# CLASSIFICATION PREDICTION FOR INDOOR LOCALIZATION WITH BLUETOOTH IBEACONS

K-Nearest Neighbor – The classifier of choice

Nishq Ravindranath
s3823660@student.rmit.edu.au

Affiliations
School of Science, Computer Science and Information Technology,
RMIT University, Australia

10th June, 2020

## Contents

## Abstract

Bluetooth Low Energy iBeacons serve as a way to allow for indoor localization and navigation. A 13 iBeacon setup on a grid was used to gather data from an individual, with their corresponding location. The labeled dataset from this was 1420 rows in size, allowing for the purpose of classification of these locations, determined from the different signal readings of each iBeacon. The dataset was normalized and explored using heatmaps and summary statistics, and it was found that some locations and iBeacons (when considering the non-zero signal readings) were less represented in the data when compared to others. This brought up certain limitations when classifying, however we were still able to tune and optimize two types of classification models – K-Nearest Neighbor and Decision Tree. The results from the K-Nearest Neighbor classifier had a 0.75 accuracy, with a 0.75 weighted average f1-score. The Decision Tree classifier, on the other hand resulted in an accuracy of 0.62, and a weighted average f1-score of 0.64. The K-Nearest Neighbor classifier was thus preferred over the Decision Tree classifier, as it is better suited to this kind of a dataset.

## Introduction

The BLE RSSI dataset comprises of the RSSI readings of an array of 13 iBeacons, located on the first floor of Waldo Library, Western Michigan University (Mohammadi, Al-Fuqaha, Guizani and Oh, 2018). The location of an individual walking around with an iPhone 6S collecting the iBeacon data was recorded in a grid-like format, with numbers 1-18 as rows, and letters A-Z as columns. The collection of the location thus gives us a labeled dataset, with a research goal of classifying the location of an individual on this grid based on the readings from all 13 iBeacons.

The analysis that is conducted in this report is done with the help of visualizations and graphs, as well as some statistics on the individual iBeacons and individual locations. Furthermore, pairs of columns are explored – each iBeacon with the location column. The analysis conducted also extends to the limitations in the dataset. The results from the models can be used for location classification, and the measure of success of the models is the classification accuracy, as well as the f1-score of the models.

In terms of the broader business goals of this research goal, indoor localization and navigation has many applications, from helping people in large departmental stores find products, to tracking and gaining insights on how people shop in shopping malls.

## Methodology

There were two datasets available, one labeled with 1420 rows, and one unlabeled with 5191 rows. For the sake of this report and the subsequent analysis we used the labeled dataset. The analysis on the labeled dataset was conducted primarily on 14 columns – 13 columns for each iBeacon, and one for the corresponding location. The date column was not used because it doesn't have any logical impact on the iBeacon readings, nor the corresponding location label for those readings. The data needed to be prepared before it was ready for exploration and modelling, and the following steps were taken in the data preparation phase:

- Custom column names were added when loading the data from the CSV file to improve readability.
- Each column's datatype was checked and confirmed to be correct.
- iBeacon reading values were normalized to the range 1-100
    - First we had to take all the iBeacon columns as a subset of the data, and find the maximum and minimum values across all beacons.
    - Applied the normalization function.
- Checked and confirmed the validity of location column values.
- The date column was split into two additional columns – one for day and one for time.
- The location column was split into two – one for the letter and one for the indexed number.

Various analytical techniques were used for exploring each column individually:

- The mean for each iBeacon was computed.
- The non-zero signal readings for each iBeacon were computed.
- A histogram was plotted for each iBeacon's signal strength.
- A bubble chart was plotted with each of the iBeacon's signal strength mean on the y-axis, and the iBeacon's non-zero signal readings as the z-value to define the size of the bubbles.
- A heatmap was constructed for the location column data, by first making a dataframe with each location's individual occurrence counts in the dataset, and then creating a pivot table with the counts for each location as the values, the letters as columns, and the location numbers as the index.

Heatmaps were constructed to explore pairs of columns, with each iBeacon's signal readings and the corresponding location to visualize iBeacon strength on a grid, much like the real world from which the data was made.

For the data modelling, the research question chosen was classification, with the models used being K-Nearest Neighbour and a Decision Tree. There were a few restrictions in the dataset, with some factors not accounted for in the data, such as the fact that there may be objects or people in the way blocking the signals, therefore making the data more impure. Moreover, there are only 1420 classified rows, which is a small number of rows to have for the number of different features available, and the number of different possible locations. Not all the locations possible (from A – Z and 1 – 18) are present in the dataset. Furthermore, for the locations that are present, there isn't an even distribution of each label. For example, there are 34 rows that correspond to location K04, but only for classifying location D14, both of which are within a block from the nearest iBeacon. There also isn't even distribution of enough non-zero signal data for each iBeacon, for example there are 497 rows with non-zero signal readings for iBeacon 2, but only 25 for iBeacon 11.

For parameter tuning, nested loops were used to try different permutations and combinations of each model's most important parameters (relevant to this dataset), and get the combination of parameters that yield the best accuracy. For the K-Nearest Neighbor model, the following parameter lists were used to make combinations:

- Metric: 'minkowski' and 'manhattan'
- Weight: 'distance' and 'uniform'
- p-value: 1, 2, 3, 4, 5, and 6
- k-value: integers from 1 to 30
- random state: integers from 1 to 20
- test sizes: 0.1, 0.15, 0.2, 0.25 and 0.3

For the Decision Tree model, the following parameter lists were used to make combinations:

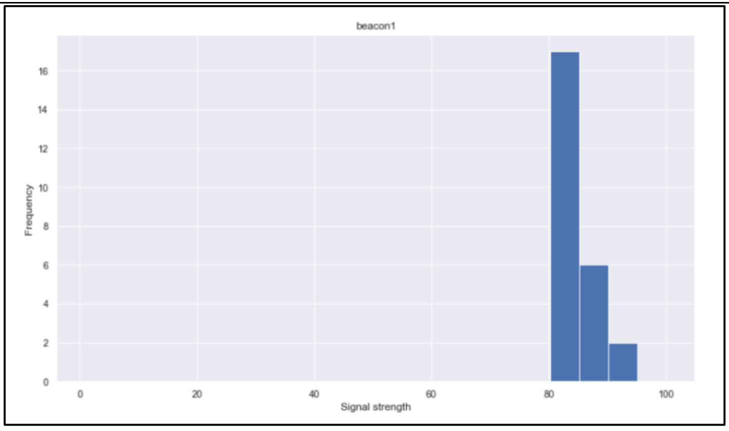
- Criterion: 'gini' and 'entropy'
- Random state: integers from 1 to 10
- Minimum sample split: integers from 2 to 34
    - Note: even though lower values of minimum sample splits can lead to overfitting, it was necessary for this model as in the dataset the lowest location value count is 2, and the highest is 34.
- Max depth: integers from 5 to 50

After tuning the parameters, the optimum model was saved. Model retraining is straightforward, as long as the parameters that have been optimized for post parameter tuning are used. K-Folds Cross Validation was then used to get the best split of training and testing data, which significantly improved the results of both the K-Nearest Neighbor model, and the Decision Tree model. However, this could be due to a smaller testing dataset from the number of folds chosen (15). Different values of k were tested for the cross validation, with anything over 15 having high accuracy but really small testing dataset, therefore skewing the model to possibly seem like it was more accurate than it truly was due to lack of testing data.

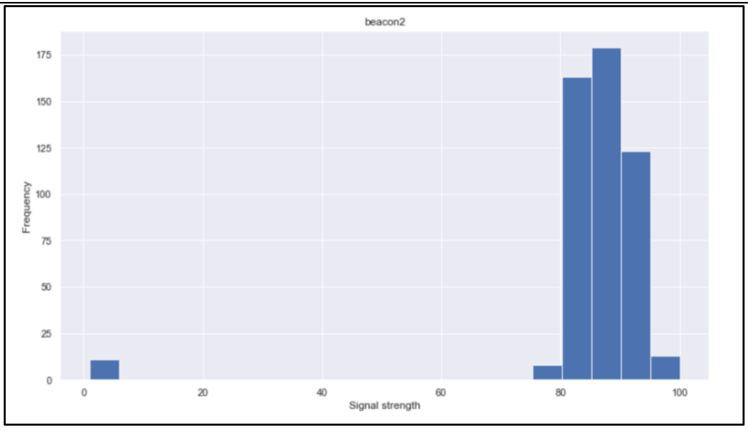
To solve this problem of less testing data, as well as underrepresented location label data, we had to employ our own method for validation, by creating our own testing data. This was possible since the iBeacons are arranged in a grid, and by passing our own testing data we can check how well the classifier works by comparing the results of the classifier with the actual locations of the iBeacons on the grid. Thus, for example, we created test rows with all the iBeacon values except iBeacon 5 set to 0, with iBeacon 5 set to 100, and then get the prediction from the classifier for this test row. This was for each iBeacon in a loop. Furthermore, since we may want to test other locations on the grid with multiple signal readings from different iBeacons (as is with the data), a grid validation function was written where the user can input their own selection of iBeacons, and corresponding strengths and get an output from the chosen classifier, and then see how closely the classifier's prediction lines up against the real location on the image.

## Data Exploration Results

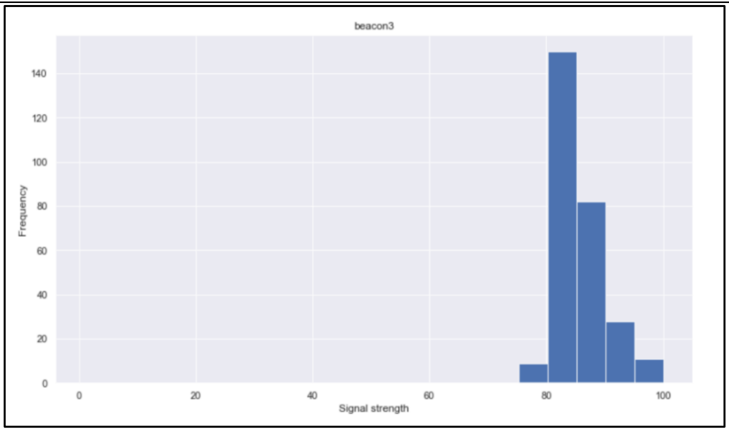
### Individual iBeacon Histograms



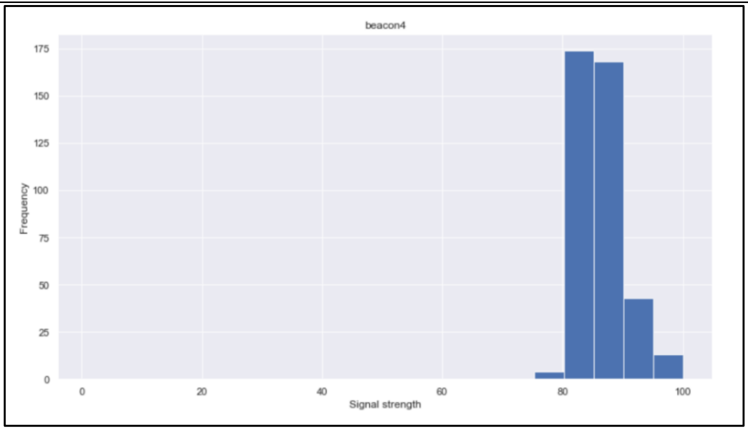
Number of non-zero signal values: 25
Mean signal strength: 85.19



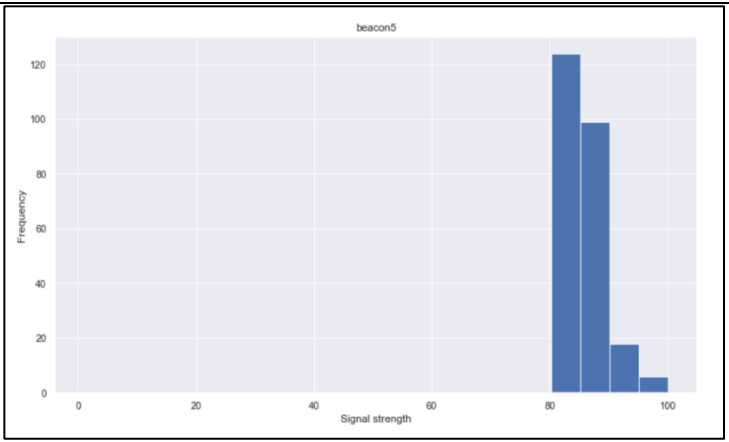
Number of non-zero signal values: 497
Mean signal strength: 85.47



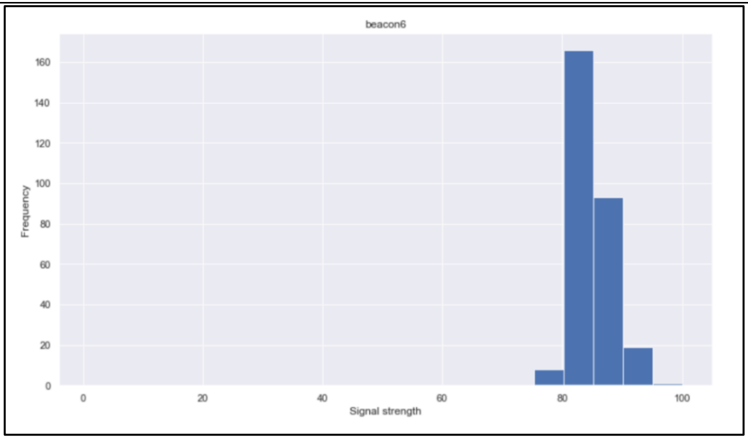
Number of non-zero signal values: 280
Mean signal strength: 85.57



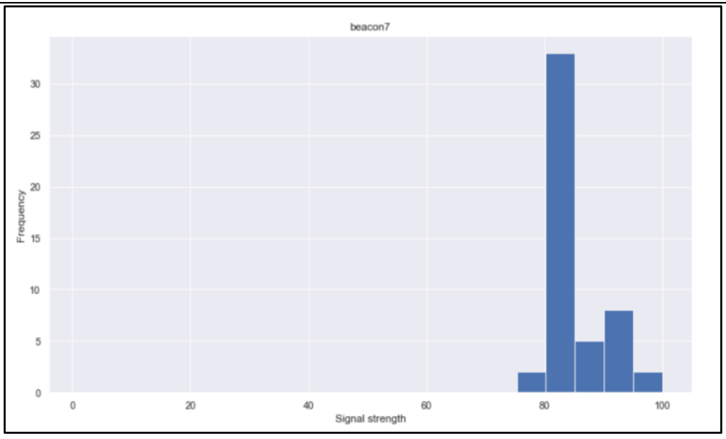
Number of non-zero signal values: 402
Mean signal strength: 86.4



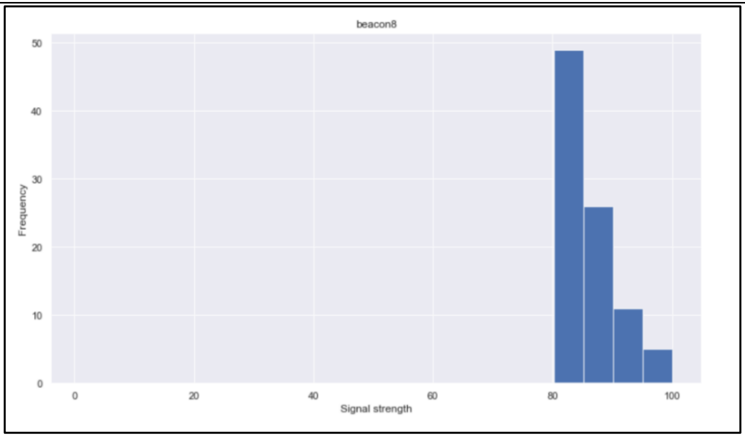
Number of non-zero signal values: 247
Mean signal strength: 85.73



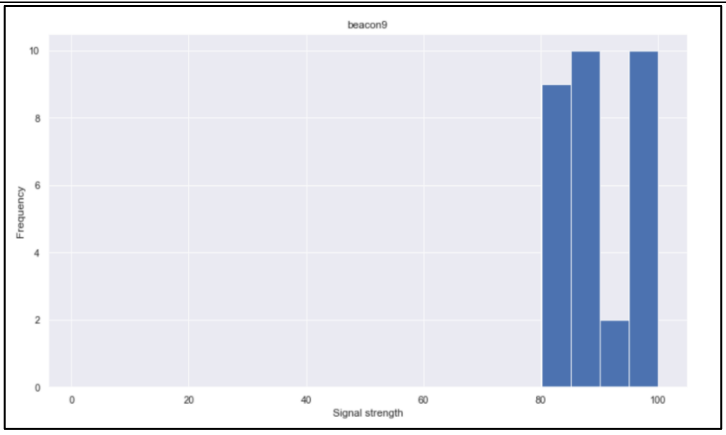
Number of non-zero signal values: 287
Mean signal strength: 85.09



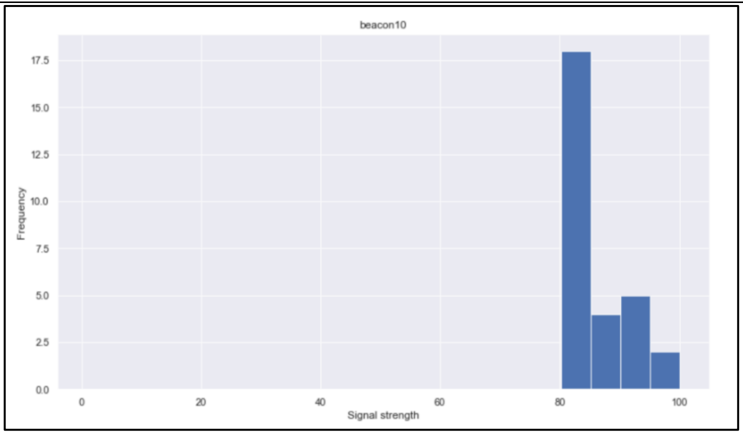
Number of non-zero signal values: 50
Mean signal strength: 85.45



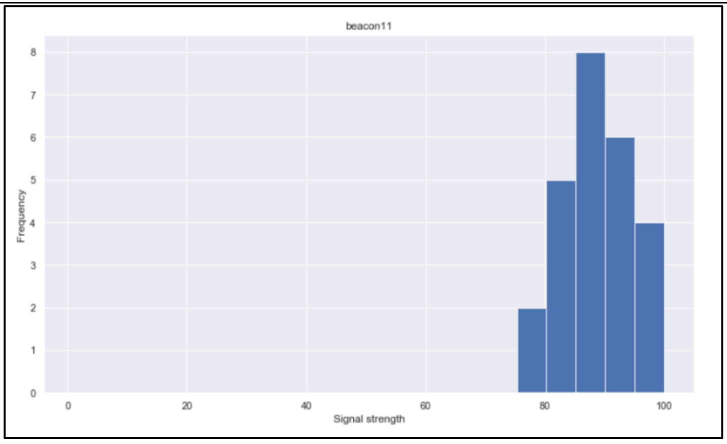
Number of non-zero signal values: 91
Mean signal strength: 86.41



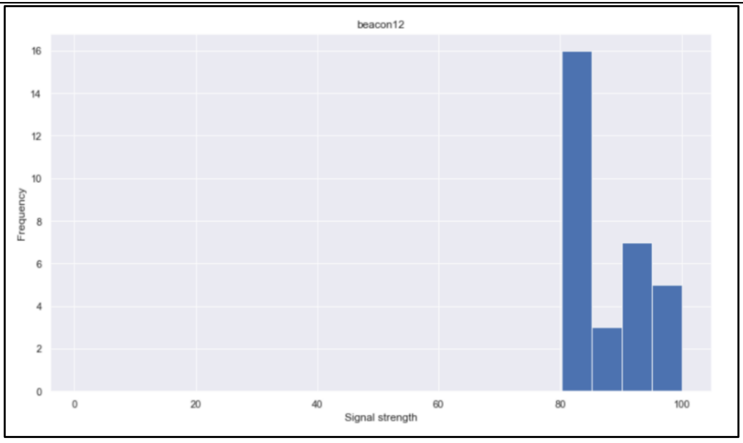
Number of non-zero signal values: 31
Mean signal strength: 90.19



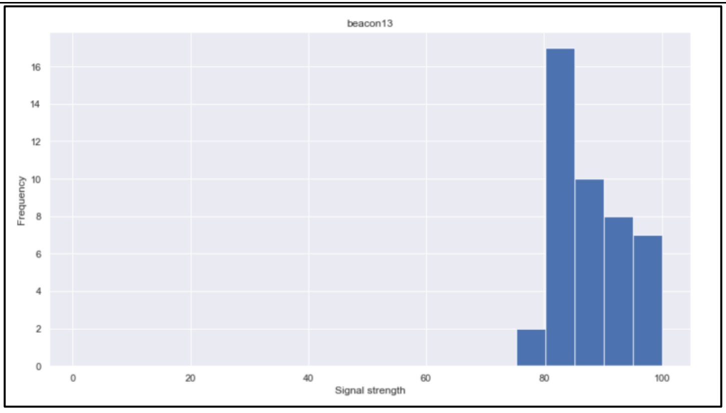
Number of non-zero signal values: 29
Mean signal strength: 86.37



Number of non-zero signal values: 25
Mean signal strength: 88.19

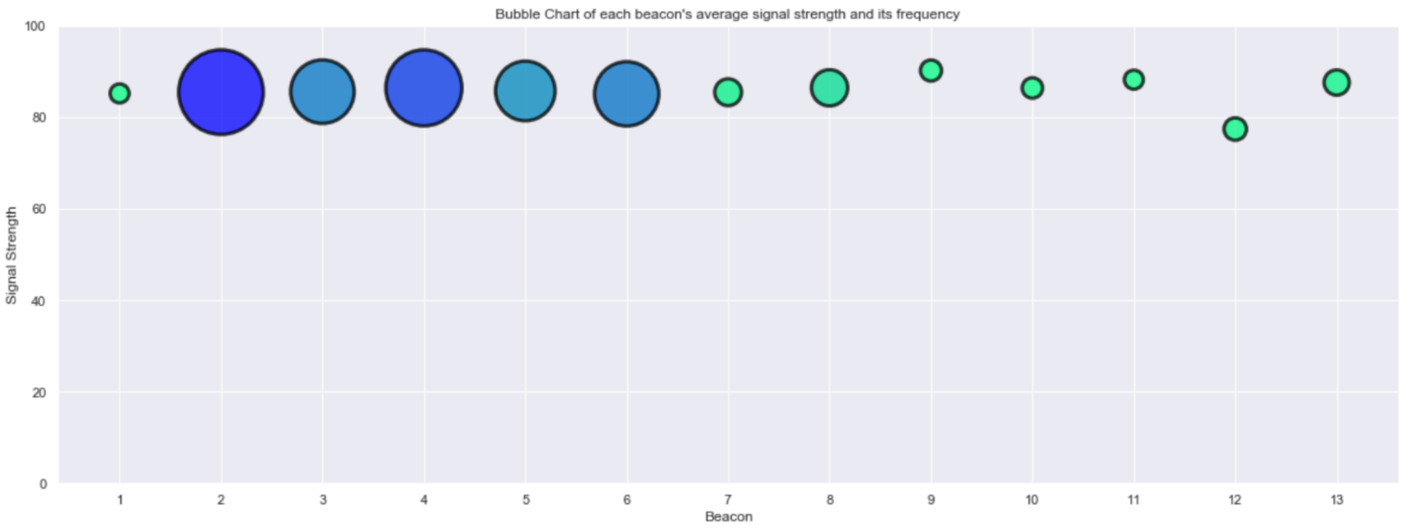


Number of non-zero signal values: 35
Mean signal strength: 77.4



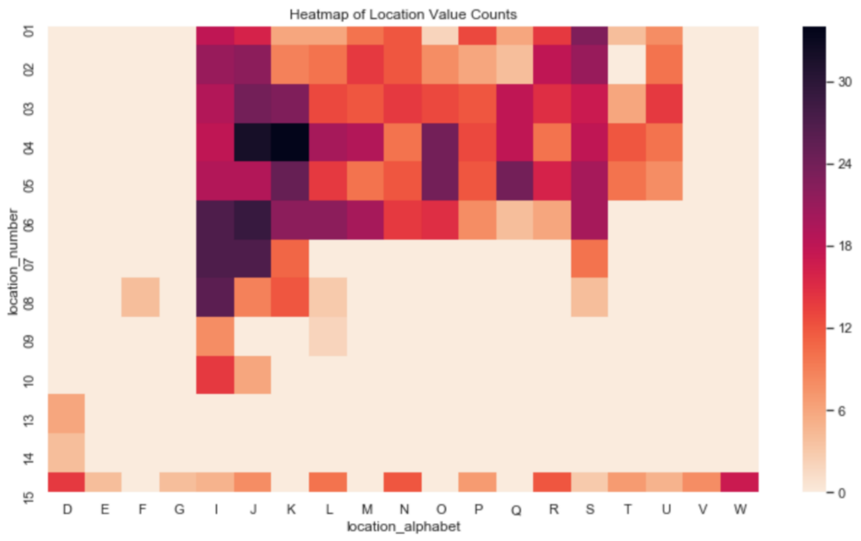
Number of non-zero signal values: 44
Mean signal strength: 87.57

## Bubble Chart



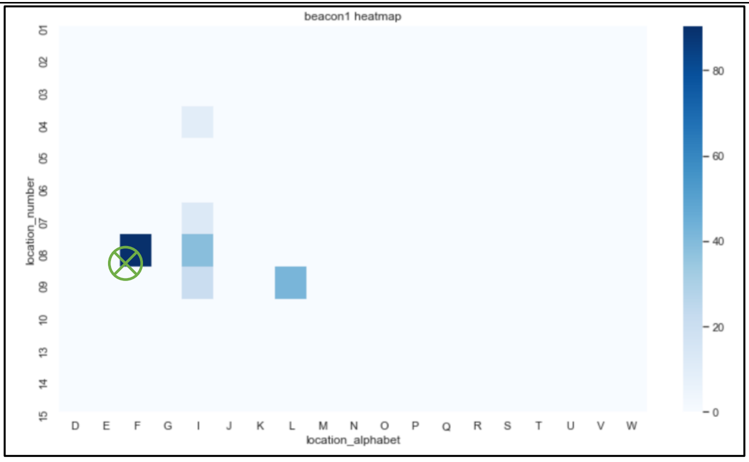
Here we can clearly visualize the disparity the in the data available for each iBeacon, with iBeacons such as iBeacons 2-6 having lots of non-zero data available, whereas iBeacons 1 and iBeacons 7-13 have comparatively very little non-zero data available.

## Heatmap of Location Value Counts

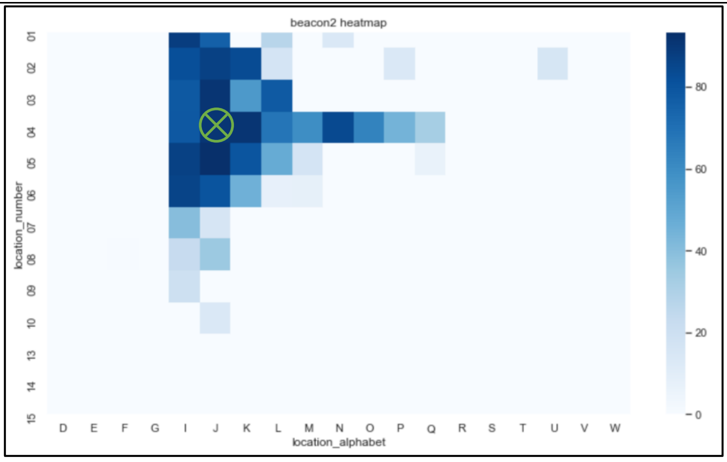


Clearly, the locational data is not evenly spread here. A lot of locations are entirely absent from the data. However, we also gain valuable insight from the heatmap, as we can see that locations such as K04 and J04 seem to have the greatest number of values, which matches up with the value counts. This tells use which locations were visited the most, or rather has the most amount of data associated with them, as the location visits are dependent on the signal strengths of the iBeacon(s) in that area.

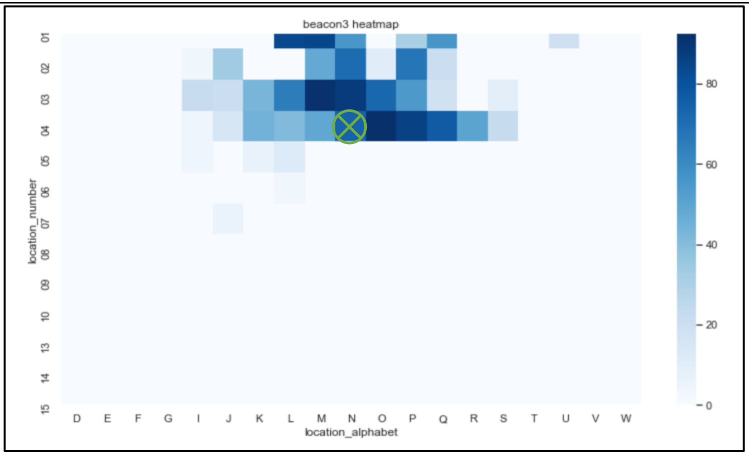
## Heatmap of every iBeacon with respect to the location



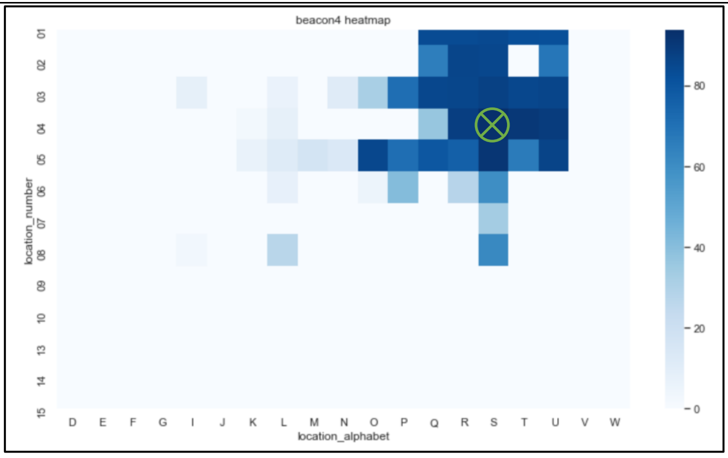
iBeacon 1 only has a relatively fewer number of non-zero signals, (25), and this might be why the heatmap is sparse. However, it is also surrounded by the greatest number of obstacles (windows, rooms, escalators/stairs), which could be another reason.
Most of the non-zero signal readings seem concentrated at the very exact position of the iBeacon itself, with fainter signals coming around iBeacon 5 and iBeacon 8.



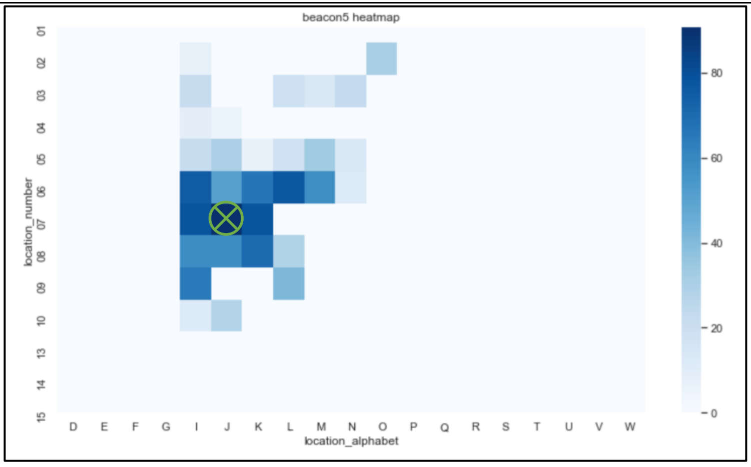
iBeacon 2 has the most number of non-zero signal readings, and a lot of the strongest signals seem to come around its position of J04, however there are also a few number of high non-zero signal readings for iBeacon 2 at the position of iBeacon 3. There are also no readings past the G column to the left, this is probably due to the wall that is present at the I column.



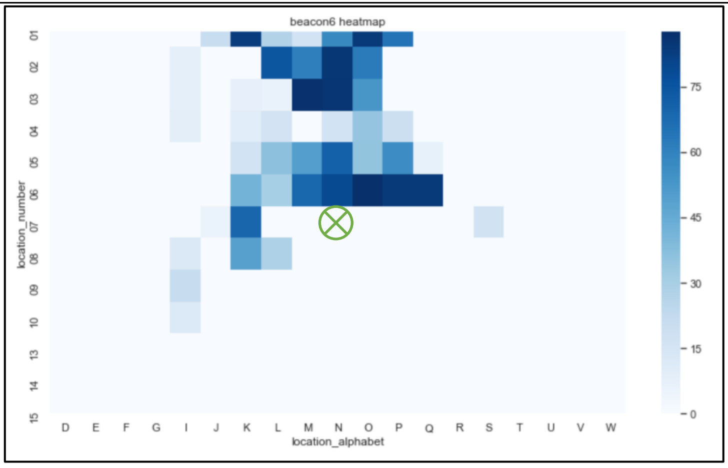
iBeacon 3 seems to be having its strongest signal readings around its actual position (N04). Its signal readings are strongest in the 3rd and 4th rows, and after the 4th row it tends to be quite faint.



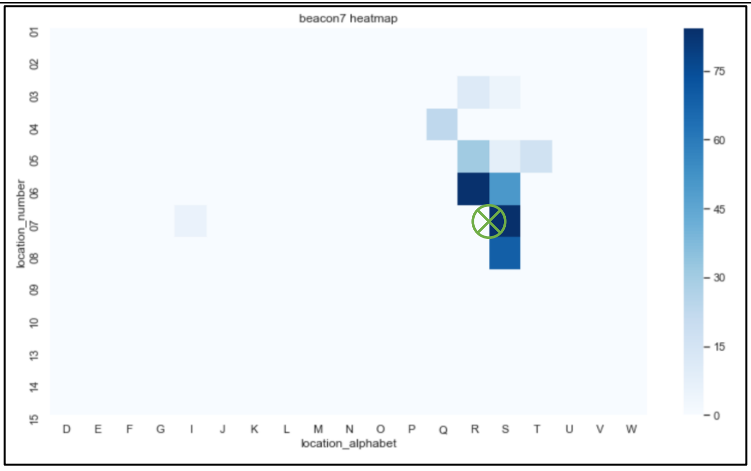
iBeacon 4 seems to be predominantly spread in the upper right corner of the grid (ignoring columns V and W due to a clear obstacle in the iBeacon layout image). It seems to have really strong signals (80+) as many as 5 blocks away from its location (S04), and covers a wide area of the grid.



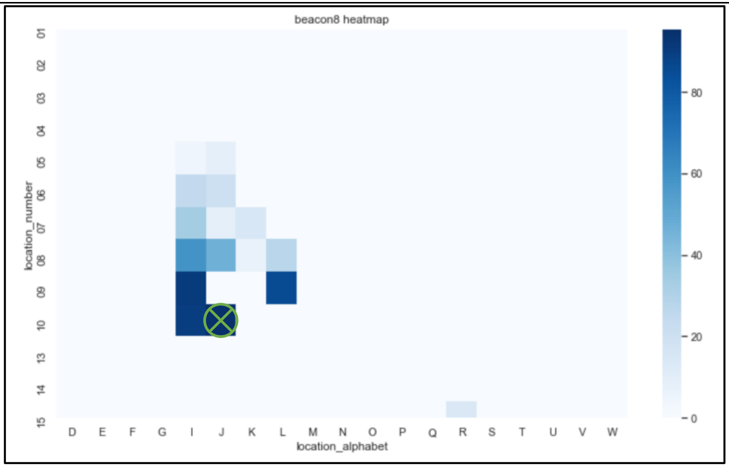
iBeacon 5's strongest signals come from its actual position (J07). There are faint signals from this iBeacon as high up as the first row. Like iBeacon 2, there are no signals past the G column to the left.



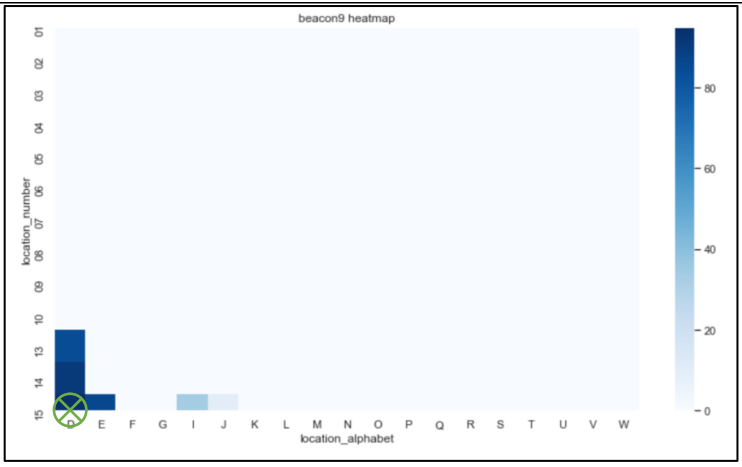
iBeacon 6 has no signal readings from its actual position (N07). There appears to be no data in the dataset for that exact location. However, iBeacon 6's signal strengths are strongest in the row above its position, as well as predominantly in the 1st, 2nd and 3rd rows with very strong (75+) signal readings, which the actual iBeacon is many blocks away from. It is therefore covering a large area.



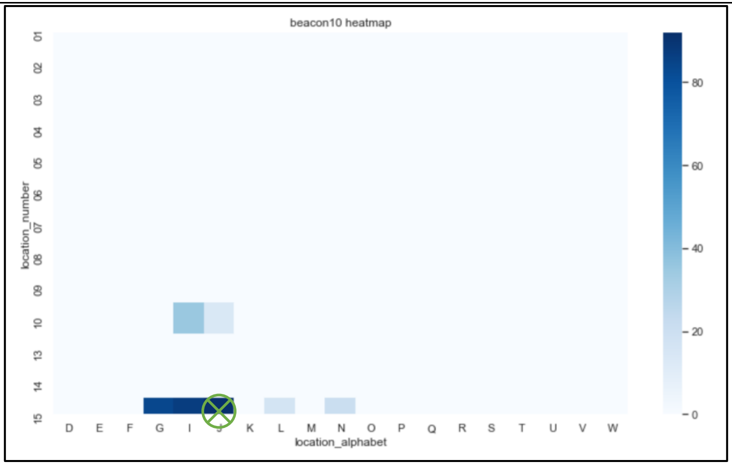
iBeacon 7 doesn't have a relatively wide area of coverage, its strongest signals coming from within 2x2 block distance from its actual position (R07/S07). This might be due to the room that it is right next to in the T column, and also due to the low number of non-zero signal readings, 50.



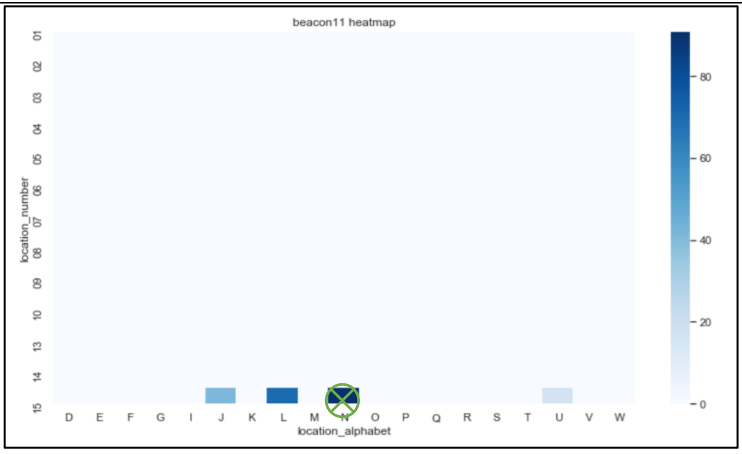
iBeacon 8 gets its strongest readings from its actual position, as well as blocks I10, I09 and L09. There are fainter signals from some blocks above, however, none from any of the other blocks in its immediate surroundings other than the ones mentioned.



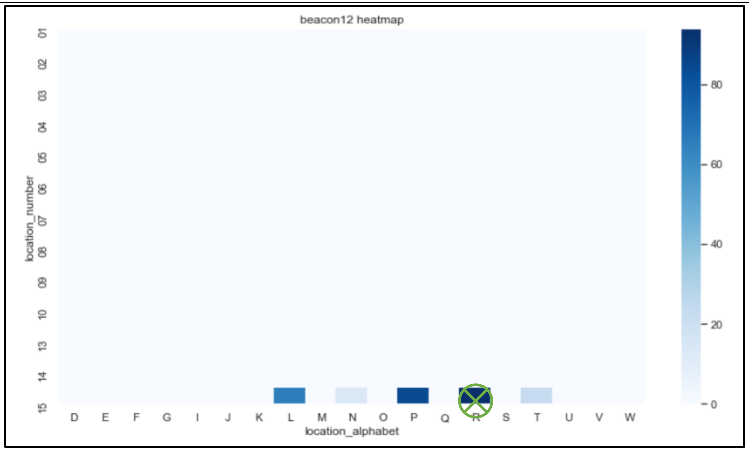
iBeacon 9 has relatively few non-zero signal readings (31). Its strongest blocks are its actual location (D15), and 2 blocks right above it (D13, D14) and one to its right (E15).



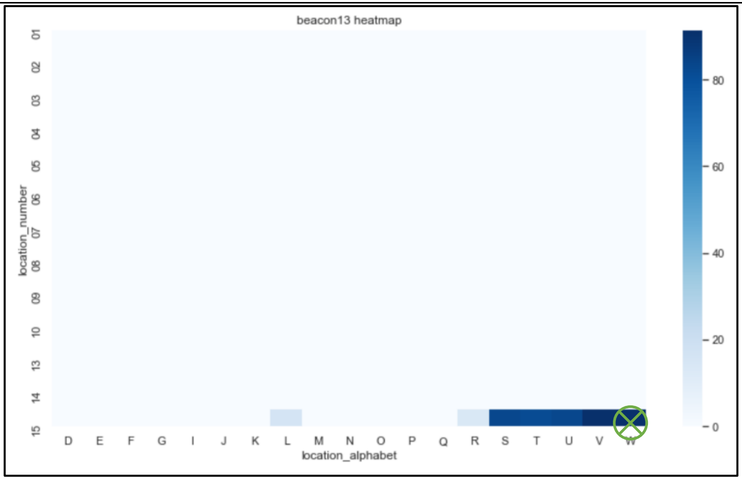
iBeacon 10 has relatively few non-zero signal readings (29). Its strongest blocks are its actual location (J15), and two blocks to its immediate left (G15, I15).



iBeacon 11, like iBeacon 1, has the lowest number of non-zero signal readings (25). Its strongest readings come from its actual position (N15). It has no readings from its immediate-surrounding blocks; however it has 50+ readings from block L15.



iBeacon 12 has the lowest mean of non-zero signal readings of all the iBeacons (77.4), and has a low number of total non-zero readings (35). Like iBeacon 11, its strongest signals are from its actual position (R15), and strangely, all of its other readings are one block apart on the same row.



iBeacon 13 seems to be have a strong effect across 4 additional columns to its left

## Data Modelling Results

### Optimized set of parameters for the K-Nearest Neighbor Classifier:

Max accuracy achieved = 0.42 with the following metrics:

- Test size: 0.1
- Random state: 3
- Metric: ‘minkowski’
- Weight: distance
- p-value: 2
- k-value: 2

### Optimized set of parameters for the Decision Tree Classifier:

Max accuracy achieved = 0.39 With the following metrics:

- Criterion: ‘gini’
- Random state: 7
- Min samples split: 2
- Max depth: 21

## K-Nearest Neighbor Classification Report results:

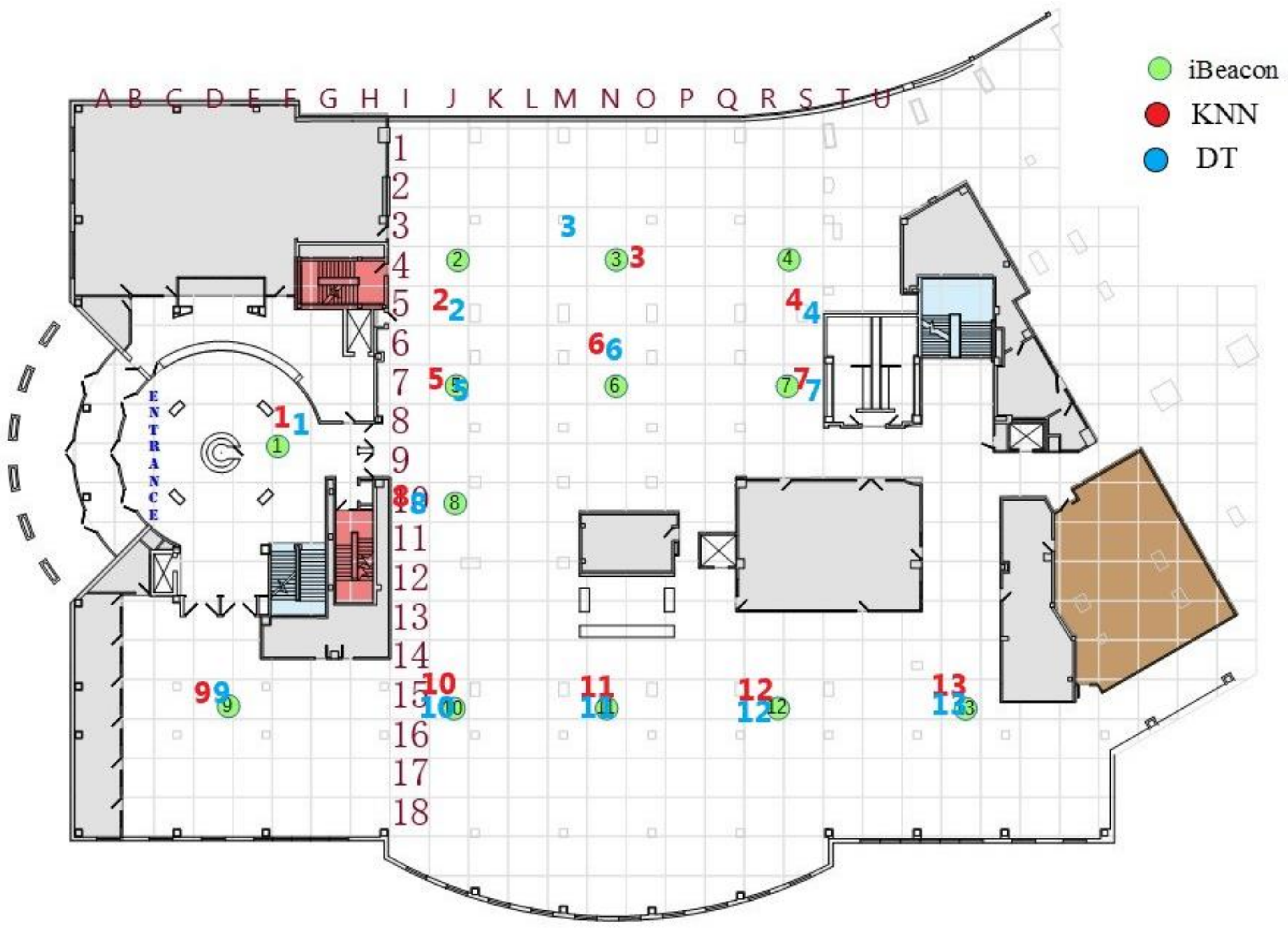
| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Accuracy | | | 0.75 | 95 |
| Macro Avg | 0.56 | 0.51 | 0.51 | 95 |
| Weighted Avg | 0.84 | 0.75 | 0.75 | 95 |

## Decision Tree Classification Report results:

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Accuracy | | | 0.62 | 95 |
| Macro Avg | 0.48 | 0.40 | 0.41 | 95 |
| Weight Avg | 0.77 | 0.62 | 0.64 | 95 |

It can be seen here that KNN yields the better precision, recall and f1-scores for the same fold of training and testing data.

## IBeacon Location Classification with both K-Nearest Neighbors and Decision Tree Classification



In terms of iBeacon location classification, both models give almost identical results, with the only difference being in the classification of iBeacon 3, which the KNN classifier gets more accurately, however the DT classifier's prediction is still within a 2x2 matrix block of the actual location of iBeacon 3.

## Grid Validation

Since there can be an innumerable number of possible permutations and combinations of signal strengths from iBeacons, some testing was done in the jupyter notebook by calling the gridValidation function on both the classifiers and comparing results. The notebook allows for any user to try it out and make their own comparisons themselves.

It should be noted that this technique of testing does not account for possible real-world factors that affect the data, such as objects and people in the way, but just considers the numerical estimates from the grid image.

## Discussion

For the KNN classifier, the best metric was ‘minkowski’. While it may seem like ‘manhattan’ is better suited to this kind of a dataset, because the data represents a grid, it just happens so that unlike a city-like grid, the movement of the individual is free and one can walk diagonally across parts of the entire grid.

If we look at the jupyter notebook and the full classification report, we can see that for some locations, the precision and recall is 0, even though there is non-zero support for them. While this is not ideal, it does make sense given the lack of data for these locations. This is further backed up by the fact that at the very maximum, the support for these locations is 2 (when we look at the final, optimized classifiers). One option to circumvent this could be to remove those locations, however then there would be even fewer locations to train and test on, and as we have already mentioned and shown in a heatmap, the data does not evenly cover all the locations.

When looking at the comparison between both models, the KNN classifier fares better in terms of precision, recall and the final f1-score (after optimization and tuning), making it a better choice for classification. Here we primarily consider the weighted average over the macro average to assess each model and compare results, because the weighted average considers the proportion of each label to others in the dataset, which is particularly useful for our dataset, since it is unbalanced in terms of label counts.

Classification report results aside, the data technically represents locational data, with relative distances measured based on the iBeacon signal readings. This sort of an application is more suited to K-Nearest Neighbors than Decision Trees. Furthermore, the data from these iBeacons is continuous, which Decision Trees are not suitable for, as they lose information when categorizing the variables into different categories. The Decision Tree could also suffer from overfitting here, as the max_depth is considerably large ( =21) and the min_samples_split and min_samples leaf are both small ( =2), however they have to be that small because the smallest location value count is 2.

## Conclusion

As has been mentioned in the report, there are certain limitations with regards to the dataset, with an event amount of non-zero signal data for each iBeacon, not enough counts for each label in the dataset, and not all possible labels from the entire grid represented. Furthermore, there are other factors to account for, such as the fact that there may be objects or people in the way blocking the signals, therefore making the data more impure. However, given these limitations it is still possible to achieve a relatively high precision, recall, accuracy and f1-score after choosing the right training/testing data with K-Folds Cross Validation. We can also create our own testing data with the grid validation function that was created for this report. In terms of choice of classifier, given the fact that we are dealing with continuous data, a Decision Tree classifier will not work as well as a K-Nearest Neighbor classifier, and the results reflect this.

## References

Mohammadi, M., Al-Fuqaha, A., Guizani, M. and Oh, J., 2018. Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services. *IEEE Internet of Things Journal*, 5(2), pp.624-635.