# DATA SCIENCE ASSESSMENT

Name : Nik Faiz Afiq Bin Nik Ab Rahman

Company : Air Selangor

Date : 15th Feb 2021 – 19th Feb 2021

# INTRODUCTION

## Hypothesis

* How does PCA affect the performance of the model?
  + improve /reduce the model performance?
  + Hypothesis: Using PCA on our data will reduce the accuracy of the model, but not that much

## Input and Expected Output

Files:

* X\_train - to train and fit the data
* Y\_train - to train and fit the data
* X\_test - to test and find the best model
* Y\_test - to test and find the best model
* X\_test.submission - for real prediction to be submitted

Expected output:

* Prediction of X\_test\_submission - x and y coordinate of the sensors

## Objective

* Finding the best model to predict the X and Y coordinate for each of the receiver in X\_test\_submission using the data from X\_train.csv

## Assumptions made:

* replace NAN value with the value -100
  + - why -100? due to the nature of the NaN value, that it represent no signal or signal too weak, it is reasonable that we mark it as the minimum signal possible. Cannot really remove the value as it will skip the one with NaN value for our test data
* Assuming X and Y coordinate is independent, as we are predicting X and Y coordinate separately

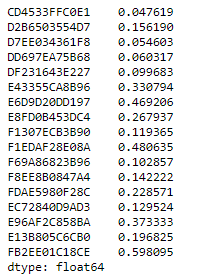
## Performance measurement:

* Euclidian distance between the predicted point and the real point

# METHADOLOGY

## Data Preprocessing

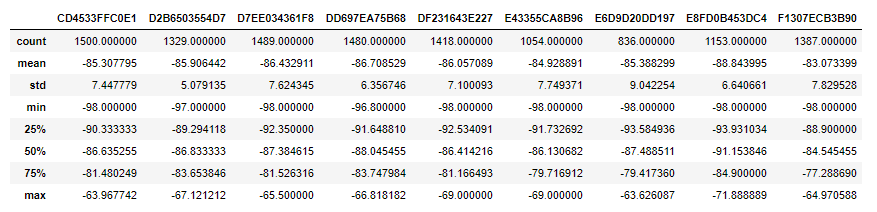
* Import the data to python
* Merge y\_train.csv file with the PinInfo.csv file to get the X and Y coordinate
  + This will enable us to have the target variable of X and Y coordinate instead of the PiinID
* Merge y\_test.csv file with the PinInfo.csv file to get the X and Y coordinate
* Find out the percentage of data in each column that have NaN value in X\_train files

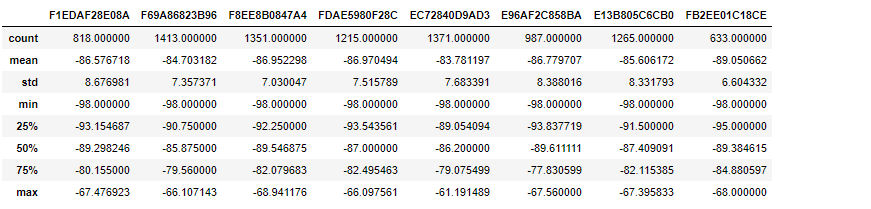


* Imput the NAN value with -100
* Repeat the same step for other X\_test files and X\_test\_submission files

## Data Analysis

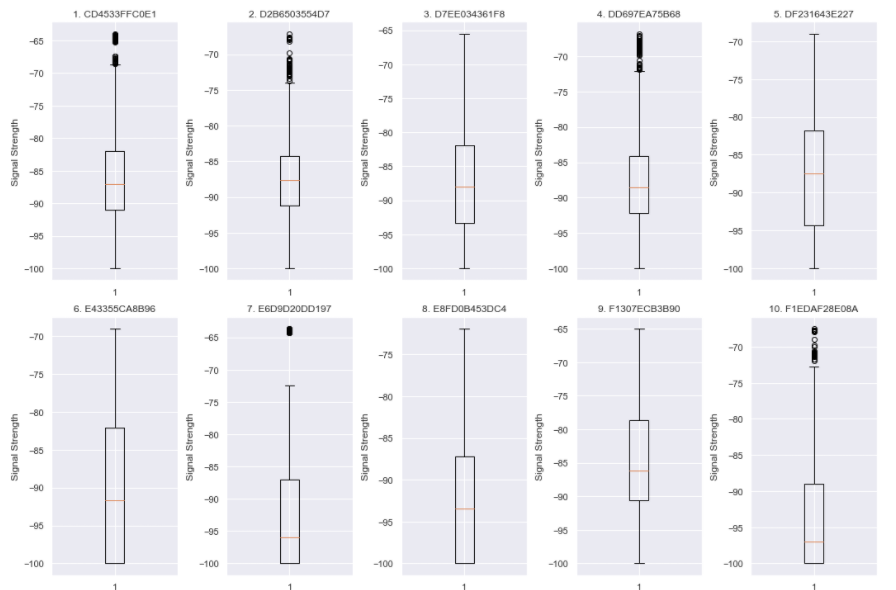
### Descriptive statistics

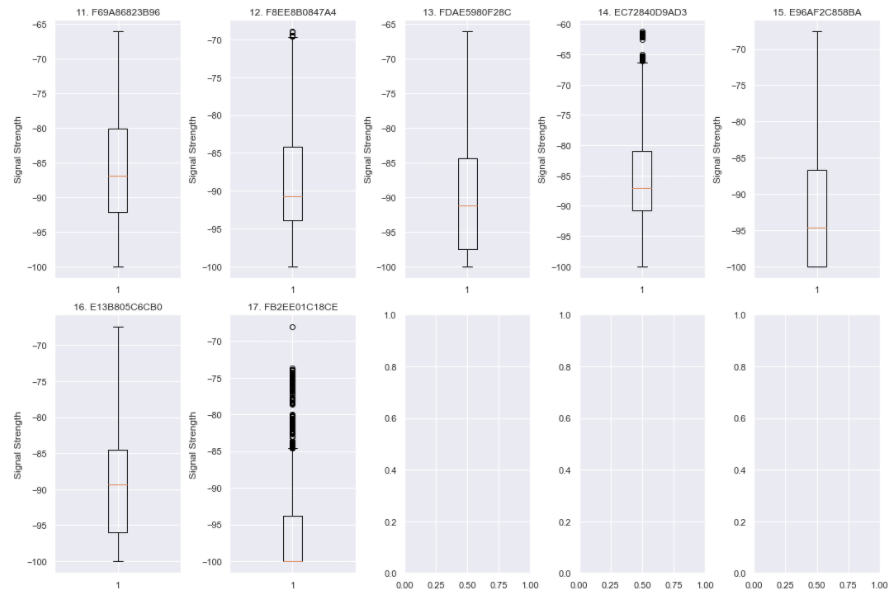




All of the data before the NA imputing have the value between -98 and -60. It is very hard to visualize all the data above, so the boxplot that is plotted in the next section can help us to visualize the statistics above.

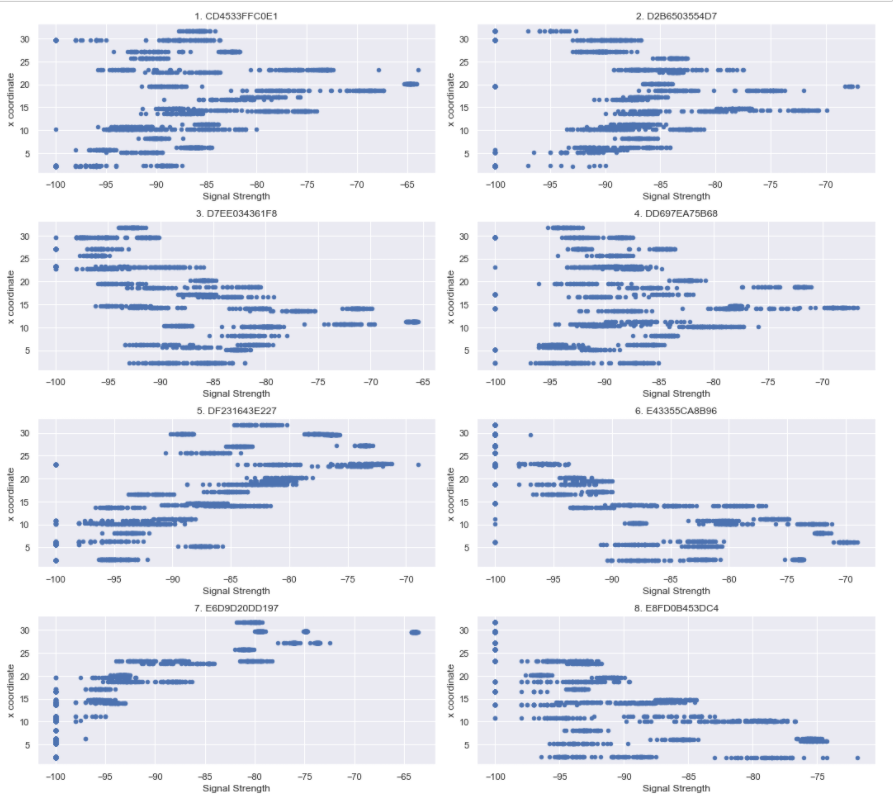
### Boxplot graph for each of the variables

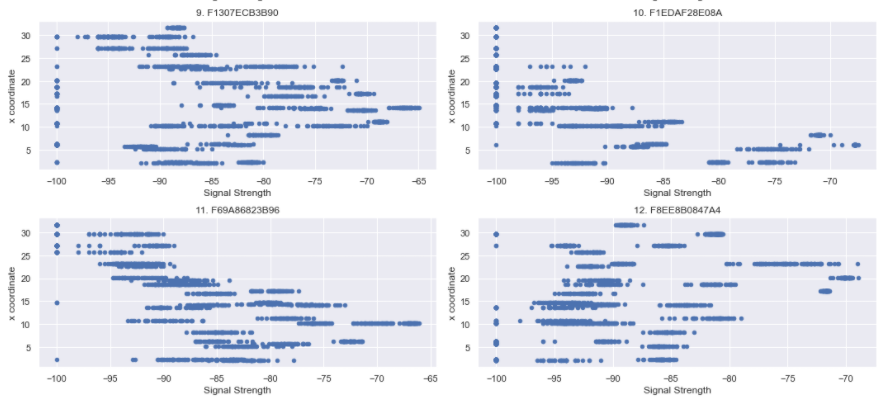


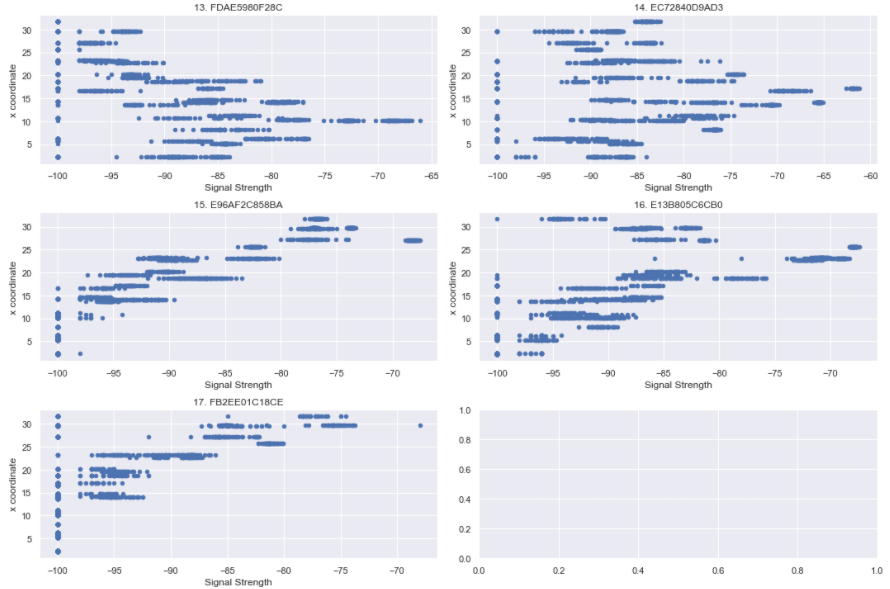
S

From above boxplot, we can easily visualize all the mean, median, outlier and the quartiles for each of the variables.

### Scatter plot of each variable vs target 1 (X coordinate of our target)

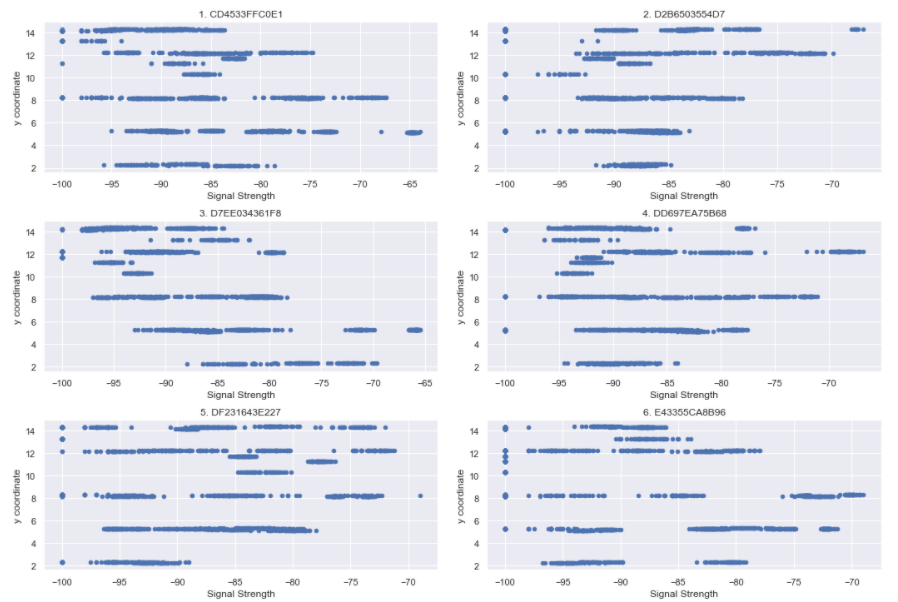


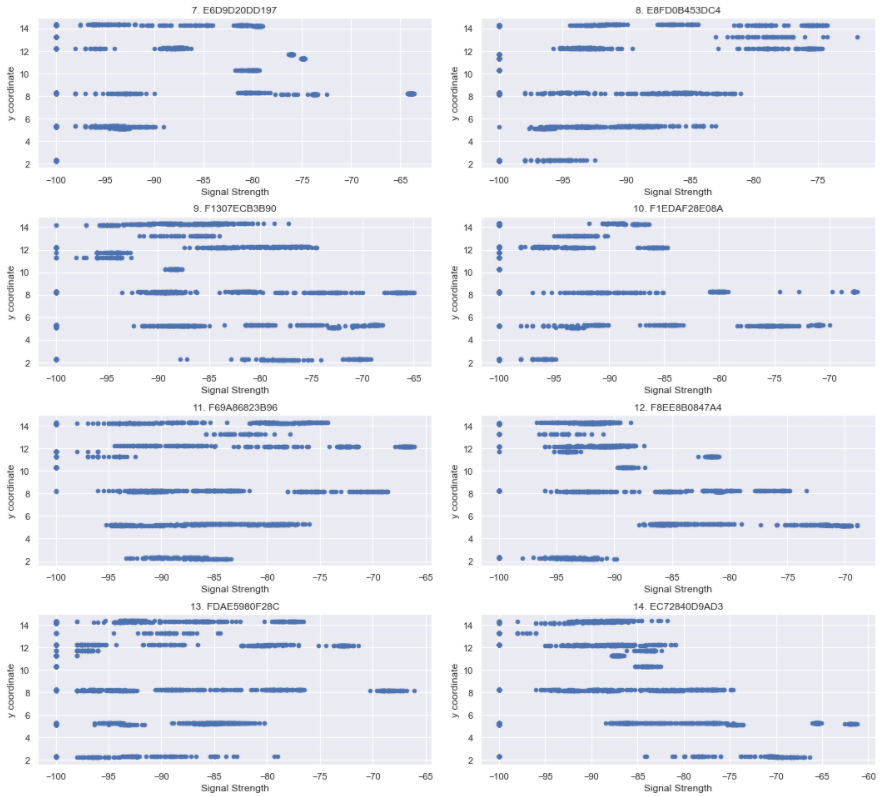


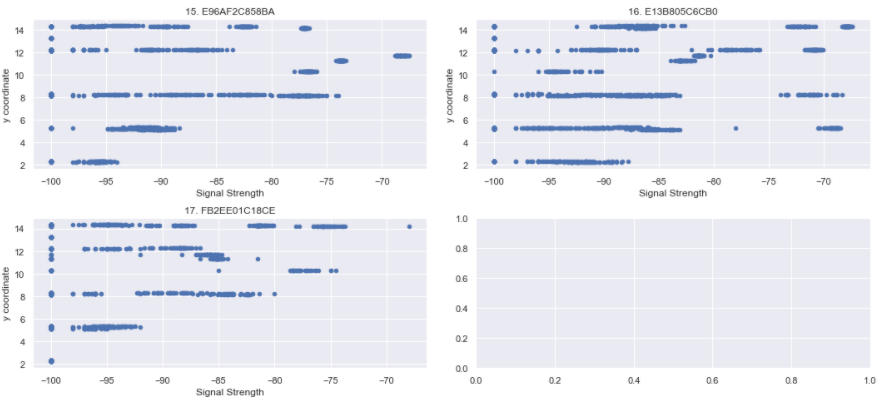


Interestingly, from this graph, we can guess the x-coordinate of the sensor by using the max signal strength. For example, we could see for the receiver 3. D7EE034361F8, the x-coordinate of the receiver is most likely between 10 and 15. The same guess can be done on almost all of the receivers.

### Scatter plot of each variable vs target 2 (Y coordinate of our target)







Unlike the scatter plot with target 1 (X- coordinate for the target), this scatter plot is really all over the place. The signal margin here is very wide and it is pretty hard to guess the y-coordinate from this graph.

## Modeling Using Machine learning

Models used:

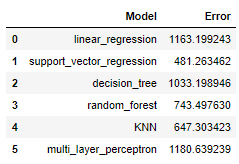
* Linear Regression
* Support vector machine
* Decision Tree
* Random Forest
* K-Neighbour Regressor
* Multi-layer Perceptron Regressor

From here we need to predict 2 target variable, in which we will name it target 1 and target 2. To measure the performance of each model, we will calculate Euclidian distance between the predicted point and the real point

### Modeling without using PCA

For the modeling, we will use mostly the default parameter for the model. After initializing, fitting and predicting the data, we will then calculate the error by measuring the Euclidian distance between the predicted point and the real point for each model.

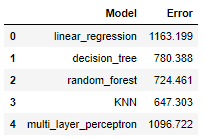
Here, to predict 2 target variables, we initially predict the two targets separately and combine the result together using the same model. The errors from these models are:



Another way to make prediction when there are 2 target variables is by using inherently multiout regression, where it is already built in some of the model in python (SVM not included).

Source: <https://machinelearningmastery.com/multi-output-regression-models-with-python/>

The error that we got from these models using multiout regression:

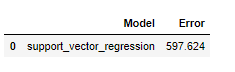


Here, we exclude Support vector Machine model as SVM model does not inherently have Multiout regression, so we need to use different ways to make prediction for SVM.

First, we can use Multi Output Regressor wrapper from SK-learn library to predict the target separately. It is actually the same from the first method above but we are using the SK-Learn library. The error for this model (same as above):



Aside from that, we can also use chained multioutput regression as the wrapper, and it gives the result as follows:

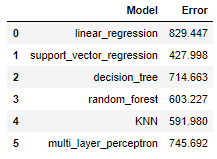


### Modeling using PCA for Dimentionality reduction

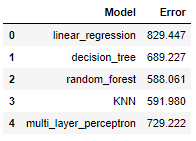
Here, we are using PCA as a dimensionality reduction before applying any prediction model. One of the benefits of using PCA is to reduce the calculation cost and time when doing Machine Learning on a very large dataset where we only have to process less data. Even though the number of columns is reduced, with PCA method, the reduced column can already explain almost all of the dataset.

Here we are using PCA to reduce the dimensionality to 5 components. On our model, those 5 components can explained 78.94% of the variance of our data.

For the modeling part, we repeat the same step as without the PCA. When manually predict the target separately, we get the following result:



The error that we got from these models using multiout regression:



The error when using chained multioutput regression as the wrapper:



When using PCA, it is expected that the performance dropped a bit or around the same with the original dataset. However, it seems here that **the model performs better when we are using PCA** across all of the model. So, we reject our hypothesis.

# RESULT AND CONCLUSION

Result summary in a table for non-PCA model:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Manually predict target separately | Inherently Multioutput Regression | Chained Multioutput Regression |
| Linear Regression | 1163.199 | 1163.199 | - |
| Support Vector Machine | 481.263 | - | 597.624 |
| Decision Tree | 1033.199 | 780.388 | - |
| Random Forest | 743.498 | 724.461 | - |
| K-Neighbour Regressor | 647.303 | 647.303 | - |
| Multi Layer Perceptron | 1180.639 | 1096.722 | - |

Result summary in a table for PCA model:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Manually predict target separately | Inherently Multioutput Regression | Chained Multioutput Regression |
| Linear Regression | 829.447 | 829.447 | - |
| Support Vector Machine | 427.979 | **-** | **427.651** |
| Decision Tree | 714.663 | 689.227 | - |
| Random Forest | 603.647 | 588.034 | - |
| K-Neighbour Regressor | 591.98 | 591.98 | - |
| Multi Layer Perceptron | 745.873 | 751.862 | - |

Overall, it seems that the model that use PCA before we do the modeling performs better. And the model that have the least error is the Support Vector Machine with PCA, and using Chained Multiout Regression. So, we will be using this model for the prediction on X\_test\_submission.csv.