Wifi Location Modelling – Capstone Project

IOT Analytics

Primary Objective:

- Evaluate the feasibility of using wifi signals to determine a customer's location in indoor spaces using the Jupyter Notebook environment.

Business Question:

- Can indoor location be predicted with enough accuracy with our models to warrant the development of a smartphone app to help customers.

Data Set

For this analysis and modeling we used the UJIIndoorLoc data set, which provides a training data set with 19300 observations and 529 variables. Of these 529 variables, 520 represent different wifi access points (WAPs) and the associated wifi signal strength detected. For these 520 WAPs, signal strength ranges from 0 (strongest) to -100 (weakest), with no-signal-detected values being recorded as 100.

The remaining variables either describe the location directly (building, etc) or are data capture variables (phone used, longitude/latitude, userID).

Exploratory Data Analysis

The data set contains the longitude and latitude coordinates associated with each reading. To validate the integrity and high-level accuracy of this data, the original building layout described within the documentation was compared to the longitude and latitude pairs of the entire data set. As can be seen in Figure 1, no data points appear to exist outside of the expected regions, validating that the readings were performed within the selected buildings as described.

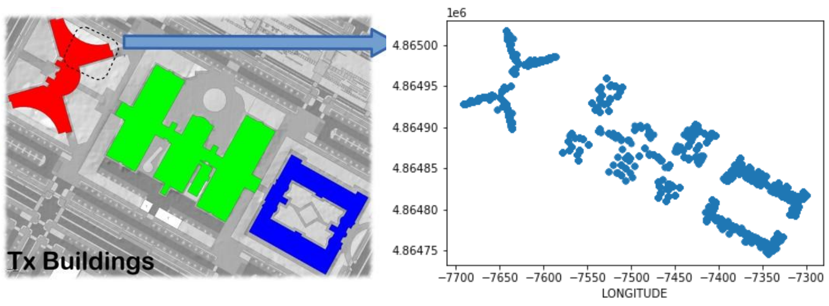


Figure 1: a) Building layout documented in the ULIIndoorLoc data set, b) Longitude/Latitude signal pairs in the data set

Signal strength, when detected, varied significantly across the data set. Through data stacking and removing the "no-signal" values, we are able to see the distribution of received signal intensities across all of the WAPs. These intensities were then separated by BUILDINGID and normalized to see the signal intensity profiles across each building. As shown in Figure 2, the majority of the reported signals

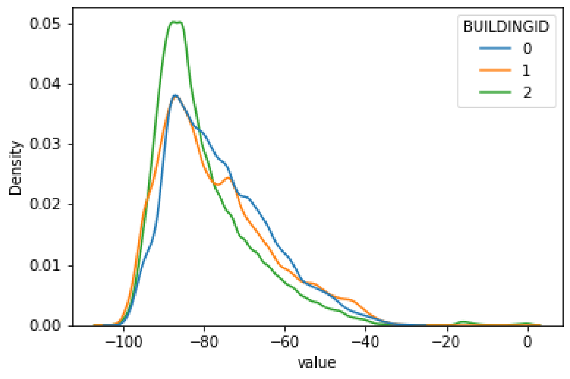


Figure 2: Distribution of WAP signal intensity for the three buildings

were in the weaker range (-80dB and below) for all three buildings. Moreover, observations in buildings 0 and 1 on average obtained marginally stronger WAP signals.

Data Preparation

After initial cleaning, the 4 variables used to directly describe the location (BUILDINGID, FLOOR, SPACEID, RELATIVEID) combined into a single variable, with letters segregating for easier human reading (ex: "B0 F1 S106 R2" = Building 0, Floor 1, Space 106, Relative 2). This combined variable, named "Location" is forms our dependent variable. To help build classification models with scikit-learn, we use the LabelEncoder on our dependent variable (DV) prior to modelling to transform our string labels to numeric values between 0 and (classes-1). While LabelEncoder can result in some models interpreting the values as ordinal, it is necessary for maintaining a singular dependent variable.

While information such as longitude and latitude could be helpful in predictive modelling, in real-world indoor environments it is unlikely that GPS signals would be available. Also, while PhoneID could be gathered from a potential app, the large number of phone models available to the general public far exceeds the handful of models represented in this data set. As such, PhoneID was removed as a variable for the training data sets. The TIMESTAMP variable was also removed – while it could help predictions in this data set, as it was deemed irrelevant for predicting in a live use case when a user queries for location in real-time. Finally, USERID was removed, as the model would be expected for perform well for new users entering the building space, and no prior data would exist for them.

The base training set therefore only consists of 520 WAP signals, as well as the categorized location dependent variable. In this way, location predictions are made solely on WAP signal input.

Training Sets

"dfS"

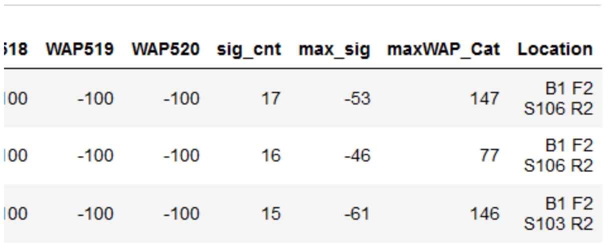
The first data set is a 25% sample of the data set created from Data Preparation. This set contains 4825 observations and 521 variables.

"dfB0"

The data set was filtered to only include locations from BUILDINGID = 0. The concept explored here is that if each building to has a separate tuned model built for it, an overall increase in accuracy could be observed. No further sampling was necessary, and the resulting data set contains 5246 observations and 521 variables.

"dfFES"

For the feature engineered data set, several variables were added:



18	WAP519	WAP520	sig_cnt	max_sig	maxWAP_Cat	Location
00	-100	-100	17	-53	147	B1 F2 S106 R2
00	-100	-100	16	-46	77	B1 F2 S106 R2
00	-100	-100	15	-61	146	B1 F2 S103 R2

Figure 3: Last 6 columns of the dfFES data set, showing the 3 new variables. Snapshot is prior to DV LabelEncoding

- **sig_cnt**: A count of the total number of detected WAPs for each observation. The motivation is that the sheer number of nearby WAPs could indicate whether a location is more centralized in a building, or in one of the building's corners. Similarly, if WAP density varies by building, this could also be insightful for identification.
- **max_sig**: The max WAP signal intensity received for each observation.

- **maxWAP**: The ID number of the WAP providing the strongest signal intensity for each observation (ex: WAP393). This string value was also encoded using LabelEncoder into a numerical value to replace the variable with **maxWAP_Cat**. While one-hot encoding would prevent some models from interpreting these values ordinally (as can happen with LabelEncoder), the high dimensionality (520 classes) would significantly slow learning algorithms. Binary encoding was considered, but it would still result in 6 new columns. Instead, given the large number of WAPs represented, and the unknown importance of this variable, LabelEncoder was used.

The feature-engineering data set was then sampled at 25% to produce a set with 4825 observations and 524 variables. A snapshot of this data set is shown in Figure 3.

Modelling

Cross-Validation

For this classification task, three models were chosen (Decision Tree, kNN, and SVM) and applied to each of these three data sets through a 3-fold cross-validation. Decision Tree and kNN were selected because they are low bias/high variance algorithms that do well with large data sets. SVM was included as it generally fares well on data sets with large numbers of features and high dimensionality. The results are shown in Figure 4 for the min, average, and maximum accuracy.

Data Set	Model	Accuracy	Max	Min
dfS	DT	0.33	0.34	0.33
dfB0	DT	0.52	0.54	0.49
dfFES	DT	0.40	0.41	0.38
dfS	KNN	0.33	0.33	0.33
dfB0	KNN	0.50	0.51	0.48
dfFES	KNN	0.40	0.41	0.39
dfS	SVM	0.49	0.51	0.48
dfB0	SVM	0.58	0.59	0.57
dfFES	SVM	0.60	0.60	0.60

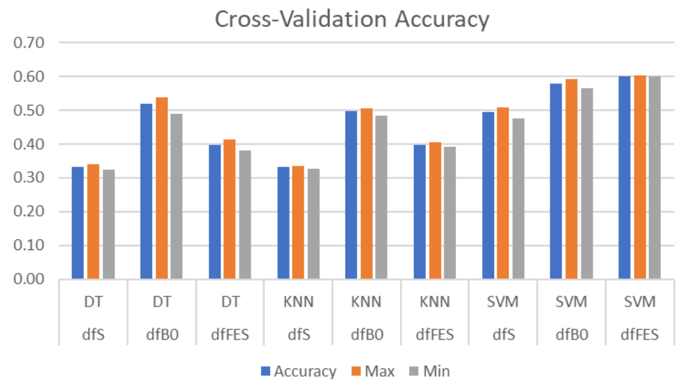


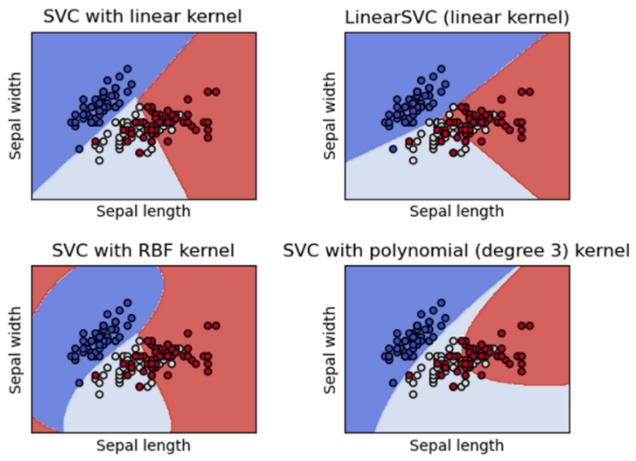
Figure 4: Cross-validation results of 3 data sets and 3 models

This cross-validation shows that our single-building data set modelling (dfB0) performs consistently better than the out-of-the-box sampled dfS data set across all three models. With accuracy improvements ranging between 9% and 19%, building separate models per building could be beneficial in commercial use. However, to implement, the app would need a way to confirm the BUILDINGID, so the correct model could be used. This could be achieved either through GPS (before the user enters the building), or through a separate model layer which is only used to predict BUILDINGID based on WAPs, which likely could be done at very high accuracy.

The feature-engineered data set (dfFES) performed consistently better than the dfS data set across all models, without any restriction on BUILDINGID. Most notably, the greatest performance benefit versus dfS occurred on the overall highest performing model (SVM). In SVM, this feature-engineered data set even out-performed the single-building data set, which is impressive.

Model Optimization

Optimization of the SVM model with the feature-engineering data set was performed to attempt to further improve performance. In this optimization different kernel functions were explored to generate different multi-dimensional classification spaces. Linear, poly, and rbf kernels were investigated, with the cross-validation results shown in Figure 5. The linear kernel (as performed in the original cross-validation) shows performance effectively matched to the poly kernel, and generally should have faster model generation times. As such, the linear SVM model was applied to the feature-engineered training and test sets.



Data Set	Model	Accuracy	Max	Min
dfFES	SVC(linear)	0.602	0.604	0.600
dfFES	SVC(poly)	0.602	0.605	0.600
dfFES	SVC(rbf)	0.115	0.124	0.109

Figure 5: SVM kernel optimization. Kernel images courtesy of scikit-learn.org

With this same sampled feature-engineering set, an accuracy was achieved on the test set of **0.63**. This is encouraging, as the lack of a performance shift means overfitting was likely minimal. Given the large number of classes in the data set for location, in this feature-engineering sampled training test set of 3618 observations many classes only had 1 or two instances on which to train. To increase the number of class instances, this model was applied to the full un-sampled feature-engineered data set (training = 14475 observations). This model demonstrated a dramatic improvement with an accuracy score of **0.81** while maintaining a reasonable training time (~4min).

Recommendations

This trial has shown that modelling indoor location using only wifi signal strength can successfully provide highly accurate predictions for location. This methodology can be used for future deployments.

Machine learning classification tasks such as these can be quite challenging primarily due to the large number of classes within a location dependent variable. This problem scales directly with the specificity that the algorithm tries to give the user in terms of precise location. As a side effect of having a high number of classes, it can be challenging for data sampling to provide enough instances of each class to generate a model. However, with high-efficiency models such as SVM large data sets can be trained in reasonable time frames, so sampling should be limited when creating future customer-specific location models to maintain a high model accuracy.

For further improving accuracy, time-stamping would be an attractive addition to the data set. In real-time-location-systems (RTLS), wifi tags can send a message that is received by multiple WAPs. As this message has a specific time stamp associated with it, and due to the fact that all WAPs are time synchronized, the transmit delays can be used to triangulate the wifi tag in real-time. Specifically, if at least 3 WAPs receive the message, the tag can be accurately triangulated in 3-D space, which is much more useful in indoor deployments than the 2-D longitude/latitude provided by GPS. However, it will require further investigation on how to institute this technology through a smartphone locationing app instead of a dedicated hardware wifi tag.