

## Data – Wall Robot Navigation

### Exploratory data analysis

The provided data contains the raw values of the measurements of all 24 ultrasound sensors and the corresponding class label. There are a 5 variables and 5456 records.

Out of the 5 variables, 4 of them are numerical (V1, V2, V3, V4) and one is categorical (Class) which has categories 1, 2, 3, 4. All the numerical columns are analyzed using the `describe()` function. The `describe()` function gives us the count, mean, standard deviation, median, max, min, etc.

### Normalization:-

- Only the numerical columns are normalized in the range of 0 to 1. The numerical columns V1, V2, V3, and V4 are extracted to a new dataframe.
- Imported ***sklearn*** library and from that used the ***preprocessing*** package. Using the ***MinMaxScaler()*** in preprocessing package, normalized the numerical data in the range of 0 to 1.
- The 'Class' variable is added back into this dataframe so as to perform one-hot encoding.

### One-hot encoding:-

- One-hot encoding is performed on the categorical variable which is 'class'. It can be only done on numerical data types. Since the categories in 'class' variable are numerical, one-hot encoding can be performed directly on it.
- From ***sklearn.preprocessing*** imported ***OneHotEncoder***.
- After encoding, the category names become column names and new columns are formed. Since there 4 categories in 'class' variable, 4 new columns are formed.
- This is in the form of an array. So, it is converted into a dataframe.

Before using logistic regression or neural networks, the data has to be split into train & test data. To do this task, ***train\_test\_split*** package from ***sklearn.model\_selection*** is used. The test size is 20% of the data. The arguments for ***train\_test\_split*** are the numerical variables, target variable ('Class'), test size (0.2), ***random\_state***, ***stratify***.

Since the values in the 'Class' variable are imbalanced, ***stratify*** is used to take care of that problem.

## Logistic Regression

- Imported ***LogisticRegression*** package from ***sklearn.linear\_model*** library.
- Imported ***accuracy\_score*** package from ***sklearn.metrics*** library.
- Ran the logistic regression model.
- Accuracy was 0.8205. This model is not so accurate.
- Imported ***classification\_report*** package from ***sklearn.metrics*** library to print the classification report to print the precision, recall & f1-score.
- Imported ***confusion\_matrix*** package from ***sklearn.metrics*** library to print the confusion matrix. Got 4x4 matrix.

## Neural network

- Imported ***MLPClassifier*** package from ***sklearn.neural\_network*** library.
- Ran a neural network model with 2 hidden layers having nodes 100 & 50.
- The accuracy was 0.9853 which is pretty good.
- The classification report and confusion matrix were printed.

## False positive & False negative rates:-

- Imported the ***numpy*** package in order to perform this task.
- Defined a function ***false\_perc()*** which will take a confusion matrix as its argument and will print the false positive and false negative rates (in %) as a dataframe.
- Since there are 4 categories, the false positive and false negative rates for each of those categories will be displayed.
- The confusion matrix obtained as a result of both logistic regression and neural network were passed as arguments, one by one.

- The false positive and false negative rates of neural network are much lower than that of logistic regression.

false_perc(cm) # for logistic regression				
	1	2	3	4
FPR(%)	19.662058	9.077381	0.682261	0.000000
FNR(%)	12.471655	1.666667	16.666667	74.545455

  

false_perc(nn_cm) # for neural network				
	1	2	3	4
FPR(%)	0.921659	0.148810	0.194932	0.755124
FNR(%)	1.814059	1.428571	3.030303	0.000000

### Experimenting the neural network model with different number of hidden layers and nodes:-

- **1<sup>st</sup> case (100,50)** – Accuracy= 0.9853
- **2<sup>nd</sup> case (100,50,20)** – Accuracy= 0.9688.  
Accuracy has been reduced. Hence including 3 hidden layers is not good. So remaining with 2 layers.
- **3<sup>rd</sup> case (100,30)** – Accuracy= 0.9853  
Accuracy remains almost the same.
- **4<sup>th</sup> case (100,60)** – Accuracy= 0.9863  
Accuracy has increased slightly.
- **5<sup>th</sup> case (100,100)** – Accuracy=0.989  
The accuracy has increased even further to 98.9%. This model seems to be very good.

### Conclusion

While comparing all the above models, the neural network model with 2 hidden layers having 100 nodes in each layer seems to be the most accurate model.

## **MODIFIED VERSION**

### **(After reviewing)**

The only difference is that in the modified version, the normalized values are used for prediction (both logistic regression & neural networks). All the numerical variables have been normalized between the range of 0-1.

The logistic regression shows a lesser accuracy than compared to the previous version with an accuracy of **71.52%**.

Whereas in neural networks the accuracy increases when 3 hidden layers are incorporated. When increasing to 4<sup>th</sup> layer, the accuracy decreases. So sticking with 3 layers, the highest accuracy of **98.99%** with hidden layers having 100, 50 & 30 nodes.