Developing Classification Models for Indoor Localisation Predictions Based On BLE RSSI Signals

By Stephanie Choo[1]
Master of Data Science
[1] School of Science, Computer Science and Information Technology,
RMIT University, Australia
Student number: s3846487
s3847487@student.rmit.edu.au
Date: 10th June 2020

I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honor code by typing "Yes": *Yes*.

Table of Contents

Introduction

There has been an increasing use and advancements of mobile devices and internet technology in recent years. Meanwhile the need for indoor localisation is increasing in everyday life. People have become reliant on using indoor navigation systems to find lifts, toilets and navigate through busy buildings. However indoor localisation brings challenges such as signal occlusion for some signalling methods such as satellite signals (Li et al., 2018). Therefore, indoor localisation requires methods which have more robust and localised signals. An example is the Bluetooth Low Energy (BLE) technology signalling. It is one of the most preferred methods of signalling due to its cost efficiency, low energy cost and how common it is in mobile devices (Li et al., 2018). It is associated with iBeacons technology which assess the Bluetooth received signal strength indication (RSSI). The RSSI works in such a way that -200 is out of range for the beacon, whilst a number closer to 0 means that the bluetooth signal is closer to the beacon (UCI Machine Learning Repository: BLE RSSI Dataset for Indoor localization and Navigation Data Set, 2020). Ideally, a machine learning model could be produced to classify a person's location indoors if it was fed with RSSI data from an array of iBeacons.

In this study, the data assessed are from the first floor of the of the Waldo Library, Western Michigan University;data were collected from the array of 13 iBeacons (UCI Machine Learning Repository: BLE RSSI Dataset for Indoor localization and Navigation Data Set, 2020). The layout is shown below **(Figure 1).** The general aim of this study is to produce a classification models (KNN and decision tree models) that can classify a location based on the RSSI signal data from the 13 beacons.

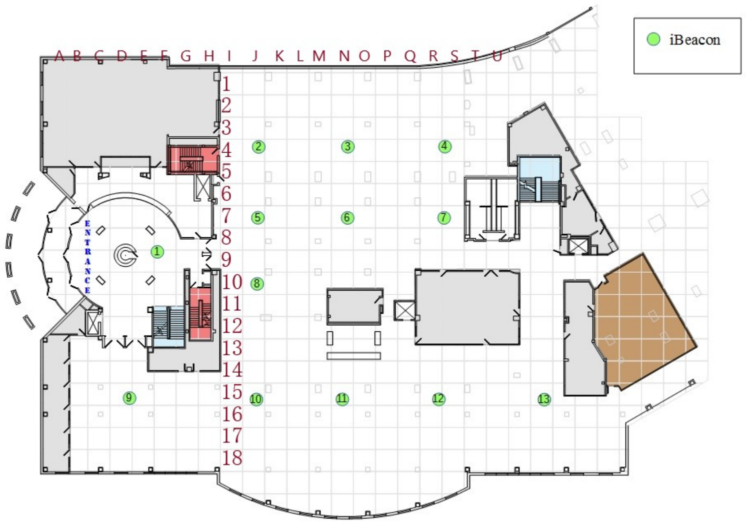


Figure 1.) The map of the 13 iBeacon array within the first floor of the of the Waldo Library, Western Michigan University, and showing coordinates. (UCI Machine Learning Repository: BLE RSSI Dataset for Indoor localization and Navigation Data Set, 2020).

Method

Data Preparation and Feature Selection

The dataset provided for this study is the BLE RSSI dataset. The labelled BLE RSSI data set was imported into the iPython Jupyter notebook environment as a CSV file, separated by commas, skipped the first row and heading = None (Clements, 2020). It was imported as First, headings were replaced with the headings from the CSV file, including 'Location', 'Date', and the beacon names. Then the data types of each column where checked using the .info() function (Pathak, 2018). Then I made a copy of the original BLE RSSI data frame for data preparation, calling it BLE_RSSI_copy. Given, that all the RSSI values should only range between 0 and -200, a sanity check was performed to see if there were any values greater than 0 or less than -200. None were found. There were no missing values or whitespaces detected. The BLE_RSSI_copy was reserved for machine learning modelling. Also, another made a copy of BLE_RSSI_copy which is called BLE_RSSI_filtered, was made for visualisations. Since this dataset mostly consisted of -200 values, all -200 values were removed. Otherwise, the -200 values would obscure any within range values in the visualisations.

All 13 iBeacons were selected as features for machine modelling. The only column that was not considered for machine learning models is the date column, since the time and date aren't considered in the research question. The research question is not
Also, in preparation for decision tree and KNN modelling, I split the BLE_RSSI_copy into the target and data. The target (y labels) consisted only of the 'Location' column of the BLE RSSI data set and dropped all other columns. The data (x data) consisted of all 13 beacons' data, with the 'Location' and the 'Date' columns dropped. Also,the target names were derived from converting the target (y labels) into a list via the set method (Kumar, 2018).

Data Exploration

Both the full dataset (BLE_RSSI_copy) and the filtered dataset(BLE_RSSI_filtered) have had their summary statistics assessed using the .describe() function. In addition, each beacon in the BLE_RSSI_filtered data frame were graphed as histograms to show how many instances of each signal strength occurred. In addition, a scatter matrix was produced comparing each beacon's data to each other. This was to determine if a signal close to one beacon was close to another. Some graphs were chosen to be graphed as individual scatter plots to highlight the most significant relationships (Arslan and Pathak, 2014).

Training and Validating KNN and Decision Tree models

Initially, chosen the train test split to be 0.2 test and 0.8 train and random state= 0. Also, a smaller test size was not chosen as the BLE RSSI data are very unbalanced. This was also to ensure that any variability in accuracy, precision and f1-score were coming from tuning the parameters, not the changing combination of the data in the train test split. Then after importing the relevant packages, both default settings for the decision tree and KNN classifier were run initially to assess baseline accuracy and performance. For each variation of the models including models produced in, a confusion matrix and a classification report were generated.

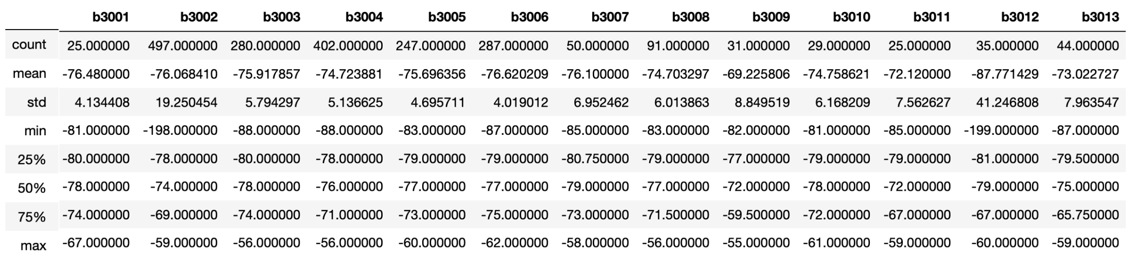
After assessing the accuracy of the default decision tree classifier in the model, parameters were manually tuned. The default settings in the decision tree classifier are class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None,min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best'. Although decision tree has a number of parameters, due to the low (non -200) value counts on some of the locations, the only parameters that were tuned were min_samples_split, criterion and max_depth. Min_samples_leaf could not be altered as some locations like L09 had only 2 non -200 RSSI values, which meant that after train test split, min_samples_leaf could only be 1 at best for those locations. In addition, the max_features considered was not altered, because the aim of the model is to predict location based on data from all 13 features, the iBeacons.

First, criterion was assessed and the accuracy was compared to the default decision tree classifier. Then the once the apporpirate criterion was chosen, models with that criterion and min_sample_split between 1-4 were assessed for accuracy. Min_samples_split wasn't a high value due to the low number of non -200 observations in some locations. Then once the model with the greatest accuracy and optimal critetion and min_sample_split, max_depth was assessed. This is because the decision tree that initially was generated for the default decision tree classifier was very deep. So max_depth was assessed in intervals of 5 between 20 to 40, and in intervals of 1 once any increases in accuracy occurred.

After assessing the accuracy of the default KNN classifier (k=5, weights='uniform', p =2) (scikit-learn 0.23.1 documentation, 2007), the influence of weight was assessed. Weights was changed to distance to assess if the accuracy increased in comparison to the accuracy obtained from the default KNN classifier settings. Then assessing which p value would give the greater accuracy according compared to the default setting and the model with the altered weights metric. Then changing the k-value ranging from 3 to 15 to determine which model has the greatest accuracy.

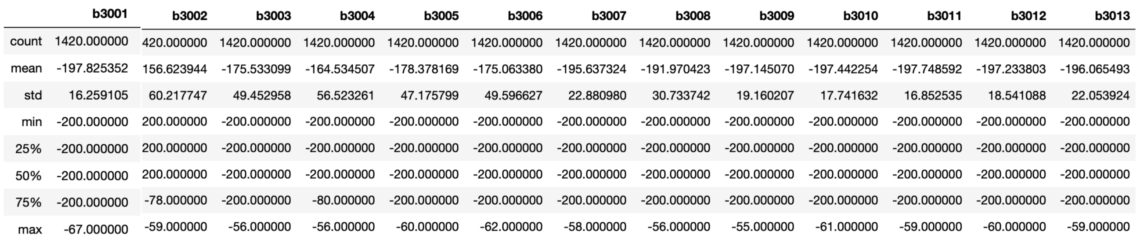
Once the most optimal parameters have been chosen for each KNN and decision tree models. Each model was validated by undergoing K folds cross validation with k folds = 10, random state= 4 and shuffle=True. This is to assess if the models would predict well to different combination of train test data. An accuracy score was produced for each fold and an average was calculated for each model to compare the models.

Results



	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
count	25.000000	497.000000	280.000000	402.000000	247.000000	287.000000	50.000000	91.000000	31.000000	29.000000	25.000000	35.000000	44.000000
mean	-76.480000	-76.068410	-75.917857	-74.723881	-75.696356	-76.620209	-76.100000	-74.703297	-69.225806	-74.758621	-72.120000	-87.771429	-73.022727
std	4.134408	19.250454	5.794297	5.136625	4.695711	4.019012	6.952462	6.013863	8.849519	6.168209	7.562627	41.246808	7.963547
min	-81.000000	-198.000000	-88.000000	-88.000000	-83.000000	-87.000000	-85.000000	-83.000000	-82.000000	-81.000000	-85.000000	-199.000000	-87.000000
25%	-80.000000	-78.000000	-80.000000	-78.000000	-79.000000	-79.000000	-80.750000	-79.000000	-77.000000	-79.000000	-79.000000	-81.000000	-79.500000
50%	-78.000000	-74.000000	-78.000000	-76.000000	-77.000000	-77.000000	-79.000000	-77.000000	-72.000000	-78.000000	-72.000000	-79.000000	-75.000000
75%	-74.000000	-69.000000	-74.000000	-71.000000	-73.000000	-75.000000	-73.000000	-71.500000	-59.500000	-72.000000	-67.000000	-67.000000	-65.750000
max	-67.000000	-59.000000	-56.000000	-56.000000	-60.000000	-62.000000	-58.000000	-56.000000	-55.000000	-61.000000	-59.000000	-60.000000	-59.000000

Figure 2.) Summary statistics for all the beacons with the -200 values removed (BLE_RSSI_filtered)



	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
count	1420.000000	420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000
mean	-197.825352	156.623944	-175.533099	-164.534507	-178.378169	-175.063380	-195.637324	-191.970423	-197.145070	-197.442254	-197.748592	-197.233803	-196.065493
std	16.259105	60.217747	49.452958	56.523261	47.175799	49.596627	22.880980	30.733742	19.160207	17.741632	16.852535	18.541088	22.053924
min	-200.000000	200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
25%	-200.000000	200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
50%	-200.000000	200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
75%	-200.000000	-78.000000	-200.000000	-80.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000	-200.000000
max	-67.000000	-59.000000	-56.000000	-56.000000	-60.000000	-62.000000	-58.000000	-56.000000	-55.000000	-61.000000	-59.000000	-60.000000	-59.000000

Figure 3.) Summary statistics for all the beacons in the BLE RSSI dataset

The above figures show that the majority of the observations in the BLE RSSI dataset consisted of out of range -200 values, with mean signals varying from -197.825 to -156.62 with large variations in standard deviation **(Figure 3).** However, in the filtered dataset **(Figure 2),** the average RSSI signal between beacons was fairly consistent with relatively lower standard deviations. Mean RSSI ranged between -69.23 and -76.62 and generally standard deviation was less than 8.85. The exception of beacon 12 (mean= -87.77, sd= 41.24). In addition not all beacons have equal number of within range RSSI signals. For example, beacons 1 and 11 have only 25 within range RSSI signals each, whereas beacon 4 has as much as 402 within range RSSI signals. The frequencies of each RSSI signal in each beacon are represented in Figure 4. With most of the beacons most of the within range RSSI signals fall between -50 and -90 with the exception of beacon 12 **(Figure 4 L.)**

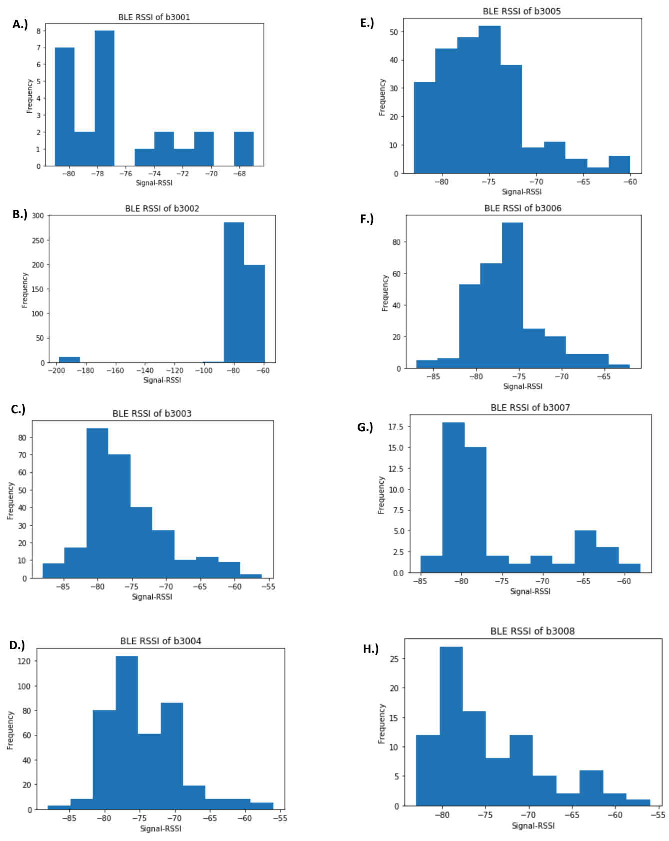


Figure 4 A-H .) Histogram plots of the RSSI signals that each beacons (beacons 1 to 8) receives (excluding the out of range readings).

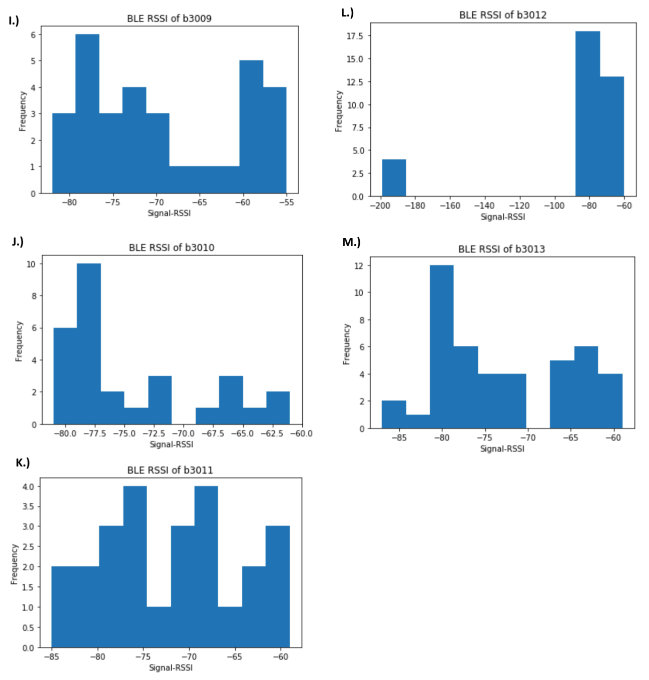


Figure 4 I-M .) Histogram plots of the RSSI signals that each beacons (beacons 9 to 13) receives (excluding the out of range readings).

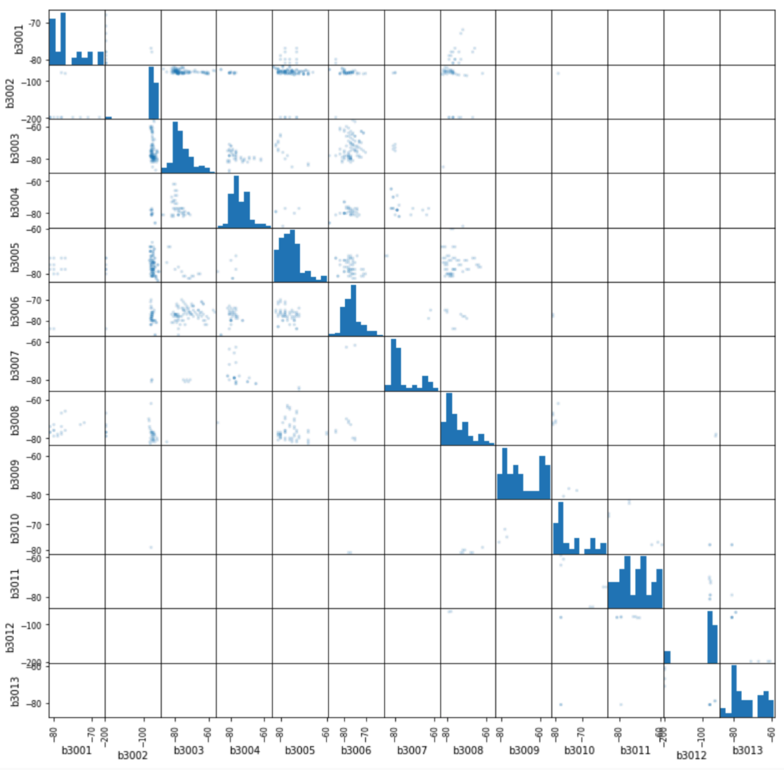


Figure 5.) Scatter matrix of all the within range RSSI signals of all 13 beacons compared to each other.

The RSSI signals of each beacon were compared to each other in order in order to explore the relationship of each beacon with each other **(Figure 5)**. Overall, it was hypothesised that the beacons closest to each other would be more likely to share more common RSSI signals and clusters of signals than beacons far away from each other. In figure 5 there are almost no common signals between beacons that are far from each other (eg: beacon 1 & 13) and the most of the RSSIS signals are clustered around beacons that are closer to the middle of the floor (beacons 2-8).

Some significant relationships between specific pairs of beacons and their RSSI signals are highlighted in **figure 6.** Some pairs of beacons showed potential relationships between RSSI signals that were close to beacon 8 were also relatively close to beacon 1**(Figure 6B).** Also, most signals that are very close to beacon 2 are within range for beacon 5, but some values that are almost out of range for beacon 2 are still within range for beacon 5 **(Figure 6C).** Similarly, RSSI signals that are very close to beacon 2 are fairly close to beacon 3 **(figure 6D).** Also beacon 2 RSSI signals are within range for beacon 6 **(Figure 6J).** Similar patterns occur between beacons 7 and 4, beacons 3 and 6 and 5 and 6 **(Figure 6 F-H).** Also in figure 6E, RSSI signal readings that were close for beacon 10 were a bit further away from beacon 11.

In contrast, some pairs of beacons showed little to no relationship between their RSSI readings. In **figure 6A** beacons 1 and 9 were hypothesised to at least share common RSSI readings, and if one was close to beacon 1 they would be within range for beacon 9. However, in **figure 6A**, there were not readings detected, indicating that there weren't any common RSSI signals. Also, in **figure 6I**, RSSI signals that were close to beacon 13 were almost out of range for beacon 12.

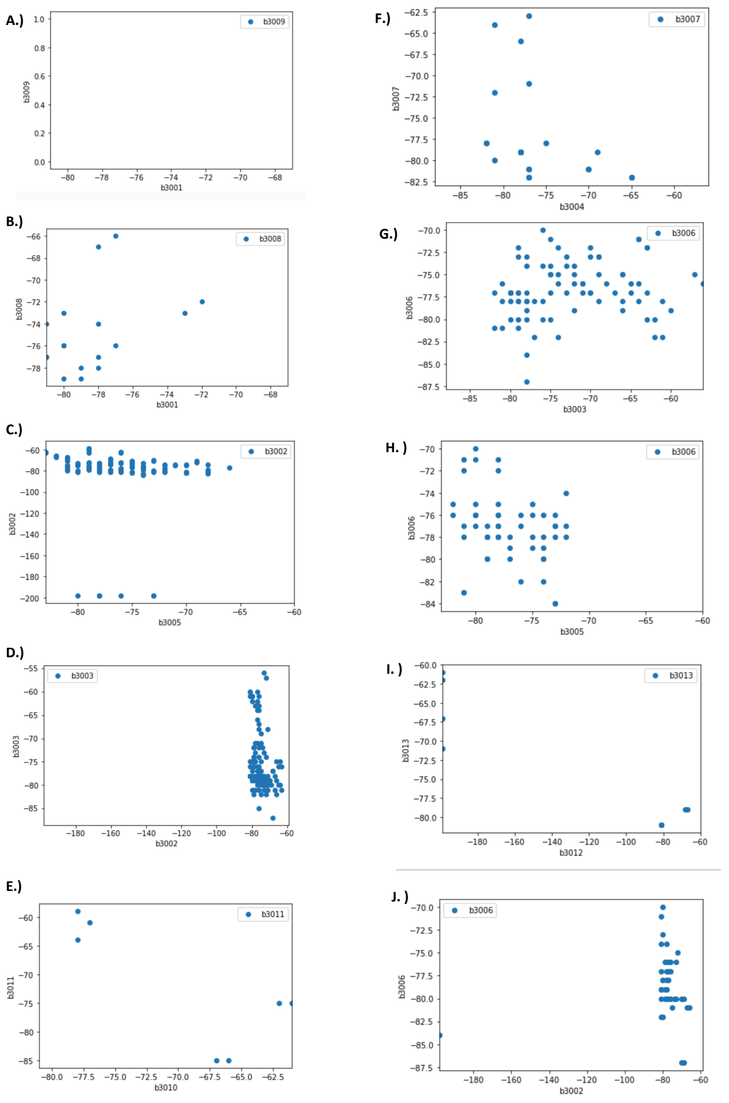
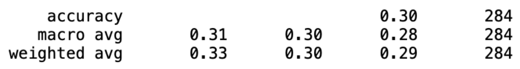


Figure 6.) Scatter matrixes comparing BLE RSSI signals of pairs of iBeacons: **A.)** beacons 1 vs. 9, **B.)** beacons 1 vs 8, **C.)** beacons 2 vs. 5, **D.)** beacons 2 vs. 5, **E.)** beacons 11 vs 10, **F.)** beacons 4 vs 7, **G.)** beacons 3 vs 6, **H.)** beacons 12 vs 13, **J.)** beacons 6 vs 2.

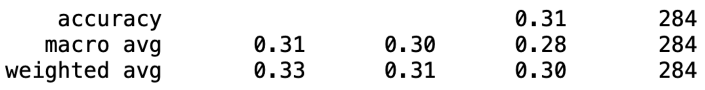
In the modelling stage of the experiment, two classification models, one decision tree and KNN models were produced. In figure 7 a decision tree model was first tested with default parameters. This induced an accuracy of 0.31. Also, that decision tree was plotted. It had a very deep decision tree. The min_sample_split, max_depth and criterion were tuned in an attempt to gain an increase in accuracy, f1_score. However, the accuracy gained from the tuned decision trees (max_depth =25) was very similar to the default decision tree classifier (0.31).

A.)



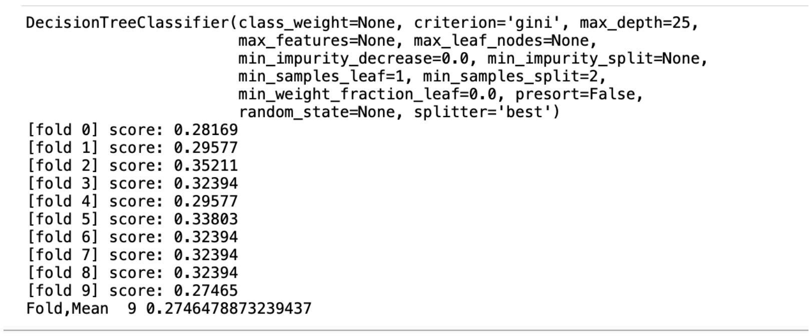
```
     accuracy                           0.30       284
    macro avg       0.31      0.30      0.28       284
 weighted avg       0.33      0.30      0.29       284
```

B.)



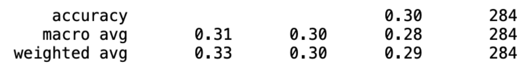
```
     accuracy                           0.31       284
    macro avg       0.31      0.30      0.28       284
 weighted avg       0.33      0.31      0.30       284
```

C.)



```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
[fold 0] score: 0.28169
[fold 1] score: 0.29577
[fold 2] score: 0.35211
[fold 3] score: 0.32394
[fold 4] score: 0.29577
[fold 5] score: 0.33803
[fold 6] score: 0.32394
[fold 7] score: 0.32394
[fold 8] score: 0.32394
[fold 9] score: 0.27465
Fold,Mean  9 0.2746478873239437
```

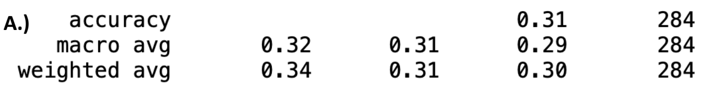
Figure 7.) Classification reports of Decision tree models with test = 0.2 and random state =0 **A.)** Decision tree w
tree with tuned parameter
fold and average between 1



```
     accuracy                           0.30       284
    macro avg       0.31      0.30      0.28       284
 weighted avg       0.33      0.30      0.29       284
```

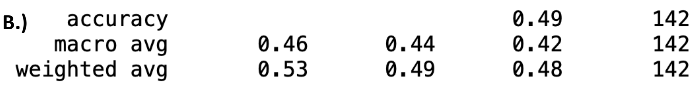
In figure 7 a decision tree model was first tested with default parameters. This induced an accuracy of 0.30. Also, that decision tree was plotted. It had a very deep decision tree. The min_sample_split, max_depth and criterion were tuned in an attempt to gain an increase in accuracy, f1_score. However, the accuracy gained from the tuned decision trees (max_depth =25) was very similar to the default decision tree classifier **(Figure 7A & B).** max depth of less than 25 had decreased accuracy and potentially underfitted the model. Therefore, the decision three that had been tuned was too similar to the original default decision tree classifier. In addition both decision trees have made the same best splits, with beacon 2 as the feature for the best initial split of the data.

A.)



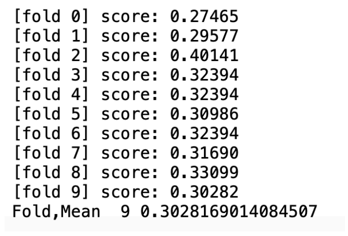
```
     accuracy                           0.31       284
    macro avg       0.32      0.31      0.29       284
 weighted avg       0.34      0.31      0.30       284
```

B.)



```
     accuracy                           0.49       142
    macro avg       0.46      0.44      0.42       142
 weighted avg       0.53      0.49      0.48       142
```

C.)



```
[fold 0] score: 0.27465
[fold 1] score: 0.29577
[fold 2] score: 0.40141
[fold 3] score: 0.32394
[fold 4] score: 0.32394
[fold 5] score: 0.30986
[fold 6] score: 0.32394
[fold 7] score: 0.31690
[fold 8] score: 0.33099
[fold 9] score: 0.30282
Fold,Mean  9 0.3028169014084507
```

Figure 8.) Classification reports of KNN models with test = 0.2 and random state =0 **A.)** KNN classifier with default parameters and **B.)** KNN classifier with tuned parameters **C.)** K folds cross-validation accuracy readings for each fold and average between 10 folds with tuned parameters.

In figure 8, the K folds validation and the classification report are shown for the tuned parameters (Figure 8 C& D) in comparison to the default parameters (Figure 8 A and B). The accuracy of 0.31 was obtained from the KNN classifier with default parameters. After tuning parameters and comparing changes in accuracy, f1-score and in the confusion matrix (refer to s3846487 Jupyter notebook), the parameters with the highest accuracy were chosen.

The optimal KNN model was chosen with the parameters k=5, weights = 'distance' , p =2 which increased accuracy to 0.49 and the interpreted the confusion matrix to have more correctly predicted instances.

Then after selecting the models, both the tuned decision tree and KNN models underwent K fold cross validation. Both the decision tree and the KNN models have similar K folds cross validation scores (Decision tree = 0.3099 vs KNN=0.3028).

Discussion

In this study, both decision tree and KNN classification models were produced. The advantage of the decision tree is that it is able to identify the most important features to consider the BLE RSSI data and the BLE RSSI signal value to split the data. In the case of both the default and the tuned decision trees, the first split was determined regarding if the readings were associated with iBeacon 2. Then the next level of splits were between the 13th beacon and the 5th beacon. Considering that beacon 2 has the largest number of non -200 data, it makes sense that this was the most important split to define homogeneity. After classifying if these signals came from beacon 2 values <-75, it determined if signals came from either the 13th beacon or the 5th beacon. This would determine if the beacon signals came from different sides of the floor. Furthermore, according to **figure 5** it appears that most of the BLE RSSI signals correspond to the entrance and the middle of the room (beacons 1-8). Very few RSSI signals were obtained from beacons 9,10,11,12 & 13 that were at the right side of the floor (from the perspective of entering the library). This indicates that most of the traffic comes from the entrance and the middle of the library floor.

However, there were flaws in the modelling and parameter tuning process for both decision tree and KNN models. For decision tree, it wouldn't have been appropriate to try to tune some parameters that would normally reduce the overfitting, like min_samples_leaf. This is because a number of the location labels do not have many non -200 data (minimum of 2 instances), so the min_samples_leaf had to remain =1. Max_features was not tuned because the goal of this machine learning model was to be able to classify the location based on data gained from all 13 beacons.

Firstly, the suitable criterion was assessed based on which algorithm would give the highest accuracy. However, in this case, both gini index and entropy provided similar accuracy and f1-score readings. Considering that the research question is trying to classify location based on readings from each beacon, gini was chosen, as it would be able to make binary splits trying to decipher if a reading was associated with a location. In addition, there were attempts to limit the extent of overfitting by min_samples_split and max_depth. By attempting to increase the min_samples_split, the accuracy actually decreased. This is due to having a limited number of non -200 data in some of the location labels. Also, attempts to reduce the max_depth also resulted in underfitting the model as the accuracy and f1_scores decreased. Although the tuned decision tree had a max_depth of 25, this was also because the accuracy decreased with smaller max_depth values, which indicate underfitting. However, a max_depth of 25 meant that the tuned decision tree plotted would allow the same depth as the decision tree plotted for the be nearly the same as the tree produced by the default parameters. Therefore, its' unsurprising that both decision trees have the same accuracy and f1-scores.

In the KNN model, the parameters, k-nearest neighbors, weights and the p value were tuned comparing their accuracy and f1_scores to the default KNN model (accuracy = 0.31). The final parameters chosen were k =5, weights= 'distance' and p=2. 'Distance' was chosen to be the weight since not only it increased the accuracy and f1_score but also the closes neighbours will have a greater influence on classification. Since RSSI signals give an indication of location via the distance away from the beacons, the signals that are closer to a location would be given greater weight than signals furthest away from a location. Also p =2 was chosen with the default metric of the Minkowski distance, which is equivalent to the Eucledian distance. Euclidean distance is a straight line measurement between two points, which are necessary to define the location based on the straight line distance of signals between each other. The number of k nearest neighbours to be considered in classification

Overall, both the decision tree and the KNN models developed performed very poorly to the provided labelled BLE RSSI data set. Both of these models had similar average cross validation scores, with decision tree having an average accuracy of 0.3098 and the KNN model an average cross validation score of 0.3028. Both models have very similar average cross validation scores, which means that they both perform equally poorly. This is potentially due to the limitations of working with a small and very unbalanced dataset.

Within this labelled BLE RSSI dataset it only covered seven days' worth of data, and only a few hours each day. The number of non -200 for any location varied from 2-34. Given the high variability, there needs to be at least a month's worth of labelled data for decision tree and KNN to classify locations based on the RSSI signals.

The majority of the issues come from the lack of non -200 RSSI signals in each of the locations. Some locations such as L09 which only have 2 non -200 instances of the RSSI signals. Therefore, it was difficult to train test split the data for a number of the locations. Often this error *" Encountered the Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. 'precision', 'predicted', average, warn_for,"* occurred in for a number of labels. This indicated that the model has not made any predictions for certain labels. This means that the predicted labels made to classify the location based on BLE RSSI signals were completely incorrect for certain locations, considering that precision =0.

Theoretically, the locations with low numbers of non -200 could have been removed and that could have resolved the issue. However, even just removing locations with less than 10 non -200 values would have 34 locations for the machine learning model to consider. However, this would have defeated the purpose of producing classification models that would be able to predict for all locations of the Waldo Library. Also, it is a possibility to oversample the underrepresented locations to make up for their low count of within range RSSI signals to counter the unbalanced dataset, in future studies. In addition, there are potentially other machine learning models which may be more suitable for classifying the BLE RSSI but in this study only two models were assessed. Also, the research question may need to have been altered to possibly suit an unsupervised learning approach. Clustering techniques such as K means and DBSCAN could identify RSSI signals to be similar in some way. Potentially it could cluster RSSI signals that are similar to each other and potentially identify their possible locations by how far they are from beacons. In these methods, there is no need to provide labelled data and this may negate the issue of having few within range signals for each location.

Overall, given the cross-validation scores of the decision tree and KNN models developed, they would not be able to generalise well to unseen data. Not only are a number of locations that were not predicted, or predicted correctly according to the confusion matrixes produced and their f1 and precision scores. If accurate classification models were to be developed to generalise to unseen data, then it would require either more labelled data or a different approach would be necessary.

Conclusion

In conclusion, it is possible to identify the locations based on BLE RSSI signals, but more data are required to classify based on decision tree and KNN models. The decision tree and the KNN models that were developed did not perform well to accurately   With few within range data jthat so few within range data, more BLE RSSI data are require, to classify.

References

Any material that isn't explicitly stated in either the Jupyter iPython notebook, presentation or the report is from the Practical Data Science with Python learning materials.

Arslan, D. and pathak, s., 2014. How To Plot Two Columns Of A Pandas Data Frame Using Points?. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/17812978/how-to-plot-two-columns-of-a-pandas-data-frame-using-points> [Accessed 10 June 2020].

Clements, J., 2020. *How To Ignore The First Line Of Data When Processing CSV Data?*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/11349333/how-to-ignore-the-first-line-of-data-when-processing-csv-data> [Accessed 8 June 2020].

Stack Overflow. 2017. How To Stop Jupyter Outputting Truncated Results When Using Pd.Series.Value_Counts()?. [online] Available at: <https://stackoverflow.com/questions/43909817/how-to-stop-jupyter-outputting-truncated-results-when-using-pd-series-value-coun> [Accessed 10 June 2020].

Kumar, K., 2018. *How To Get Only Distinct Values From A List?*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/52681747/how-to-get-only-distinct-values-from-a-list/52681933> [Accessed 8 June 2020].

Li, G., Geng, E., Ye, Z., Xu, Y., Lin, J. and Pang, Y., 2018. Indoor Positioning Algorithm Based on the Improved RSSI Distance Model. *Sensors*, 18(9), p.2820.

Pathak, M., 2018. (Tutorial) Handling Categorical Data In Python. [online] DataCamp Community. Available at: <https://www.datacamp.com/community/tutorials/categorical-data?fbclid=IwAR36wHr5Wr9NIcWXFBjI9Bev1ZsO1hF81entMp7U9sfZ_nvRp9IMLs2Sttw> [Accessed 8 June 2020].

Scikit-learn.org. 2007. *Sklearn.Neighbors.Kneighborsclassifier — Scikit-Learn 0.23.1 Documentation*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> [Accessed 8 June 2020].

Archive.ics.uci.edu. 2020. UCI Machine Learning Repository: BLE RSSI Dataset For Indoor Localization And Navigation Data Set. [online] Available at: <https://archive.ics.uci.edu/ml/datasets/BLE+RSSI+Dataset+for+Indoor+localization+and+Navigation> [Accessed 8 June 2020].