Practical Data Science with Python Assignment 2

Student ID: s3602584
Student Name: Peter Moorhead

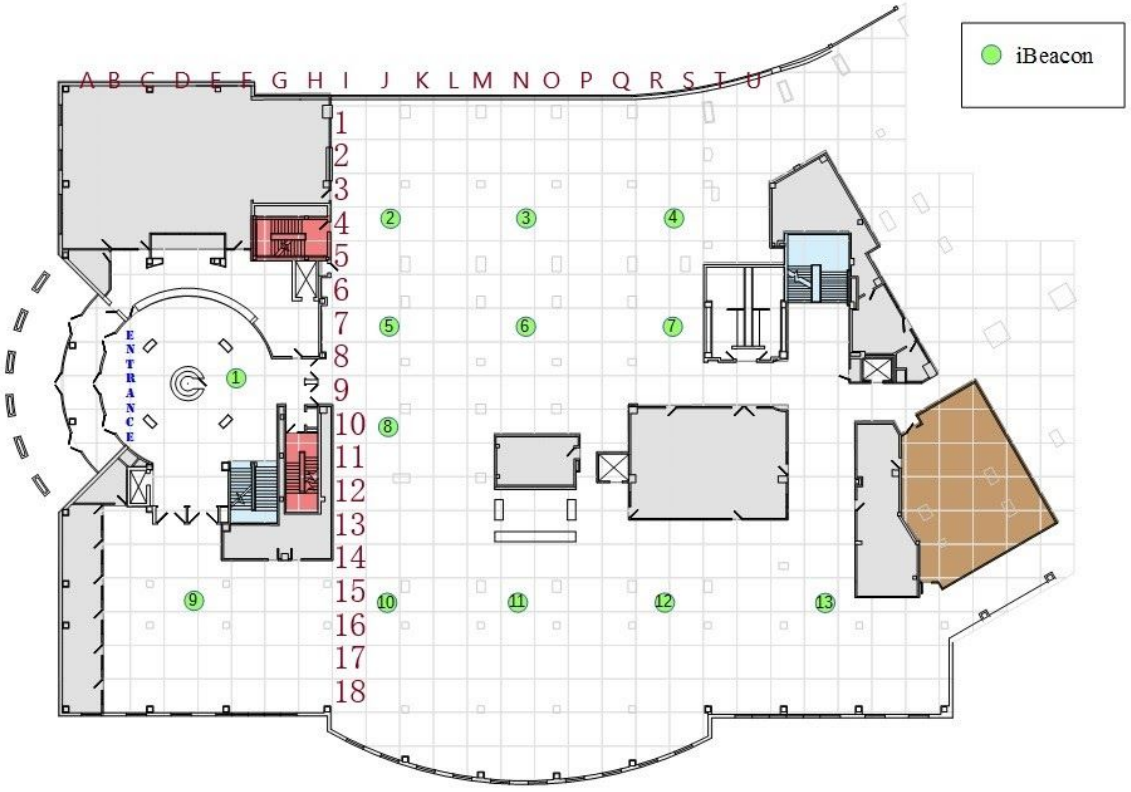
I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honor code by typing "Yes": Yes.

Author Information: Peter Moorhead RMIT Student 2020

Affiliations: RMIT University

email: s3602584@student.rmit.edu.au

Date of report: 06/06/2020



Practical Data Science with Python Assignment 2 **1**
Abstract 2
Introduction 2
Methodology 3
Preprocessing 3
Exploration 4
Hypotheses 6
Model building 7
Results 7
Base accuracy 7
Hypertuning - Grid Search 8
Feature tuning - Hill climbing 9
Hypertuning Results Table 10
Decision Tree 11
Discussion 11
Conclusion 12
References 12

Abstract

This report aimed to research and construct the best possible classifier from the RSSI data obtained from 13 iBeacons around Waldo Library in Western Michigan, so as to create the base information for which a navigation app could be made. The data was cleaned for any possible errors before being analysed through tables and charts to find any interesting or useful relationships between the data columns. This data was then fed as features to both the K Nearest Neighbour classifier as well as a Decision Tree classifier to discover which would be able to give a more accurate location for the hypothetical user of the app. These classifiers were further tuned by using the Hill Climbing and Grid Search algorithms to try and improve their accuracy scores. We found that while both classifiers were able to produce some accuracy, the K Nearest Neighbour algorithm was best suited for this particular data set. This report will take the reader through the process of obtaining, preparing, analysing data as well as building a suitable classifier to succeed in the first step towards a navigation app.

Introduction

This report aims to explore the localization data from the Waldo Library in Western Michigan University and use various combinations of algorithms to build the best classifier for locating individuals inside the library. The data comes from 13 iBeacons situated around the first floor of the library in a grid that allowed an iPhone 6S to capture RSSI readings off of these beacons. The RSSI readings come in the form of a series of negative integers ranging from around -60 to -200 where -60 means that beacon is close to the phone and -200 means the

beacon is out of range of the phone. The purpose for this research is to try and build a localisation classifier for which a navigation app can be built upon so that library users can find resources they need without having to ask librarians or look up a map. This report analyses the beacon data and where the iPhone was in relation to that data and compares the correlation between them and uses those findings to build various classifier models to figure out the most effective localization model.

Methodology

Preprocessing

To begin the data was imported to a jupyter notebook and preprocessed using the pandas and NumPy python tools to try and find any discrepancies or errors that may make the classifiers less accurate than they could be. To begin the 'head' command was used to check that the data was imported into the correct columns and that there weren't any import errors between the csv file and the jupyter notebook.

```
labeledData.head(5)
```

	location	date	b3001	b3002	b3003	b3004	b3005	b3006	b3007	b3008	b3009	b3010	b3011	b3012	b3013
0	O02	10-18-2016 11:15:21	-200	-200	-200	-200	-200	-78	-200	-200	-200	-200	-200	-200	-200
1	P01	10-18-2016 11:15:19	-200	-200	-200	-200	-200	-78	-200	-200	-200	-200	-200	-200	-200
2	P01	10-18-2016 11:15:17	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
3	P01	10-18-2016 11:15:15	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200
4	P01	10-18-2016 11:15:13	-200	-200	-200	-200	-200	-77	-200	-200	-200	-200	-200	-200	-200

'Dtypes' was then used on the dataframe to check that the data successfully kept their appropriate types during the import and was the first check for impossible values for the beacon data.

```
#Looking for mistakes in labled data
labeledData.dtypes
```

```
location    object
date        object
b3001        int64
b3002        int64
b3003        int64
b3004        int64
b3005        int64
b3006        int64
b3007        int64
b3008        int64
b3009        int64
b3010        int64
b3011        int64
b3012        int64
b3013        int64
dtype: object
```

Typos, extra whitespace, upper/lower case discrepancies and impossible values were further checked for as each column was checked with the 'value_counts' command to make sure that all the data for the beacons was appropriate and made sense.

During the checks on the location column it was discovered that there were values for 'V15' and 'W15' which were not coordinates on the map. Unsure whether the map was inaccurate or the data, the offending rows were taken as impossible values and dropped from the dataset.

```
labeledData['b3001'].value_counts()
```

```
-200    1395
-78        5
-80        4
-81        3
-77        3
-79        2
-75        1
-74        1
-73        1
-72        1
-71        1
-70        1
-68        1
-67        1
```

```
labeledData.loc[labeledData['location'] == 'V15'].shape
```

```
(8, 15)
```

```
labeledData.loc[labeledData['location'] == 'W15'].shape
```

```
(17, 15)
```

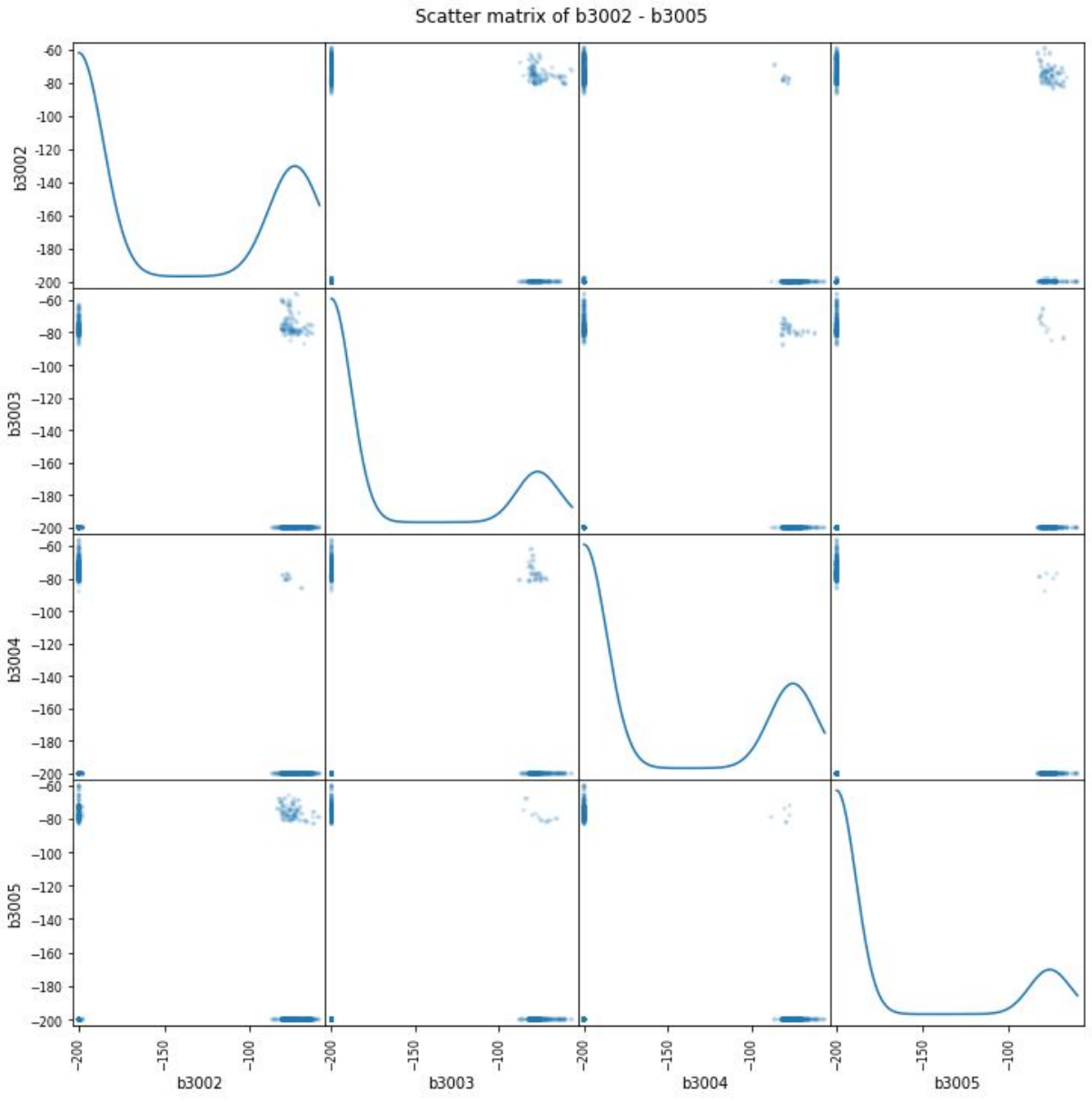
Finally the ‘describe’ command was used across all columns so as to check for outliers. This table also showed that the -200 was throwing off the standard deviation, the source of the data expressed that there were missing values in the data but none were found, it was now obvious that the ‘-200’ was to be interpreted as missing values. During the model building process the report will explain how the missing data was handled.

```
labeledData.describe()
```

	b3001	b3002	b3003	b3004	b3005	b3006
count	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
std	16.259105	60.217747	49.452958	56.523261	47.175799	49.596627
min	-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
50%	-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
max	-67.000000	-59.000000	-56.000000	-56.000000	-60.000000	-62.000000

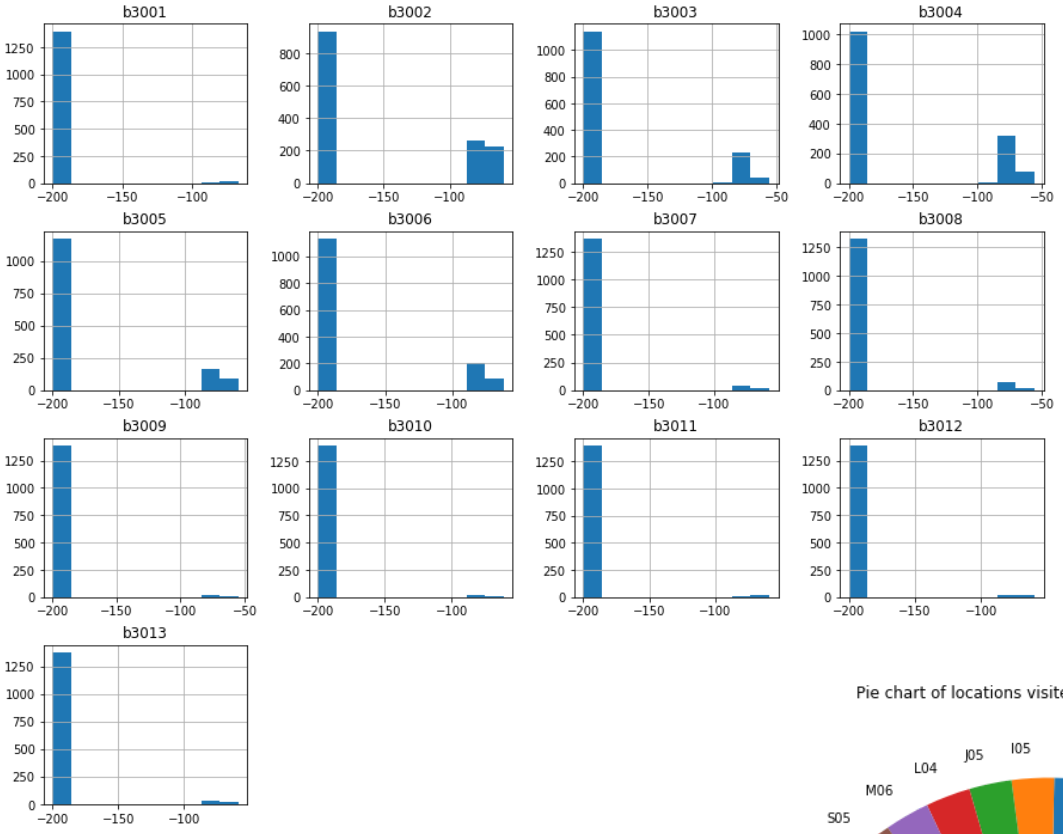
Exploration

The data was explored initially individually with ‘value_counts’ and ‘describe’ commands in the preprocessing section. During this next phase the data was explored visually through scatter matrices and histograms to gain a more thorough understanding of what the data meant and how it could be used efficiently and effectively to build a stronger classifier.

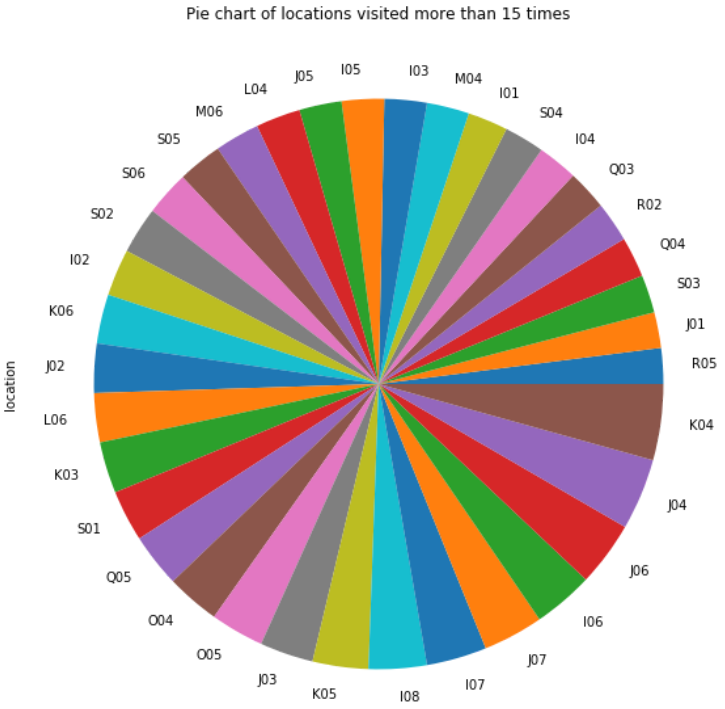


Above we can see the first visualisation, a scatter matrix of the most active beacons, b3002 to b3006 (6 not included). This diagram explores the relationships between each of these beacons and also includes a kernel density estimation in the diagonal. These beacons are obviously going to have a strong relationship as they are all situated in the same general vicinity towards the top of the given map (pg 1.) and would likely all ping the iPhone simultaneously with varying strengths. This is supported by the two diagrams below.
Firstly the histogram plots showing the activity of all the beacons we can see again that the most active are beacons 2 - 6 which demonstrates their relationship with the location column of the dataset shown in the pie chart further down. We saw that the grid places most visited were again the ones around the top of the map, situated near the most active beacon b3002. This relationship was obvious as the stronger your connection is to the beacon the closer you will be.

Histograms of each beacon

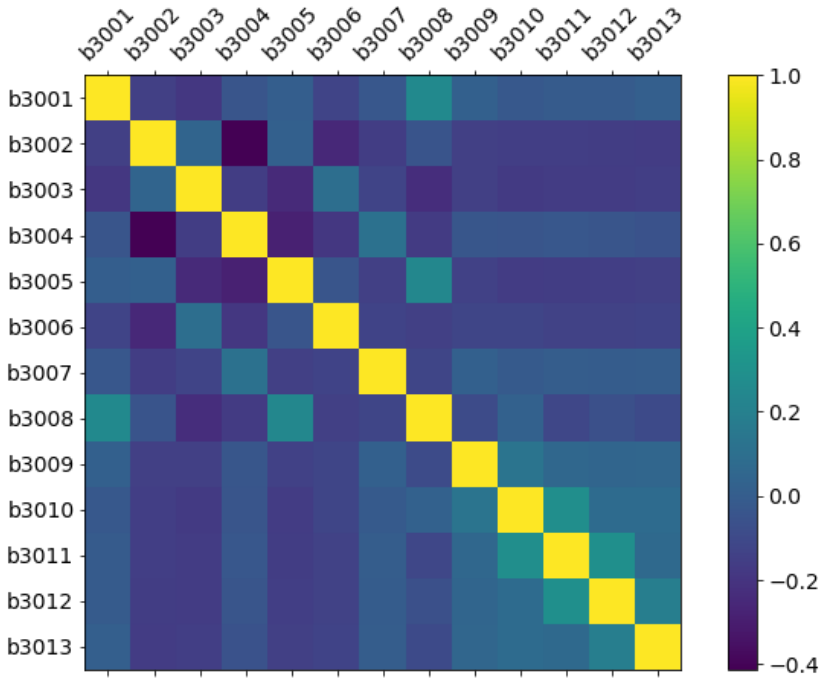


Beacons 1 and 7 - 13 were less active and this is mirrored in the location plot. The map shows beacons 9 - 13 were separated from the rest by a few grid squares. This is shown from the iPhone data as these beacons were much less active as well as depicted by the pie chart, there were no grid points that were visited more than 15 times below grid point 10 on the map.



Finally a correlation matrix was constructed for the beacons to further support the data showing that the closer the beacons are to one another the stronger their relationship will be to both each other and the location data. Knowing these relationship strengths will in turn provide for a stronger and more reliable data model to construct our classifiers on.

Beacon correlation matrix



Hypotheses

The previous figures have shown the various relationships between the column data and from this some hypotheses can be formed.
Firstly that the iPhone carrier visited the beacons towards the top of the map more frequently than the bottom, this may be because that is where the entrance to the library appears to be as well as a stairway, this is supported by the frequency in which the beacons closest to these areas have a high occurrence score.
Secondly we can see the exact opposite, where in the bottom half of the map, beacon 10 - 13 does not have nearly as much activity and therefore whoever conducted the data gathering did not explore this area much.
Thirdly we can see from the correlation matrix above that the areas are clearly grouped together without needing a map. Beacons 9 through 13 are all much more correlated with each other than with the lower numbered beacons.
Interestingly we see a negative correlation between beacons 2 and 4 which seems counter intuitive as those beacons are relatively close to each other geographically. This may be because of some interference for which we can not account for going by just the data we have.
Beacon 1 did not have as much traffic as I would have thought as it was located in the entrance of the building but had hardly any frequency, making it seem like whoever recorded

the data only started recording once they were already inside the library and not while they entered. This was checked against the location column and compared and this supported the theory as there were very few occurrences of the grid locations d through f where the beacon 1 may have pinged the phone. Other hypotheses were explored in the jupyter but were not featured here as they did not have much supporting evidence.

Model building

The beacon data was then copied and spliced into variables so different combinations of scaling algorithms could be applied and tested with and against the varying classifier models. The data was normalized so as to change the values in the beacon columns to a common scale without distorting differences in the ranges of values. Normalization was performed as it was specifically proven to improve the K Nearest Neighbour classifier. Unscaled, Standard scaled (left), and Min Max scalings(right) were all used in combination with each other to try and find the best type of data manipulation that would help construct the best localization classifier.

$$x' = \frac{x - \bar{x}}{\sigma} \qquad x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

The scaled and unscaled data was then split into training and testing sections with a 80% - 20% split respectively and the initial un-hypertuned data was fed to both the K nearest neighbour classifier and a decision tree classifier to get the base accuracy level which would be built upon and improved by more advanced processes such as using a combination of scaled data and the Grid search algorithm and Hill climbing to hypertune the given features to give a more accurate classifier. This included using k-folds cross validation to ensure over and under fitting was kept to a minimum.

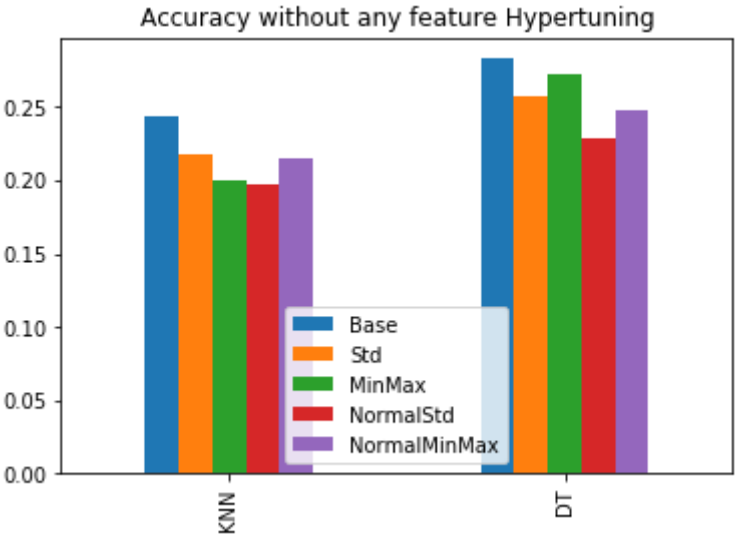
Results

Base accuracy

	KNN Accuracy	DT Accuracy
Base Score	0.24372759856630824	0.2831541218637993
Standard Scaler Score	0.21830985915492956	0.25806451612903225
Min_Max Scaler Score	0.2007168458781362	0.2724014336917563
Normalized Standard Scaled	0.1971326164874552	0.22939068100358423
Normalized MinMax Scaled	0.21505376344086022	0.24731182795698925

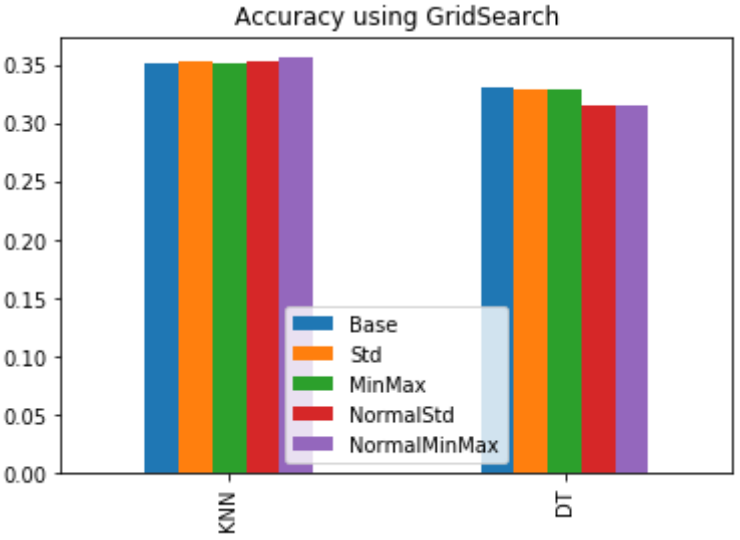
The above table shows the accuracy that the K nearest neighbour (KNN) and decision tree (DT) classifiers got using just differently scaled data without any hypertuning.

The graph to the right visually shows how without using any form of feature tuning the accuracy scores capped out at only 0.28% out of a possible 1%. Interestingly this was scored by the decision tree where no scaling was used, although the min_max scaling came in a close second at 0.27%. This initial accuracy left much to be desired and so more advanced measures were utilized.



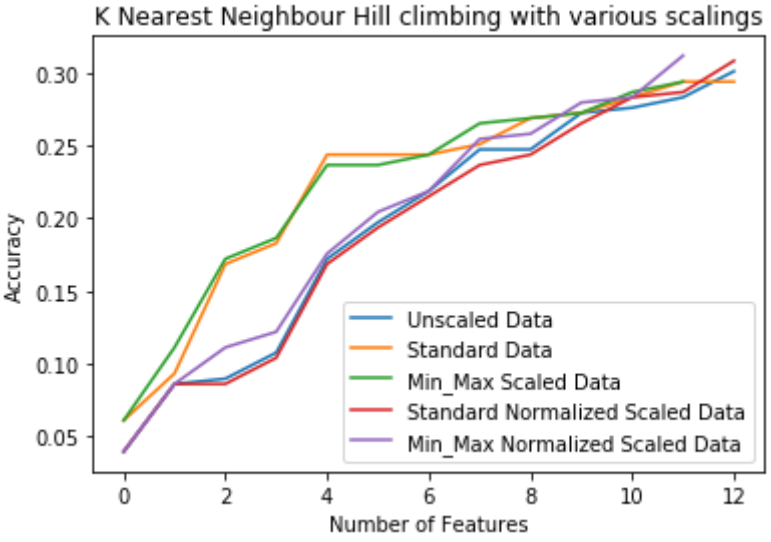
Hypertuning - Grid Search

Two feature tuning strategies were used, the GridSearch function from sklearn and hill climbing for feature selection. The former results are seen right where all the KNN results were more accurate than the DT which is opposite the base results as DT was overall more accurate than KNN. The number of neighbours ‘K’ and the power level ‘p’ for the Minkowski metric were chosen to be hypertuned for KNN as they had the greatest effect on the accuracy of the classifier. Fed as arrays of integers the GridSearch algorithm tests all combinations of the K array and the p array and generates the best scoring classifier. This analysis was performed on each type of scaling combination as in the first bar chart. The max depth and the max number of features were chosen to be tuned by GridSearch for DT and were fed in a similar way as an array of parameters to the function. During this process there were some issues with the stratification of the cross-fold number as some of the locations had less than the chosen cv count ‘10’ and therefore caused the data to be less accurate without further processing which may have led to some unseen over or under fitting.



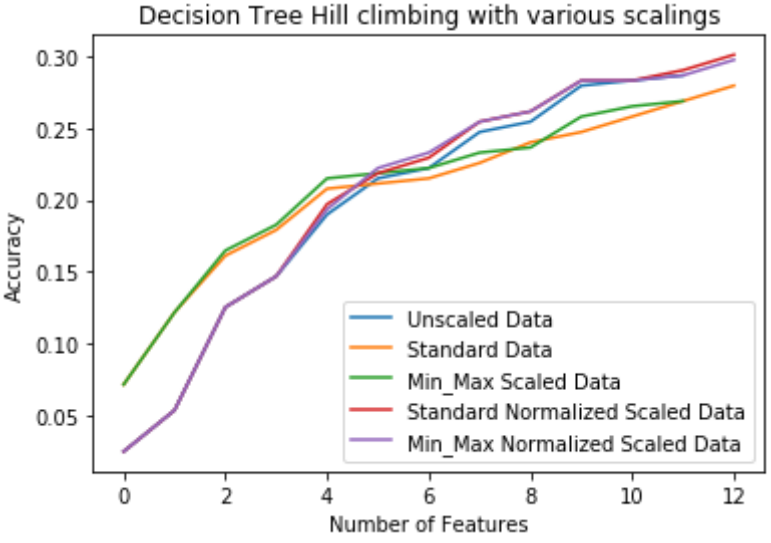
Feature tuning - Hill climbing

Next the classifiers and features were improved with the hill climbing algorithm, which tries to find the best features to use for a classifier. Seen right it initially improved the KNN standard and min max scaling data with 4 features being the most obvious spike in accuracy, but then all data converged to around .30% accuracy which while still better than unscaled and untuned data, was not near an optimal solution.

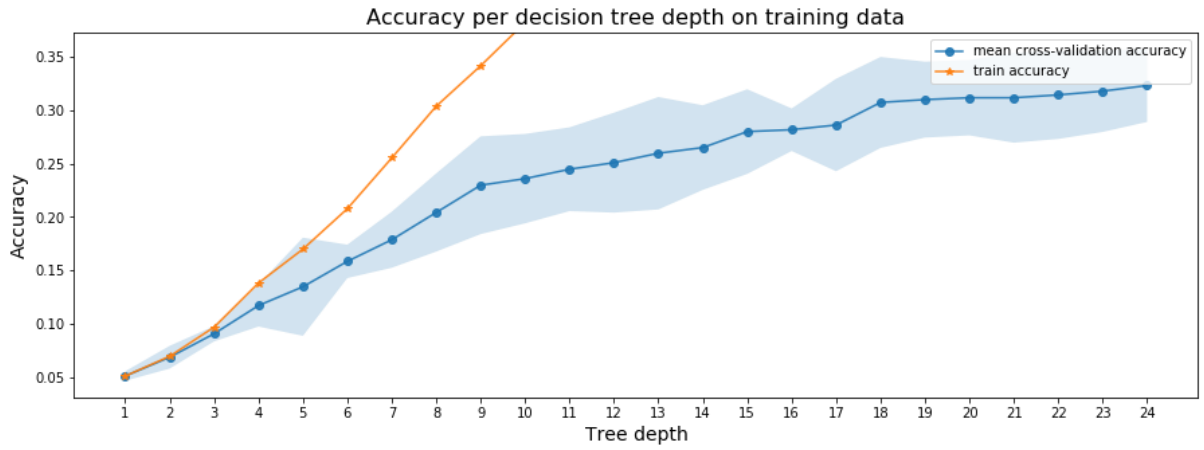


The same hill climbing technique was applied to the DT classifier with similar results. While bumping the accuracy of all data scalings a few percentage points higher they still only capped out at .30%.

Overall the GridSearch hypertuning was more effective in getting a more accurate classifier for both K Nearest Neighbour and Decision Tree.



Separately, the max depth of the decision tree depth was also tested via cross validation to see if it lined up with what GridSearch came up with and the results were similar. Seen below the most accurate the Decision Tree got was when it had the max depth set to 24 which was also what GridSearch got 3 out of the 5 different data inputs.

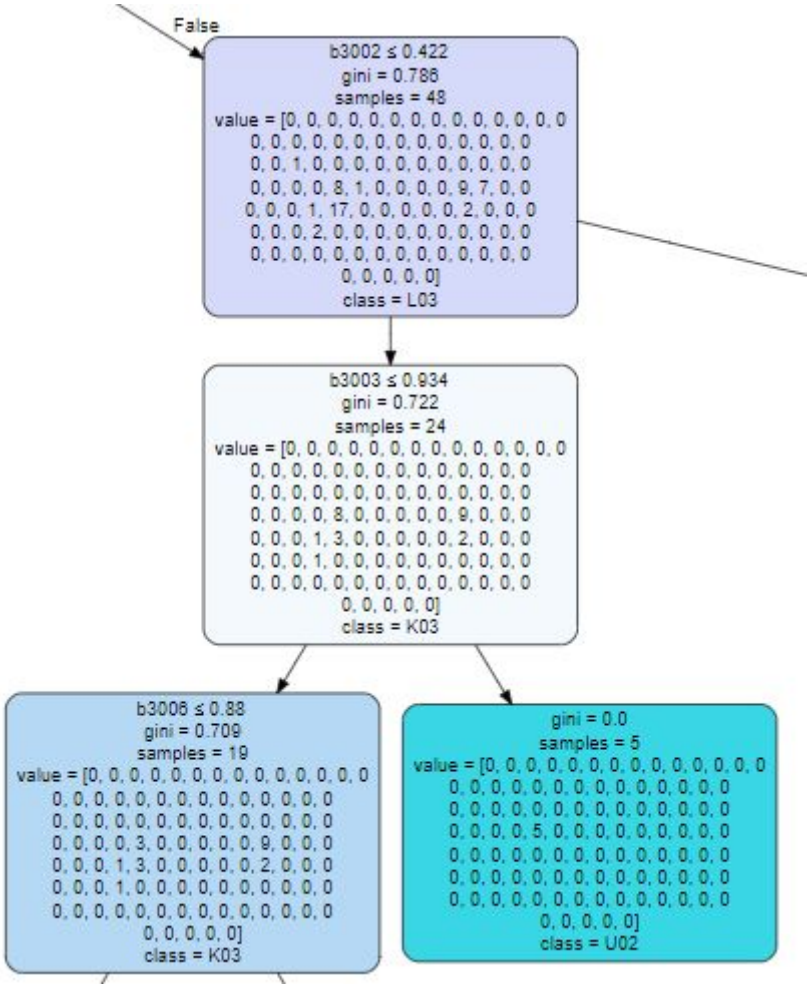


Hypertuning Results Table

Shown below are the accuracy scores for each combination of data and classifier that was experimented and their respective labels. Also included were the best resultant features that the hypertuning came up with for the respective calculations.

K = number of neighbours, p power level, MD = max_depth of tree, MF = max No. of feature

	Technique	KNN	DT
Base Score	Hill Climbing	0.30824372759856633 13 Features	0.3046594982078853 12 Features
	GridSearch	0.35214768339768343 K = 3, p = 1	0.3306654388212684 MD = 24, MF = 10
Standard Scaler Score	Hill Climbing	0.3118279569892473 12 Features	0.2974910394265233 12 Features
	GridSearch	0.3539253539253539 K = 5, p = 1	0.3288637091607944 MD = 26, MF = 10
Min_Max Scaler Score	Hill Climbing	0.3010752688172043 13 Features	0.2867383512544803 13 Features
	GridSearch	0.3512226512226512 K = 3, p = 1	0.32976057014734145 MD = 24, MF = 8
Normalized Standard	Hill Climbing	0.2939068100358423 13 Features	0.27956989247311825 12 Features
	GridSearch	0.3530405405405405 K = 3, p = 1	0.31541079436258807 MD = 14, MF = 8
Normalized MinMax	Hill Climbing	0.2939068100358423 12 Features	0.27598566308243727 12 Features
	GridSearch	0.3566361003861004 K = 3, p =1	0.31541479820627805 MD = 24, MF = 8



Decision Tree

Shown left is part of the Decision Tree generated by the highest accuracy GridSearch executed.
It demonstrates how the decision tree uses the ‘Gini index’ to know when to split

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

the data into branches and how to evaluate those branches next. Here it takes in the column data and target column ‘location’ and decides how to split the beacon data up into sections to be classified. Full tree was too large to include in this report.

Discussion

After reviewing the data and building various classifiers we can see that while some improvement was made to the accuracy of our classifier, there is much to be made in the way of progress before an app can be built upon this technology to help people find resources and their way around Waldo Library. The results found that there was a significant relationship between those beacons placed close together as well as an overall correlation between the location data and the RSSI data from the iPhone.
This report recommends using Normalized data which is then scaled with the Min Max algorithm and fed to a GridSearch function so that the best parameters and features are chosen for the K Nearest Neighbour classifier. As this is the best classifier that was found during this process it is recommended that you use K Nearest Neighbour for RSSI data.

Conclusion

Before this report we knew nothing of the way these beacons could relay location data and now we have uncovered that while our classifier is primitive there is room for improvement and there may yet be a way to build this navigation app, which if proved successful, could be explored further with not just libraries but other public or private buildings in the future. These beacons and technology could be used for location in all public settings guaranteed there be prevalent security installed with the beacons so that no user data was made public and there was no tampering of the beacons.

References

https://medium.com/datadriveninvestor/increase-10-accuracy-with-re-scaling-features-in-k-nearest-neighbors-python-code-677d28032a45

https://towardsdatascience.com/building-classification-models-with-sklearn-6a8fd107f0c1

https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a

https://towardsdatascience.com/how-to-find-decision-tree-depth-via-cross-validation-2bf143f0f3d6

https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029#:~:text=Normalization%20is%20a%20technique%20often,in%20the%20ranges%20of%20values.

https://www.datacamp.com/community/tutorials/machine-learning-in-r#five

https://en.wikipedia.org/wiki/Decision_tree_learning

https://scikit-learn.org/

https://matplotlib.org/

https://stackoverflow.com/

Practical Data Science with Python Lecture slides and Tutes from RMIT