Problem 2: IMT Classification using Neural Network

Analyzed by: Sahana Mukherjee

UBIT Name: mukherj3

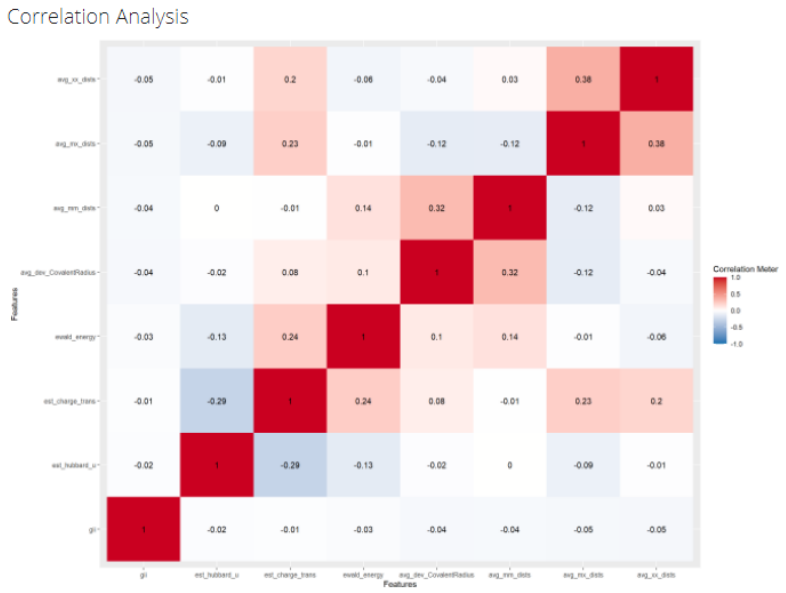
Environment used: RStudio

Packages used: DataExplorer, neuralnet

Exploratory Data Analysis:

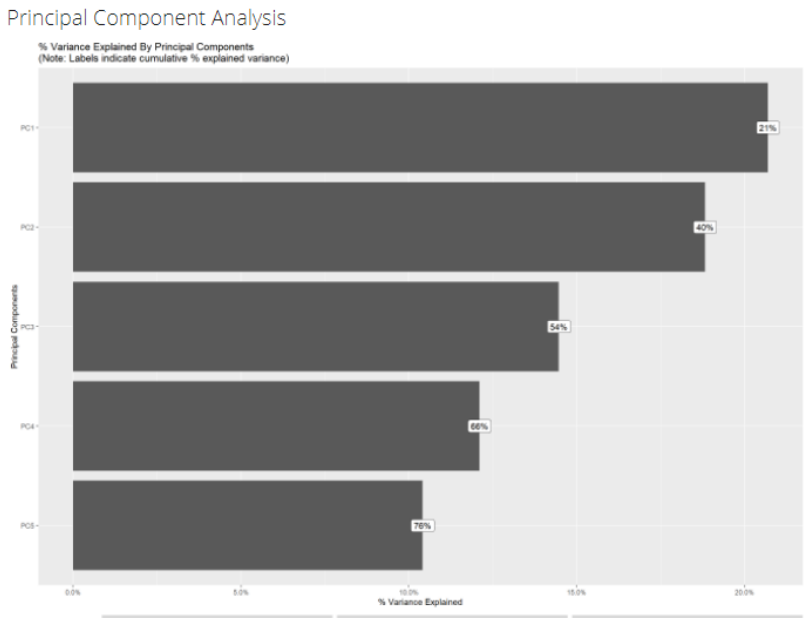
The concerned data set deals with three types of materials, Metals, Insulators and Transitions. They are respectively labelled as 0, 1 and 2. This classification is done on the basis of several geometric factors like covalent radius, atomic distances, and electrostatic factors like Hubbard potential, Ewald energy, charge and others. The consolidated dataset consists of 224 records and 13 variables, of which few are just metadata, and not contributing towards the analysis of the data. Hence, the data is being cleaned an analysis is done on reduced data set which consists of 9 variables, of which one is the label of the material, indicating the target variable. The target variable is converted to factors with 3 levels, 0, 1 and 2.

The correlation analysis is represented using heatmap below:



The variables are not highly correlated among themselves, so we are not proceeding towards dimension reduction, also as number of dimensions is fairly small, we do not head towards principal component analysis.

Just for reference, PCA has been performed to observe how much variance is explained by the principal components and below is the observation obtained.



It is found that about 76% of the variance is explained by the first 5 PCs. However, as we have only 9 variables to deal with, we do not proceed towards training our model on principal components, rather we do it on the complete data set, to retain all information.

Scaling the data

The dataset contains entries which belong to enormously different scales. To avoid higher value ranges, dominate when calculating distances, we scale the data to the same level.

Train-Test Split

General regime of 70%-30% split of the entire dataset, is followed, where 70% of the data is used for training the model, and remaining 30% is used for testing it.

Fitting the Neural Network:

The artificial Neural Network is used for training the model, and then making predictions. The network consists of one input layer, one hidden layer, and one output layer. The output layer consists of 3 nodes, depicting 3 labels of the target, 0, 1 and 2. Among the several tunable parameters, in this exercise, we mainly focus on tuning the ‘size’ parameter and the parameter of ‘activation function’.

The size function determines how many neurons or units will be there in the hidden layer. The neurons typically are associated with weight vector, and these weights are adjusted as learning proceeds. The neural network can be perceived as a complex perceptron, the major difference being, the perceptron is capable of learning only linear separation, whereas, a NN is developed to learn curves as well, and this is being determined by the activation function parameter used in training the model.

Trial 1: Size Used: **18**, Activation function used: **tanh**

Confusion Matrix Obtained:

	Actual		
Predicted	0	1	2
0	8	7	4
1	5	22	13
2	1	4	4

Performance Metrics:

	Sensitivity	Specificity	F1 Score
Class 0	0.57	0.79	0.48
Class 1	0.66	0.48	0.6
Class 2	0.19	0.89	0.26

Trial 2: Size Used: **15**, Activation Function used: **tanh**

Confusion Matrix obtained:

	Actual		
Predicted	0	1	2
0	7	10	1
1	5	18	10
2	2	5	10

Performance Metrics:

	Sensitivity	Specificity	F1 Score
Class 0	0.5	0.79	0.43
Class 1	0.54	0.57	0.54
Class 2	0.47	0.85	0.52

Trial 3: Size used: **18**, Activation function used: **relu**

Confusion Matrix obtained:

	Actual		
Predicted	0	1	2
0	5	6	0
1	6	22	10
2	3	5	11

Performance Metrics:

	Sensitivity	Specificity	F1 Score
Class 0	0.35	0.88	0.4
Class 1	0.66	0.54	0.61
Class 2	0.52	0.82	0.55

Trial 4: Size used: **20**, Activation Function Used: **Sigmoid**

Confusion Matrix obtained:

	Actual		
Predicted	0	1	2
0	5	6	3
1	8	22	10
2	1	5	8

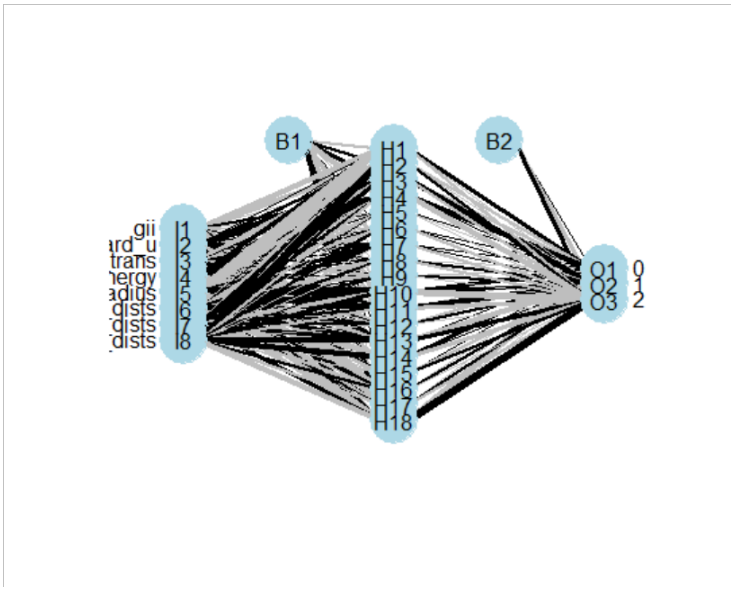
Performance Metrics:

	Sensitivity	Specificity	F1 Score
Class 0	0.35	0.83	0.35
Class 1	0.66	0.48	0.6
Class 2	0.38	0.87	0.45

Comparison among different trials:

		Class 0	Class 1	Class 2
Size	Activation Function	F1 score		
18	tanh	0.48	0.6	0.26
15	tanh	0.43	0.54	0.52
18	relu	0.4	0.61	0.55
20	sigmoid	0.35	0.6	0.45

Model Neural Network showing connections and nodes: (for size = 18, activation function = tanh)



Inferences:

As the model does not predict too accurate values, with extremely high scores, we assume that there is rarely any overfitting to be immediately taken care of.

From the above analysis, we could possibly draw the inference, that the NN performs quite well when size of the hidden layer = 18 and activation function is relu. With this combination, we obtain the maximum accuracy for all the three classes.

Testing on unforeseen data:

Now the trained model is tested on absolutely new unforeseen data of 4 records and predictions are made.

All three labels are obtained as predictions.

The compounds were as follows in the unforeseen data: W18O49, W O2, Ag2BiO3, Y bF e4(CuO4)3

They have been labelled as 1, 0, 1, 2 respectively. From the above observations, we can possibly conclude that since our model is pretty well at classifying class 1 and class 2, and relatively less efficient in classifying class 0, using relu activation function and size = 18, the confidence level with which we can report this result to the business, would be higher in case of class 1 and 2, and lower in case of class 0.

Problem 1: Gaussian Stochastic Process

a) **Given input x:** {0.0, 0.10, 0.20, 0.35, 0.55, 0.65, 0.80, 0.90, 1.00}, where x belongs to the real space

y = {1.0000, 0.9664, 0.9530, 1.0117, 1.1197, 1.1373, 1.1219, 1.0475, 0.7710}, where y belongs to the real space.

The mapping of x to y is given by the following:

f ← function(x) {ifelse (x < 0, 3 ∗ x + sin (5 ∗ x), x ∗ cos (5 ∗ x))}

z ← f(x)

g ← function(z) {cos (7 ∗ z/5) − z ∗ sqrt(abs(z))}

y ← g(z)

Task: To construct a Gaussian Stochastic Process to fit the inputs x and output y

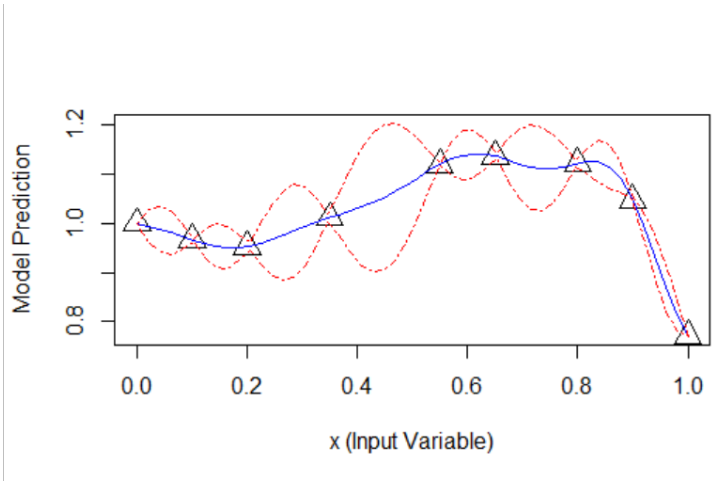
Approach and Solution:

A stochastic process is one which randomly change over time and involve uncertainty. A gaussian process is called stochastic if every finite linear combination of the random variables follows a normal distribution.

The GPFit Package in R uses multi-start gradient-based search algorithm with the aim of optimizing the deviance. The beta parameter is obtained by minimizing the deviance by using (L-BFGS-B).

Using the package, the one-dimensional input x and output y which is a function of x, as defined above, is fit.

Below is the plot obtained in visualization of RStudio.



Interpretation of the plot:

Correlation Parameters: beta_hat = 1.7465, sigma_hat = 0.0129.

The plot above shows the model predictions in blue and the uncertainty bounds in red line. The triangles denote the data points.

The 5%-95% error is maximum in the interval between 0.4 and 0.6. If we are to select three additional data points, we would select them closer to where the error is high, so we have our **new set of inputs** as follows:

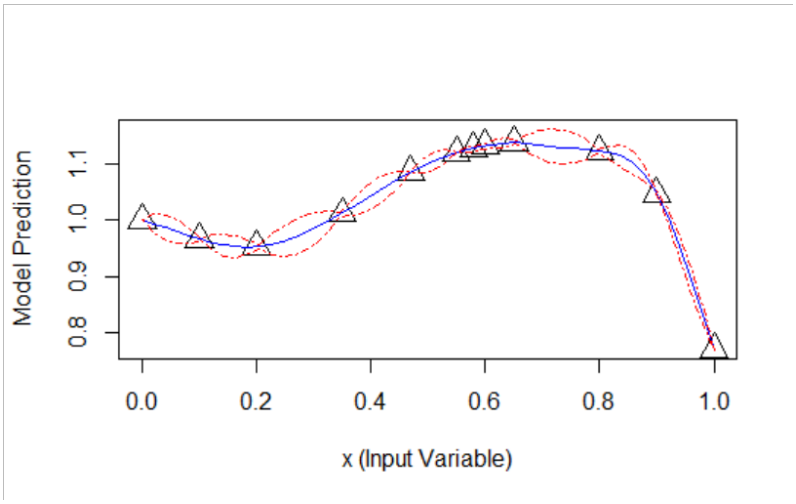
x = {0.0, 0.10, 0.20, 0.35, 0.47, 0.55, 0.58, 0.60,0.65, 0.80, 0.90, 1.00}

Through the mapping the new y values are:

1.0000000, 0.9664645, 0.9530561, 1.0117706,1.0847987, 1.1197085, 1.12758111,1.314977, 1.1373935, 1.1219198, 1.0475681,0.7710976

The new 12 x's are refit using the GPFit package and the below plot and parameters are obtained.

Correlation Parameters: beta_hat = 1.206, sigma_hat = 0.0288.



Inference after adding 3 additional x's:

From the graph we can observe that the error interval is reduced significantly. Previously with 9 data points, the largest error interval ranged from ~0.9 to ~1.18 and after re fitting the new x's and the y, into the Gaussian Stochastic Process, the error bound is seen to be reduced

and now it ranges from 1.0-1.08. Therefore, the error bound over the interval 0.4-0.6 reduced by about 71% [((1.18-0.9)- (1.08-1.0))/ ((1.18-0.9) = 0.71].

So, we can possibly conclude that adding more data points, can reduce the error interval, again adding too many data points can lead to overfitting, as too many data can count towards noise, and lead to inaccurate predictions.

b) A call is issued x ← maximinLHS (17, 2), which defines a sample of 17 input data points in 2 dimensions and the function is defined by:

```
f ← function(x) {pmin (3 ∗ x [, 1] + cos (5 ∗ x [, 2]), 1)}
z ← f(x)
g ← function(z) {cos (7 ∗ z/5) − z ∗ sqrt(abs(z))}
y ← g(z)
```

Task: To fit a two-dimensional Gaussian Process to the 2D x-input and 1D y-output.

Approach and solution:

The above call is issued in RStudio, and 17 data points are generated in 2D space. The x's are mapped with y, by the function defined above.

Now, using GPFit Package of R, the correlation parameters and surface plot of the Gaussian Fit of the x's and y is obtained as shown below:

Correlation Parameters: beta_hat 1: 0.82, beta_hat 2: 1.37, sigma^2 hat: 0.544

The surface plots:

As we are dealing with 2D input, we obtain a surface plot. The surface plot shows the inputs x1, and x2 on two axes, and the predicted values on the 3rd axis.

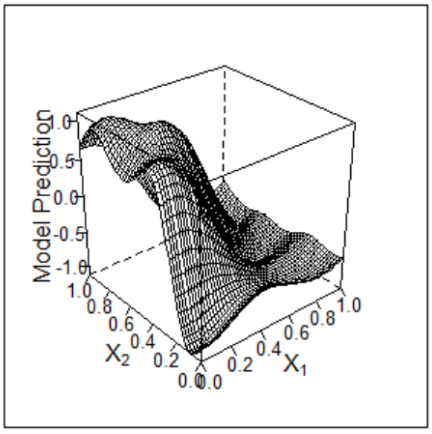


Figure 1: Inputs vs Predictions

The two plots on the left show the 3-dimensional plotting of the input values (2D), and the corresponding output value (1D).

Figure 1 shows the predicted y values, which are obtained after fitting a 2-dimensional Gaussian process to the x's and the 1D y output.

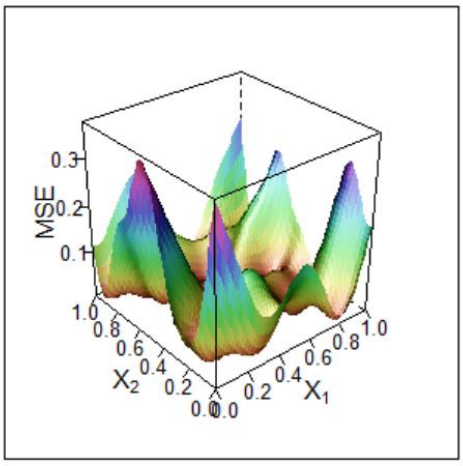


Figure 2:Error surface plot

Figure 2 shows the error or variability surface, obtained in the prediction. It plots the Mean Squared Error (MSE), on the 3rd axis, and the inputs on X1 and X2 axes respectively. The fit GP model can be used to predict new y-values and the average squared error between y-actual and y-predicted is plotted as the Mean Squared Error (MSE) in the surface plot.