with varying sample sizes. The actual values are denoted by a red line whereas the predicted values of both methods is depicted by a blue line.

As displaying the plots for all 70 patients would be tedious, only a few number of patient's data at random were chosen. The patient was chosen in such a way as having different sample sizes ranging from 100 samples to around 600 samples.

Table 2. Plot Comparison between Proposed algorithm and SVR algorithm

Patient	Feed Forward Neural Network (proposed)	SVR Regression Algorithm
1		
7		

29		
41		
67		

VII. DISCUSSION

From the above two tables it can be seen that the proposed Neural Network model performs better as compared to the SVR algorithm having large decrease in error for some patients. The proposed NN model gives a more accurate predicted output than the SVR because the relationship between the input attributes is highly dynamic. This Dynamicity cannot be perceived by a SVR Regression.

VII. Conclusion

Training the Feed Forward neural network using the data obtained from several different patients throughout the day, gives our model some generality. Also, our proposed method requires some fine adjustments and tuning when used with a specific patient. Our model can scale by including more input attributes such as lifestyle of the patient, carbohydrate consumption per meal, number of steps taken between blood glucose measurements, etc. It can also be seen that our model predicts well for the initial 45 – 50% of the future predictions and there after its performance deteriorates. Improvements to the prediction can be made by further adjusting the network parameters or by modifying the dataset structure in such a way as to make it easier for the Neural Network to learn. We conclude that, the proposed Feed Forward Neural Network succeeds to predict the future glucose values. It can be used for online blood glucose prediction in diabetes management systems.

References

[1] [online] Available: https://brilliant.org/wiki/feedforward-neural-networks

[2] [online] Available: https://www.learnopencv.com/understanding-feedforward-neural-networks

[3] Deep Learning Book" by Goodfellow, Bengio, and Courville. [online] Available: http://neuralnetworksanddeeplearning.com/

[4] Kevin Plis, Razvan Bunescu, Cindy Marling, Jay Shubrook, Frank Schwartz, "A machine learning approach to predicting blood glucose levels for diabetes management", in *Modern Artificial Intelligence for Health Analytics*. Papers from the AAAI-14, 2014.

[5] Juan Lui, Chandima Fernando, "Smartphone-based personalized blood glucose prediction", ICT Express 2, (2016), pp. 150-154.

[6] J Jianchao Han, Juan C. Rodriguez, Mohsen Beheshti, "Diabetes data analysis and prediction model discovery using rapidminer", in *2008 Second International Conference on Future Generation Communication and Networking,* vol. 3, IEEE, 2008, pp. 96–99.

[7] Evan M. Benjamin, Self-monitoring of blood glucose: the basics, Clin. Diabetes 20 (1) (2002) 45–47.

[8] Lidong wang, Cheryl Ann Alexander, "Big Data Analytics as Appied to Diabetes Management", in *European Journal of Clinical and Biomedical Sciences*, vol. 2, no. 5, 2016, pp. 29-33.

[9] T. Van Gestel B. Baesens G. Loterman et al. "Benchmarking Regression Algorithms for Loss Given Default Modeling" in *Credit Risk Management: Basic Concepts: Financial Risk Components Rating Analysis Models Economic and Regulatory Capital Oxford University Press,* vol. 28 no. 1 pp. 161-170 2009.

[10] V. Sujatha and V. Dharanidharan, "Predictive Modelling and Analytics on Big Data for Diabetic Management," *2017 World Congress on Computing and Communication Technologies (WCCCT)*, Tiruchirappalli, 2017, pp. 80-83.

[11] Viceconti, M., Hunter, P., & Hose R. Big Data, big knowledge: Big Data for personalized healthcare. *IEEE J Biomed Health Inform*, 2015, 19(4), 1209-1215.

APPENDIX

A. Code in PHP for Data Cleaning and preparation

```
Data Cleaning Code in PHP
<?php

$options = array(
               'length' => 0,
               'delimiter' => ',',
               'enclosure' => '"',
               'escape' => '\\',
               'headers' => true
obj <- s3HTTP(
    verb = "GET",
    bucket = bucket,
    headers = headers,
    path = "Nn_dataset.csv",
    key = Sys.getenv("AWS_ACCESS_KEY_ID"),
    secret = Sys.getenv("AWS_SECRET_ACCESS_KEY"),
    check_region = FALSE,
    base_url = url)

df.data.2 <- read.csv(text = rawToChar(obj$content))
head(df.data.2)
               );

$files = scandir("/home/mysystem/Downloads/Diabetes-Data");
foreach($files as $file) {

$data = file_get_contents('/home/mysystem/Downloads/Diabetes-Data/'.$file)
;


if ($file = @fopen("/home/mysystem/Desktop/outputfiles/".$file.".csv", 'w'
)) {

        $headers = frameheader();
        $data = combinedate($data);
        $data = manipulateData($data);
        $rows = framerows($data);
        if ($options['headers']) {
                // write the 1st row as headings
                fputcsv($file, $headers, $options['delimiter'], $options['e
nclosure']);
        }
        // Row counter
        foreach ($rows as $row) {
                fputcsv($file, $row, $options['delimiter'], $options['enclo
sure']);
        }

        // close the file
        fclose($file);
```

```
}
}
function frameheader() {

        $headers = array('CGL','NGL','STI','MTI','Exercise','Meal Type','Ti
me Diff');
        return $headers;

}

function combinedate($data) {

        $data1 = explode(PHP_EOL,$data);
library("aws.s3")

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It
includes your credentials.
# You might want to remove those credentials before you share your noteboo
k.
Sys.setenv("AWS_ACCESS_KEY_ID" = "4ada20a663ad4e0299be76e6ad5e2efa", "AWS_
SECRET_ACCESS_KEY" = "066232d2316e0bf9cc9bc231879e5ee7f3a7c8b1bb2431c6")
url <- "s3-api.us-geo.objectstorage.service.networklayer.com"
bucket <- "dataanalyticsfordiabetes709baa0f4f944f28a16717ca2694f645"
headers <- list(`x-amz-content-sha256`="e3b0c44298fc1c149afbf4c8996fb92427
ae41e4649b934ca495991b7852b855")

obj <- s3HTTP(
    verb = "GET",
    bucket = bucket,
    headers = headers,
    path = "Nn_dataset.csv",
    key = Sys.getenv("AWS_ACCESS_KEY_ID"),
    secret = Sys.getenv("AWS_SECRET_ACCESS_KEY"),
    check_region = FALSE,
    base_url = url)

df.data.1 <- read.csv(text = rawToChar(obj$content))
head(df.data.1)
        $data1 = array_filter($data1);
        $tmp = array();
        for($i=0;$i<=count($data1)-1;$i++) {
                $count = 0;
                $parts = preg_split('/\s+/', $data1[$i]);
                if(!empty($tmp)) {
                        foreach($tmp as $key => $val) {
                                if($parts[0].' '.$parts[1]==$val[0]) {
                                        $tmp[$key][]= $parts[2].'-'.$parts[3]
;
                                } else {
                                        $count++;
                                }
                        }
                        if($count==count($tmp)) {
                              $tmp[] = array($parts[0].' '.$parts[1],$parts
[2].'-'.$parts[3]);
                       }
               } else {
                       $tmp[] = array($parts[0].' '.$parts[1],$parts[2].'-'
.$parts[3]);
               }
       }
       //echo print_r($tmp,true);
       return $tmp;
}

function getPrevKey($key, $hash = array())
{
    $keys = array_keys($hash);
    $found_index = array_search($key, $keys);
    if ($found_index === false || $found_index === 0)
        return false;
    return $keys[$found_index-1];
}


function manipulateData($data) {

       $glucose = array('48','57','58','59','60','61','62','63','64');
       foreach($data as $dkey => $datum) {
               $glcount = 0;
               $stcount = 0;
               $mtcount = 0;
               $iniCount = count($data[$dkey]);
               foreach($datum as $key => $info) {
                       if($key!=0) {
                               $codeval = explode('-',$info);
                               if(in_array($codeval[0],$glucose)) {
                                       $data[$dkey]['cgl']=$codeval[1];
                                       $glcount++;
                               } else if($codeval[0]=='33') {
                                       $data[$dkey]['sti'] = $codeval[1];
                                       $stcount--;
                               } else if($codeval[0]=='34') {
                                       $data[$dkey]['mti'] = $codeval[1];
                                       $mtcount--;
                               }
                               $mtcount++;
                               $stcount++;
                       } else {
                               $mtcount++;
                               $stcount++;
                       }
               }
               if($stcount==$iniCount) {
                       $data[$dkey]['sti'] = 0;
               }
               if($mtcount==$iniCount) {
```

```
                        $data[$dkey]['mti'] = 0;
                }
                if($glcount==0) {
                        $prevKey = getPrevKey($dkey, $data);
                        if($dkey!=0) {
                                $data[$prevKey]['meal'] = 0;
                        }
                        unset($data[$dkey]);
                } else {
                        $data[$dkey]['meal'] = 1;
                }
        }

        $data = array_values($data);
        //echo print_r($data,true);
        return $data;

}


function framerows($data) {
        foreach($data as $dkey => $datum) {
                echo print_r($datum,true);
                $temp[0] = $datum['cgl'];
                if($dkey==(count($data)-1)) {
                          $temp[1] = 0;
                  } else {
                          $temp[1]=$data[$dkey+1]['cgl'];
                  }
                $temp[2] = $datum['sti'];
                $temp[3] = $datum['mti'];
                $temp[4] = 1;
                $temp[5] = $datum['meal'];
                if($dkey!=0) {
                        $date = explode('-',$data[$dkey-1][0]);
                        $date1new = $date[1].'-'.$date[0].'-'.$date[2];
                        $date = explode('-',$datum[0]);
                        $date2new = $date[1].'-'.$date[0].'-'.$date[2];
                        $from = date_create($date1new);
                        $to = date_create($date2new);
                        $diffObj = date_diff($to, $from);
                        $diffHours = $diffObj->format("%h");
                        $diffmin = $diffObj->format("%i");
                        $temp[6] = ($diffHours*60)+$diffmin;
                } else {
                        $temp[6] = 0;
                }
                $rows[]=$temp;
                unset($temp);
        }
        return $rows;

}
```

B. CODE USED FOR DATA VISUALIZATION IN R

```
Data Visualization in R

#Reading in the dataset and converting all missing values in the form of "
0" to NA's to remove them later
Data <- read.csv2("Nn_dataset.csv", header = TRUE, sep = ",", na.strings =
"0")
head(Data)
View(Data)
summary(Data)

#To know the no of NA's in each column
sapply(Data, function(x) sum(is.na(x)))

#Removing all the NA's
Dataset <- Data[complete.cases(Data),]
head(Dataset)
View(Dataset)
summary(Dataset)

#Some visualizations to get the feel of the dataset
hist(Dataset$CGL)
hist(Dataset$NGL)
hist(Dataset$STI)
hist(Dataset$MTI)


#Matrix of scatterplots
pairs(Dataset, panel = panel.smooth)

#Logistic regression model
#Preparing the DataSet
library("stats")
set.seed(123)
n <- nrow(Dataset)
train <- sample(n, trunc(0.70*n))
Dataset_Training <- Dataset[train, ]
Dataset_Testing <- Dataset[-train, ]


# Training The Model
glm_1 <- glm(NGL ~., data = Dataset_Training)
summary(glm_1)

#Removing non significant ones from the analysis above
glm_2 <- update(glm_1, ~. - MTI - Exercise - Meal.Type - Time.Diff)
summary(glm_fm2)

par(mfrow = c(2,2))
plot(glm_2)
```

C. CODE USED FOR IMPLEMENTING ALGORITHMS IN R

```
#Implementing Algortihm and checking the accuracy of the developed algorti
hm
#Simpe linear regression
#install.packages('neuralnet')
set.seed(123)
dataset = read.csv('Nn_dataset.csv')
library("caTools")
DataFrame <- dataset


str(DataFrame)

hist(DataFrame$NGL)

head(DataFrame,2)

apply(DataFrame,2,range)
#scale
maxValue <- apply(DataFrame, 2, max)
minValue <- apply(DataFrame, 2, min)

DataFrame <- as.data.frame(scale(DataFrame,center = minValue,scale = maxVa
lue-minValue))
# Making dummy variables
#dataset$level2 = dataset$level^2
#dataset$level2 = dataset$level^3
#dataset$level2 = dataset$level^4
#dataset$level2 = dataset$level^5
#dataset$level2 = dataset$level^6
#dataset$level2 = dataset$level^7
#dataset$level2 = dataset$level^8
#train

#ind <- sample(1:nrow(DataFrame),300)
trainDF <- DataFrame[1:300,]
#trainDF$Exercise <- NULL
#trainDF$Meal.Type <- NULL
#trainDF$MTI <- NULL


testDF <- DataFrame[301:368,]
#testDF$Exercise <- NULL
#testDF$Meal.Type <- NULL
#testDF$MTI <- NULL

regressor = lm(formula = NGL ~ Time.Diff,
               data = trainDF)
y_pred = predict(regressor, newdata = testDF)
# Visulaizing the test set results
library(ggplot2)
ggplot() +
```

```r
  geom_point(aes(x= testDF$NGL, y = testDF$Time.Diff),
             color = 'black') +
  geom_line(aes(x= trainDF$NGL, y = predict(regressor, newdata = trainDF))
,
            color = 'blue') +
  ggtitle("NGL vs timme diff") +
  xlab("NGL") +
  ylab("time diffrence")
# We applied Multiple linear regression but it failed
# Due to the fact that our data is not in sequence there are so much fluct
uatuion in our data set
# We can't use Recurrent neural network because it is good when it has lar
ger data consist of more than 10 gb of data because
# in ths case we need a large memory to train neural network
#
# New Algortihm
#implementing Neural Network
library(MASS)
library(neuralnet)
#install.packages('neuralnet')
set.seed(123)
dataset = read.csv('Nn_dataset.csv')

DataFrame <- dataset


str(DataFrame)

hist(DataFrame$NGL)

head(DataFrame,2)

apply(DataFrame,2,range)
#scale
maxValue <- apply(DataFrame, 2, max)
minValue <- apply(DataFrame, 2, min)

DataFrame <- as.data.frame(scale(DataFrame,center = minValue,scale = maxVa
lue-minValue))

##train

#ind <- sample(1:nrow(DataFrame),300)
trainDF <- DataFrame[1:300,]
trainDF$Exercise <- NULL
#trainDF$Meal.Type <- NULL
#trainDF$MTI <- NULL


testDF <- DataFrame[301:368,]
testDF$Exercise <- NULL
#testDF$Meal.Type <- NULL
#testDF$MTI <- NULL
```

```
#

allVars <- colnames(trainDF)
predictorVars <- allVars[!allVars%in%"NGL"]
predictorVars <- paste(predictorVars,collapse = "+")
form=as.formula(paste("NGL~",predictorVars,collapse = "+"))
form
neuralModel1<-neuralnet(formula = form,hidden = c(4,2),learningrate = 0.1,
linear.output = T,
                              data = trainDF)
#plot
plot(neuralModel1)


predictions <- compute(neuralModel1,testDF[,1:5])
str(predictions)

predictions <- predictions$net.result*(max(testDF$NGL)-min(testDF$NGL))+mi
n(testDF$NGL)
actualValues <-  (testDF$NGL)*(max(testDF$NGL)-min(testDF$NGL))+min(testDF$
NGL)


MSE <- sum((predictions - actualValues)^2)/nrow(testDF)
MSE

#plot(testDF$NGL, predictions, col='blue', main='Real vs Predicted',
     #lines(pch=1,cex=0.9,type="p",xlab ="Actual",ylab = "Predicted")
MSE <- sum((predictions - actualValues)^2)/nrow(testDF)
MSE

#plot(testDF$NGL, predictions, col='blue', main='Real vs Predicted', pch=1
,cex=0.9,type="p",xlab ="Actual",ylab = "Predicted")
#abline(0,1,col="black")
prediction <- predictions[1:68]
t <- testDF$NGL
plot(t, type ="o", col ="red", xlab="no.1", ylab="actual(red)   predicted(
blue)")
lines(prediction, type = "o", col="blue")
```