Indoor Localization based on WiFi Fingerprinting

Ryan Clark, Matt Hong, So Sasaki, Sophia Huang
Group 27
Department of Electrical and Computer Engineering
University of California, San Diego
rmclark@eng.ucsd.edu, mattjoo.hong@gmail.com, sosasaki@ucsd.edu, jih201@ucsd.edu
Github: https://github.com/ryanmclark/Localization_via_WiFi_Fingerprinting

Abstract—**Many applications depend on being able to reliably localize a target in order to provide their respective services. This is especially true in the field of robotics where localization is the pinnacle of path planning. That being said, automatic localization has been a major focus in robotics for quite some time. GPS sensors have been able to reliably localize various devices in many cases, however, this is not true when a GPS signal cannot be established due to environmental conditions such as being indoors. WiFi fingerprinting has been a very popular approach to solve this problem by utilizing the Received Signal Strength Indicator values across multiple Wireless Access Points (WAPs) to localize a target with respect to the locations of the WAPs. In this paper, we investigate multiple supervised learning techniques on the UjiIndoorLoc dataset in order to localize targets based on a dataset of WAP intensities against their respective localization labels. K-Nearest Neighbors (KNN) had the highest accuracy with a 99.79% building accuracy, 99.76% floor accuracy, and average error of 1.78m.**

Index Terms—**Supervised Learning, WiFi Fingerprinting, Localization, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machine, Principal Component Analysis**

I. Introduction

WiFi fingerprinting uses the Received Signal Strength Indicator (RSSI) values given by multiple Wireless Access Points (WAP) in order to localize a target with respect to the WAP locations. WiFi Fingerprinting has been a popular approach used to study human behavior [3], localization for robotic applications, and many other studies [1]. WiFi Fingerprinting is a popular approach due to the pre-existing WLAN infrastructure that comes in many larger buildings [1]. The additional equipment needed is a reliable receiver. As attractive as the WiFi Fingerprinting approach may appear, it can be rather difficult due to the noisy/unreliable nature of WAP signals as well as inconsistent indoor environmental conditions; e.g. walls, furniture, floors, and so on. In this paper, we approach the WiFi Fingerprinting problem by utilizing supervised learning on a publicly available dataset, UJIIndoorLoc, that contains samples of the RSSI values from many WAPs with their respective localization labels. The localization labels were defined by quantitative values of latitude and longitude, and categorical labels of floor and Building ID.

The supervised learning techniques explored in this paper were K-Nearest Neighbors (KNN), Decision Trees (DT), Random Forests (RF), and Support Vector Machines (SVM). Principle Component Analysis (PCA) and Variance Thresholding were also done for dimensionality reduction. The supervised learning was accomplished by inputting WAP RSSI values to predict the location of the device. A separate regression and classification method was implemented for each machine learning technique predicting the quantitative latitude and longitude, and categorical Building ID and Floor ID, respectively.

II. Related Work

A number of studies have been conducted on WiFi fingerprint localization, many of which use the WAP RSSI values in addition to a variety assisting sensors such as magnetic field-based methods, inertia-based methods, etc [8]. A paper by Hong et al. demonstrates how WiFi fingerprints can be effective source of localization in a temporal based particle filter approach [4]. Out of all of the approaches explored, the particle filter approach produced the best results. Unfortunately, the UJIIndoorLoc does not provide the data needed in order to conduct a reliable temporal based solution to this problem.

Many researchers from both academia and industry have worked on UJIIndoorLoc especially for EvAAL 2015. [6] EvAAL 2015 had a competition track called ”Wi-Fi fingerprinting in large environments,” where only UJIIndoorLoc is available and another track called Foot-mounted pedestrian dead reckoning positioning, where UJIIndoorLoc and MEMS sensors (inertial, compass and pressure sensors) are available. In the former track, Maximum Likelihood Estimation and k-Nearest Neighbor algorithm [7] won the first place with 11.6-meter mean error on an off-site evaluation. In the latter track, Advanced Heuristic Drift Elimination [8] which can remove azimuth drift error in indoor environments won the first place with 1.9-meter mean error on an on-site evaluation.

Other studies have been also conducted with UJIIndoorLoc datasets. Bozkurt et al. [9] and Aydin et al. [10] employ statistical machine learning methods while Nowicki et al. [11] and Nowicki et al. [12] adopt deep neural network algorithms. In particular, Bozkurt et al. compare a wide range of machine learning algorithms: k-Nearest Neighbor, Decision tree, Adaboost,Bagging, Nave Bayes, Bayesian Network, and Sequential minimal optimization. Bozkurt et al. conclude that with the 99.7% building classification accuracy and 98.5% floor classification accuracy, k-Nearest Neighbor algorithm is the best algorithm for indoor positioning. However, their comparative study is limited only for the classification problem. Our research compares and investigates machine learning algorithms in both classification and regression problem for indoor localization

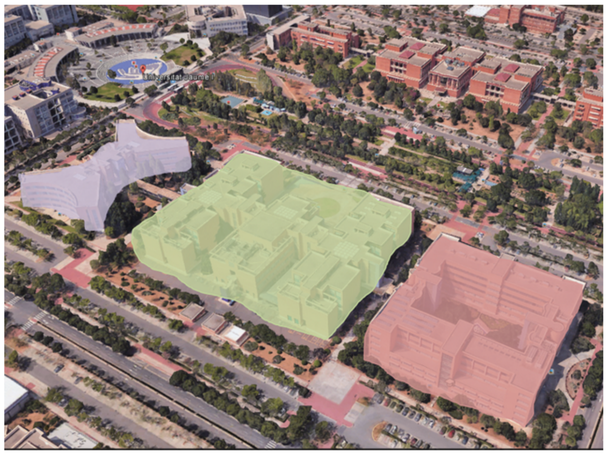

Fig. 1. Universitat Jaume I dataset buildings - Source: http://www.analyzingdata.org/portfolio/003-Indoor_localization_Wifi_Footprint/

III. Dataset and Features

This dataset covers a 108,703m^2 area that includes 3 buildings with 4 or 5 floors depending on the building [1]. It consists of 21,049 samples that have been partitioned into two pre-built datasets. There is a training dataset that contains 19,938 samples, and a testing dataset that contains 1,111 samples [1]. Each sample consists of 529 attributes, and an example is shown in Table 1. The first 520 attributes consist of different WAPs with RSSI values that range from -104dB to 0dB and denoted null value of 100 for WAPs that are not detected for each sample. Longitude and Latitude values for each sample are given in meters. A building ID corresponding to one of the three buildings is given as either 0, 1, or 2. A floor number corresponding to one of the four floors is given as either 0, 1, 2, 3, 4. The rest of the attributes were discarded in our supervised learning approach but included a space ID corresponding to the type of location that the measurement was captured at (e.g. office, classroom, etc.), a relative position with respect to the space ID (front of door, outside of door, etc.), a user ID, a phone ID, and a timestamp. The space ID and relative position attributes were of little interest to begin with, however, the user ID, phone ID, and timestamp were considered in a potential temporal approach to the problem but proved to be of minimal use due to the lack of causality in the timestamps given gaps between measurements that could range from a few dozen seconds to a few minutes. We will note that the phone ID and timestamps were used for quite a bit of analysis that assisted in preprocessing decisions but was ignored during classification and regression.

The UjiIndoorLoc dataset comes in two sub-datasets - training and testing. However, some initial experiments led us to believe that the two datasets were not drawn randomly from an initially combined dataset, but rather created separately with the other dataset in mind. In fact, the authors of this dataset state that "Dataset independence has been assured by taking Validation (or testing) samples 4 months after Training ones" [1]. This would not be an issue if the two datasets were created in a similar manner, however, figure 2 reveals that this is not the case. In figure 2, we can see one angle of the training and testing datasets' latitude, longitude, and floor number plotted onto a 3D scatter plot. What this reveals is that a significant portion of the testing dataset does not align with the training dataset, and it appears that the testing dataset was created purposefully to be different from the training set. The testing set explored many portions of the buildings (entire corridors, rooms, hallways) that were not even closely passed through in the training set. This directly goes against the statistical learning assumption that all data points are realizations of independent and identically distributed random variables when temporal or spatial correlations are not a factor. As stated before, there is not any reliable temporal relation in this dataset, and the authors of the dataset discuss how they draw the datasets independently which removes all possibilities of spatial correlations [1]. This was not discovered until we made a subset of the original training set to be a validation set for parameters and discovered that the validation errors were drastically lower than the testing errors. Following this, we decided that it was best to drop the provided testing set entirely and then draw 80/20% splits for training and testing datasets or 60/20/20% splits for training, validation, and testing datasets where appropriate. These sub-datasets were drawn randomly and all prepossessing and analysis was done using the training and validation set. The test set was not interacted with until it was time to produce final results. This was enforced by setting a random seed when the initial data was separated and then drawing the validation set from the already separated training set when needed. Unfortunately this adjustment compromises the ability to compare our results with the results generated by other papers. We believe that the statistical learning assumption violation is a much more severe thing to address over producing comparable results against a flawed dataset. In a similar manner to the provided testing dataset, all of the data associated to phone ID 17 in both datasets were removed due to having dozens of duplicate data points with many different labels. The data from this phone ID was clearly corrupted.

Following this correction, some preprocessing steps were delicately used on the dataset. The initial prepocessing step that was applied was to linearly normalize the dataset between 0 and 1 with 0 being null and/or weak RSSI intensities and 1 being the strongest RSSI intensities. To do this, we would reassign the null value to be one below the minimum RSSI intensity of -104dB and then we would simply divide the dataset by that minimum value since the data was negative. Afterwards, we subtracted the data from 1 to flip the signal strength. From this point forward, we will assume that the data is normalized in this range, however we will refer to the original decibel scale for readability. The next step was to find out what range of RSSI intensity values were reliable. To do this, we used the K-Nearest Neighbors Regressor from the python package sklearn, and compared the performance as we swept the null value from -110dB to -80dB. Since

WAP_{001}	...	WAP_{520}	Longitude	Latitude	Floor	BuildingID	SpaceID	RelativePosition	UserID	PhoneID	TimeStamp
100	...	100	-7515.916799	4864890	1	1	0	0	0	0	1380872703

TABLE I
EXAMPLE OF SINGLE DATASET SAMPLE

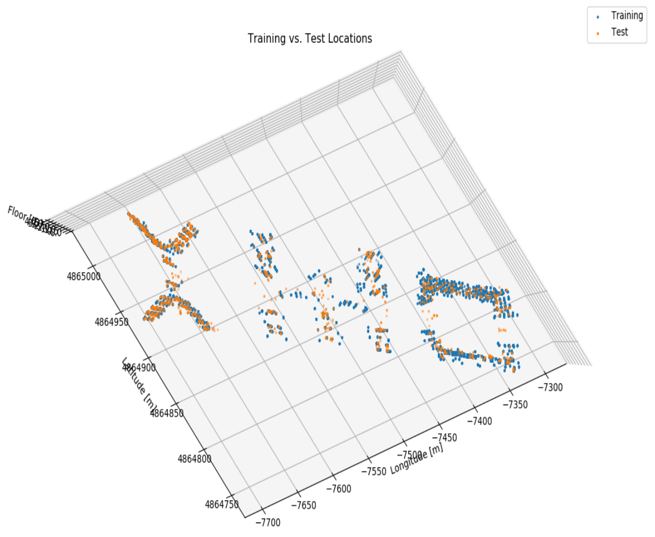

Fig. 2. Difference in Training and Validation Sample Locations

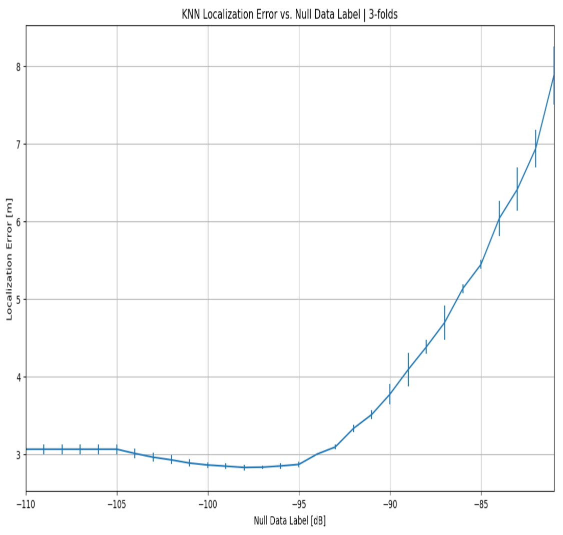

Fig. 3. Localization Error vs. Null Intensity Label

leaving the null value at 100 created a high penalty for some classifiers and regressors, we thought it would be a beneficial step to query the best null value. The null value would be set to one of the numbers in the previously mentioned range, and all values below it would be changed to that value thus determining which WAP RSSI intensity values proved to be reliable. Figure 3 shows us that the null value that produces the least localization error (euclidean distance from true location) on the validation set was -98dB. From this point forward, any RSSI intensity value that was either null or below -98dB was set to be -98dB.

Separately from the previous preprocessing, we decided to analyze how many WAPs were active (giving non-null measurements) for each given sample. Figure 4 shows a histogram of the number of samples having a certain number of active WAPs. From this we can draw the conclusion that the majority of samples had many active WAPS and that a potentially high source of error was coming from samples without enough active WAPs. To prove this, we compared the error from the same K-Nearest Nearest Neighbors Regressor against a query that would eliminate samples from the dataset that did not have at least n active WAPs on it, where $n \in [0, 20]$. Figure 5 shows a plot of the error versus the minimum required active WAPs per sample as well as the amount of data that would be lost from choosing to do so. It is important to note that this preprocessing step was done in addition to and following the previous preprocessing step following the analysis. The results show that even requiring one active WAP showed drastic improvement (Note that some samples had zero active WAPs in the unaltered dataset). Since there is a reduced error versus data lost trade off, we decided that requiring at least 9 active WAPs per sample at a cost of 8% of the data was a worthwhile trade. To back up this case in point, it is physically impossible to localize an object in 3D space without having at least four isotropically generated RSSI intensity values active on it when restricted to a single moment of time.

The last form of preprocessing that was done on this dataset was feature reduction. We decided that our dataset needed some form of feature reduction given Figure 6, which shows the number of active samples each WAP has. It is clear that some WAPs hardly appear in the dataset and thus have a very limited variance. Two separate approaches were used here. One was to query the results while using the sklearn variance threshold package and the other was to use Principle Component Analysis (PCA). The variance threshold was set to remove any feature that had zero variance following all of the previous preprocessing, and the PCA was used to keep the orthogonal components that explained 95% of the variance. Figure 7 shows the variance explained versus the component in descending order for the dataset. this plot reveals that the majority of the variance in the dataset is in the first 100 components. In fact, 95% of the explained variance was in the first 96 components.

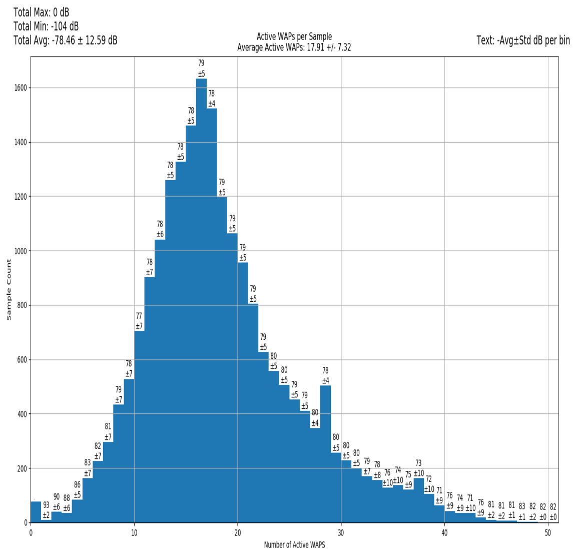

Fig. 4. Histogram of Sample Count vs. Number of Active WAPS

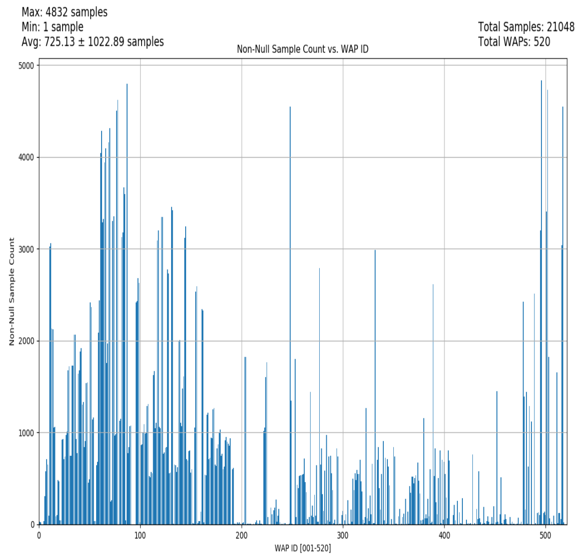

Fig. 6. Histogram of Number of Active Samples vs. WAP ID

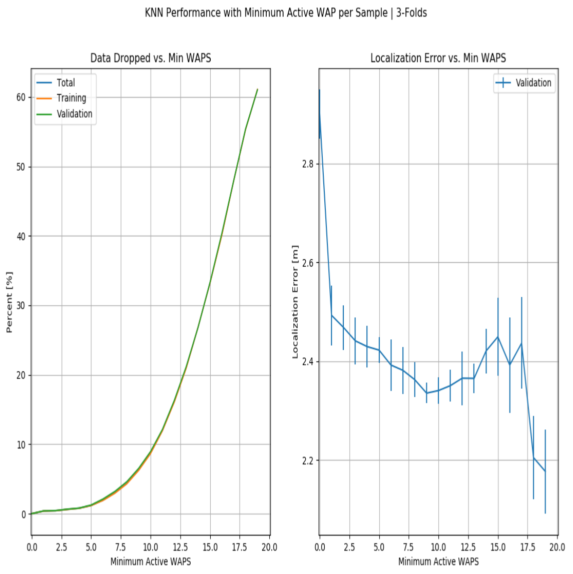

Fig. 5. Localization Error vs. Minimum Required Active WAPs

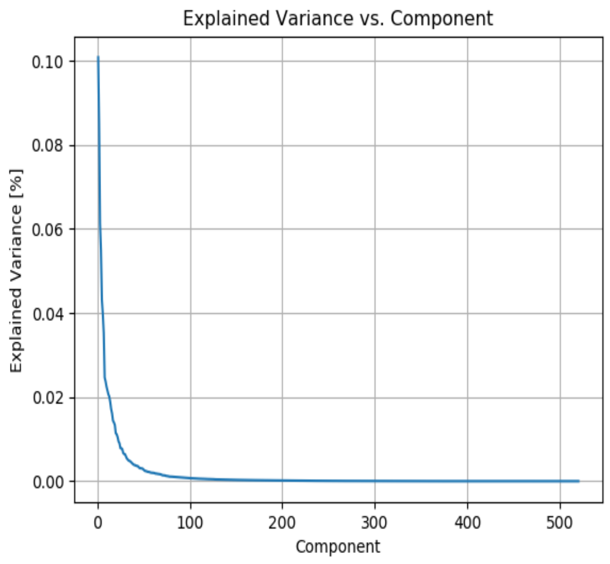

Fig. 7. PCA: Variance Explained vs. Components

IV. Methods

A. Decision Tree

A Decision tree is a tree whose internal nodes can be taken as tests (on input data patterns) and whose leaf nodes can be taken as categories (of these patterns). More general, it is a form of template matching which compare the input x to the stored prototypes μ_k.

$$\phi(x) = [\kappa(x, \mu_1), ..., \kappa(x, \mu_N)]$$

Instead of using a pre-defined kernel function κ, the basis function ϕ is learned from the input data by selecting the useful features (formally, this method belongs to the adaptive basis function model of the form $f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x)$). The basis function define the regions (part of the input feature space partitioned by the node), and the wights specify the response value in each region.

B. Regression Tree

Each node partition the input based on a feature. For example, the first node compare the feature x_1 of every input against the threshold t_1. If yes, then the sub-node compare the feature x_2 of every input against the threshold t_2. If no, then the sibling-node would compare the feature x_1 of every input

against the threshold t_3. And so on. The result of these axis parallel splits the feature space into multiple regions.
The model can be formulated in the following form

$$f(x) = \mathbb{E}[y|x] = \sum_{m=1}^{M} w_m \mathbb{I}(x \in R_m) = \sum_{m=1}^{M} w_m \phi(x; v_m)$$

where R_m is the m'th region (m'th leaf), w_m is the mean response in this region, and v_m encodes the choice of variable to split on, and the threshold value, on the path from the root to the m'th leaf.

Regression Cost

$$cost(D) = \sum_{i \in D} (y_i - \bar{y})^2$$

where

$$\bar{y} = \frac{1}{|D|} \sum_{i \in D} y_i$$

is the mean of the response variable in the set of data.

C. Classification Tree

Similar to the Regression Tree, Each node partition the input based on a feature. For example, the first node compare the feature x_1 of every input against the threshold t_1. If yes, then the sub-node compare the feature x_2 of every input against the threshold t_2. If no, then the sibling-node would compare the feature x_1 of every input against the threshold t_3. And so on. The result of these axis parallel splits the feature space into multiple regions.
The model can be formulated in the following form

$$f(x) = \mathbb{E}[y|x] = \sum_{m=1}^{M} w_m \mathbb{I}(x \in R_m) = \sum_{m=1}^{M} w_m \phi(x; v_m)$$

where R_m is the m'th region (m'th leaf), w_m is the distribution over class labels in this region, and v_m encodes the choice of variable to split on, and the threshold value, on the path from the root to the m'th leaf. The distribution over class labels stored at each leaf node is the empirical fraction of positive examples that satisfy each conjunction of feature values, which defines a path from the root to a leaf.

Classification Cost Entropy. Specifically, minimizing the entropy is equivalent to maximizing the information gain between test $X_j < t$ and the class label Y

$$infoGain(X_j < t, Y) \equiv \mathbb{H}(Y) - \mathbb{H}(Y|X_j < t)$$

D. (Greedy) Growing a tree

The greedy procedure is implemented to compute the locally optimal MLE partitioning of the data.
The split function chooses the best feature, and the best value for that feature

$$(j^*, t^*) = \underset{j \in \{1,2,\ldots,D\}}{\text{argmin}} \min_{t \in \tau_j} Y$$

where

$$Y = cost(\{x_i, y_i : x_{ij} \leq t\}) + cost(\{x_i, y_i : x_{ij} > t\})$$

and τ_j defines the possible threshold for feature j.

E. Random Forest (Decision Tree Forests)

Several varied decision trees are trained independently and the response is then averaged across those trees. The variance of an estimate is reduced when average many estimates. More specifically the ensemble is computed as below

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x)$$

where f_m is the m'th tree. Since simply re0running the same learning algorithm on different subsets of the data can result in highly correlated predictors, which limits the amount of variance. The random forests de-correlate the base learners by learning trees based on a randomly chosen subset of input variables.

F. K-Nearest Neighbors

The K-Nearest Neighbor algorithm is one of the simplest yet powerful algorithms in supervised machine learning. In classification, an object , in this case the fingerprint, is classified to the class most common among its k nearest neighbors. In regression, an object, has the average values of its k nearest neighbors. In this project, we used $K = 1$ and a 1-norm distance metric, thus the distance between the object and its neighbors is determined by the Manhattan distance, which is computed as below

$$D = \sqrt{\sum_{i=1}^{d} |x_i - y_i|}$$

where x_i is the sample of interest and y_i is the neighbor over all d features.

G. Support Vector Machine

Support Vector Machine is a machine learning algorithm which uses a separating hyperplane to classify data. In a linear kernel, the hyperplane can be defined as

$$y = w^T x + b$$

and the margin is defined as

$$r = \frac{w^T x + b}{||w||}$$

The maximum margin can be represented as

$$\underset{w}{\text{argmax}} \frac{r}{2}$$

This can be normalized and transformed into an optimization problem as

$$\underset{w,b}{\text{argmin}} \frac{||w||}{2}$$

A slack variable ξ is introduced to allow for misclassification. The final optimization problem becomes

$$C \sum_{i=1}^{N} \xi_n + \underset{w,b}{\text{argmin}} \frac{||w||}{2}$$

H. principal components analysis (PCA)

The principal component method allows dimension reduction by selecting an orthogonal linear basis vector $w_j \in \mathbb{R}^D$ and corresponding scores $z_i \in \mathbb{R}^L$ such that minimize the average reconstruction error

$$\frac{1}{N}\sum_{i=1}^{N} ||x_j - \hat{x_i}||^2$$

The optimal solution is obtained by setting the $\hat{W}$ in $\hat{x_i} = WZ^T$ by the eigenvectors with largest eigenvalues of the covariance matrix $\frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$

V. Experiments/Results/Discussion

Most of the model's hyper-parameters were tuned with 3 or 5 fold cross validation depending on the model runtime. For K-Nearest Neighbors (KNN), we choose to implement 1 neighbor, the kd-tree algorithm with 50 samples per leaf for efficient pruning, and most importantly the L1 norm distance metric for less sensitivity to outliers. The KNN produced the best results with 1 neighbor simply because this dataset consisted of many duplicate datapoints with the same label. When the results of the KNN are explored in detail, one will see that it either produces 0 error or very large error in the regression case. The Random Forest Model boosted the amount of trees to 100 as opposed to the default 10 trees. The Support Vector Machine used a linear kernel and the slack penalty term was tuned with cross validation to arrive at a value of 100 which corresponds to a relatively hard margin. The linear kernel preformed the best for the same reason the KNN preformed the best with a single neighbor. Lastly, the Decision Tree model was left unaltered.

The metrics used to compute the regression error are the mean coordinate error which is the mean L2 norm between the predicted latitude and longitude and the actual latitude and longitude. The original authors of the dataset created standardized way to calculate the error which we will call the standard error [2]. We will provide the formula for this metric, however, we chose not to implement it due to the fact that we have dropped their provided test set and therefore forfeited all means of comparing results. This standard error penalized building and floor missclassifcations with respective weights and then summed them to the coordinate errors. We also used the R2 Score for regression and accuracy for the classification. Formulas for the metrics are provided below.

Mean Coordinate Error:

$$e_m = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{x}_{0,i}||_2$$

where $\mathbf{x}$ is the predicted latitude and longitude and $\mathbf{x_0}$ is the true latitude and longitude.

R2 Score:

$$r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

where y_i is the data point, $\hat{y}_i$ is the estimated regression line, and $\bar{y}_i$ is the mean.

Regression Results				
	Mean Coord Error		R2 Score	
Model	Training	Testing	Training	Testing
KNN	0.44	1.78	0.992	0.995
RF	0.41	4.56	0.997	0.994
DT	1.75	4.09	0.996	0.996
SVM	45.56	57.19	0.799	0.806

TABLE II

Classification Results				
	Accuracy Building		Accuracy Floor	
Model	Training	Testing	Training	Testing
KNN	0.992	0.998	0.992	0.998
RF	0.993	0.997	0.993	0.995
DT	0.993	0.996	0.993	0.962
SVM	0.993	0.997	0.983	0.992

TABLE III

Accuracy:

$$Accuracy = \frac{True\ Positive\ +\ True\ Negative}{Total}$$

Precision and recall are computed based on the confusion matrix which provides the count for True Positive, False Positive, False Negative, True Negative. For this multilabel problem, the precision and recall are computed for each label (0, 1, 2), which is equivalent to precision and recall for predicting label 0 (or 1, 2) or non-0 (or non-1, non-2) in the binary classification problems.

$$Precision = \frac{True\ Positive}{Actual\ Results}$$

Precision measures the fraction of predicted class label that are relevant/correct.

$$Recall = \frac{True\ Positive}{Predicted\ Results}$$

Recall measured the percentage of relevant results classified by the model.

Beginning with the regression results, we can see that the KNN model significantly outperformed all of the other models while the SVM preformed terribly. Both of these results can be explained due to the fact that much of the data had nearly identical features but vastly different latitude and longitude labels in conjunction with the fact that there were also many identical duplicates in the dataset. SVM was unable to separate this issue under many different kernels, which caused the regression error to be high. KNN however, was able to find the matching sample which caused it to have the lowest error. Scrutinizing the error more closely reveals that much of the KNN errors were either zero or very high (+10 meters) due to the large amount of duplicates in the dataset. Even though KNN preformed the best, we believe that a similar dataset that was better put together would reveal that the Random Forest would preform the best for this test.

The classification results reveal that all of the models preformed relatively well, even the Support Vector Machine which produced terrible regression results. This can be attributed to the duplicate data sharing a similar label as opposed

K-Nearest Neighbors Confusion Table - Buildings				
		Predicted Results		
		0	1	2
Actual	0	1088		
	1		939	8
	2			1785

TABLE IV

K-Nearest Neighbors Confusion Table - Floors						
		Predicted Results				
		0	1	2	3	4
Actual	0	877				
	1	1	937			
	2			923		
	3	8			945	
	4					129

TABLE V

to the regression case. There were 2 trends we were able to detect from the confusion matrices, which can be seen in Tables 4-11. The first is that the DT, RF, and SVM models all classified 11 floors as floor 3 when it was actually floor 0. The cause behind that is of interest, however, that may be something to look into for future work. The second is that a misclassification is more likely to occur by misclassifying the predicted floor as a lower floor than the truth. The precision and recall matrices can be seen in Tables 12 and 13. However, because the accuracy was very high, no new information was gained. Overall, KNN out preformed all of the other models for the same reasoning as the regression case.

Figures 8 through 11 show how each algorithm preformed with localizing targets for phone ID 20. These results hold similarly for other phone IDs and phone ID 20 was chosen due to is visibility from having less data points. It is difficult to tell which model preformed the best by looking at these plots - SVM excluded. However, if we show an example of a phone ID with many duplicate points, we will see how KNN significantly outperforms everything else. Let's compare phone id 19 between the KNN and RF regressors in figures 12 and 13. Clearly KNN outpreforms its biggest competitor, RF, here due to the high number of duplicate data points.

Random Forest Confusion Table - Buildings				
		Predicted Results		
		0	1	2
Actual	0	1088		
	1		947	
	2		11	1774

TABLE VI

Random Forest Confusion Table - Floors						
		Predicted Results				
		0	1	2	3	4
Actual	0	862	4		11	
	1		936	2		
	2		2	919	2	
	3				953	
	4					129

TABLE VII

Decision Tree Confusion Table - Buildings				
		Predicted Results		
		0	1	2
Actual	0	1088		
	1	1	944	2
	2		13	1772

TABLE VIII

Decision Tree Confusion Table - Floors						
		Predicted Results				
		0	1	2	3	4
Actual	0	841	22	3	11	
	1	16	892	22	6	2
	2	2	16	883	17	5
	3	2	4	10	931	6
	4			2	2	125

TABLE IX

Support Vector Machine Confusion Table - Buildings				
		Predicted Results		
		0	1	2
Actual	0	1088		
	1		947	
	2		11	1774

TABLE X

Support Vector Machine Confusion Table - Floors						
		Predicted Results				
		0	1	2	3	4
Actual	0	862	4		11	
	1	1	934	3		
	2		4	914	5	
	3			2	950	1
	4					129

TABLE XI

Floor Accuracy			
Method	Label	Precision	Recall
KNN	0	1.0000	0.9898
	1	0.9989	1.0000
	2	1.0000	1.0000
	3	0.9916	1.0000
	4	1.0000	1.0000
	avg	0.9981	0.9980
RF	0	0.9840	1.0000
	1	0.9979	0.9947
	2	0.9957	0.9967
	3	0.9990	0.9845
	4	0.9845	1.0000
	avg	0.9922	0.9922
DT	0	0.9624	0.9803
	1	0.9510	0.9622
	2	0.9567	0.9619
	3	0.9811	0.9560
	4	0.9612	0.9118
	avg	0.9625	0.9544
SVM	0	0.9829	0.9988
	1	0.9957	0.9915
	2	0.9902	0.9946
	3	0.9969	0.9834
	4	1.0000	0.9923
	avg	0.9931	0.9931

TABLE XII

Building Accuracy			
Method	Label	Precision	Recall
KNN	0	1.0000	1.0000
	1	1.0000	1.0000
	2	0.9916	1.0000
	avg	1.0000	0.9955
RF	0	1.0000	1.0000
	1	1.0000	0.9885
	2	0.9938	1.0000
	avg	0.9979	0.9962
DT	0	1.0000	1.0000
	1	0.9989	0.9874
	2	0.9933	0.9994
	avg	0.9974	0.9956
SVM	0	1.0000	1.0000
	1	1.0000	0.9885
	2	0.9938	1.0000
	avg	0.9938	0.9962

TABLE XIII

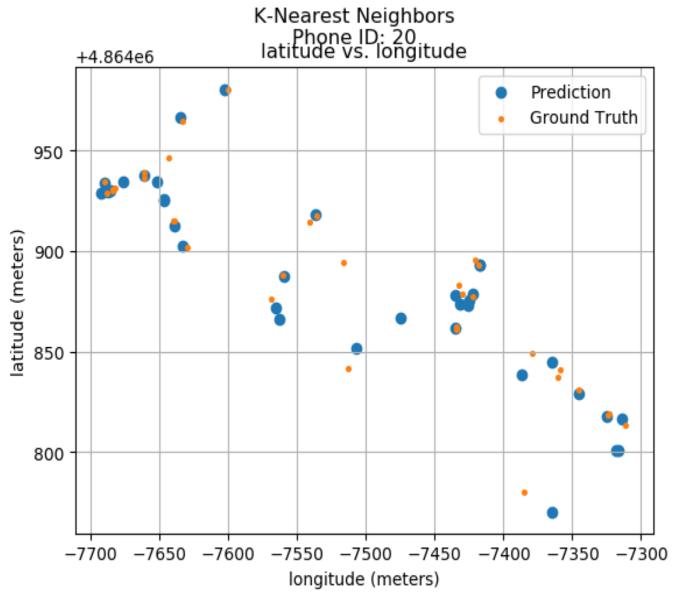

Fig. 8. KNN Prediction - Latitude vs. Longitude

VI. Conclusion/Future Work

The best results were given by the K-Nearest Neighbor model. This was due to the high number of overlapping measurements in this dataset. The Decision Tree and Random Forest models also preformed relatively well, but the Support Vector Machine model preformed awfully. This is likely due to the fact that the data was not very separable no matter the kernel used. Unfortunately, the UjiIndoorLoc dataset does not seem like a reliable or even practical dataset to approach this problem in our opinion. The statistical learning assumption violation between the two provided datasets, the high number of overlapping measurements while also leaving much of the map unexplored, and the lack of reliable temporal data have all caused concern for how useful a dataset like this would be for a practical WiFi fingerprinting localization application.

There are many use cases for reliable WiFi fingerprinting for localization and there are many ways to improve the results

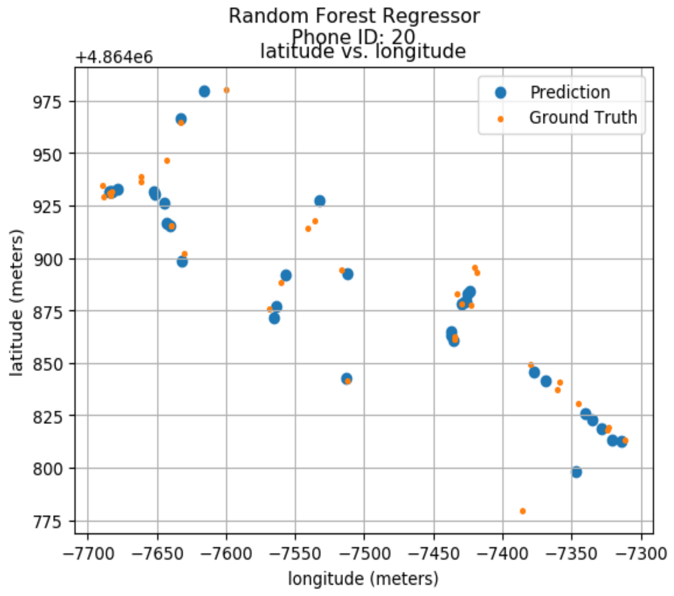

Fig. 9. RF Prediction - Latitude vs. Longitude

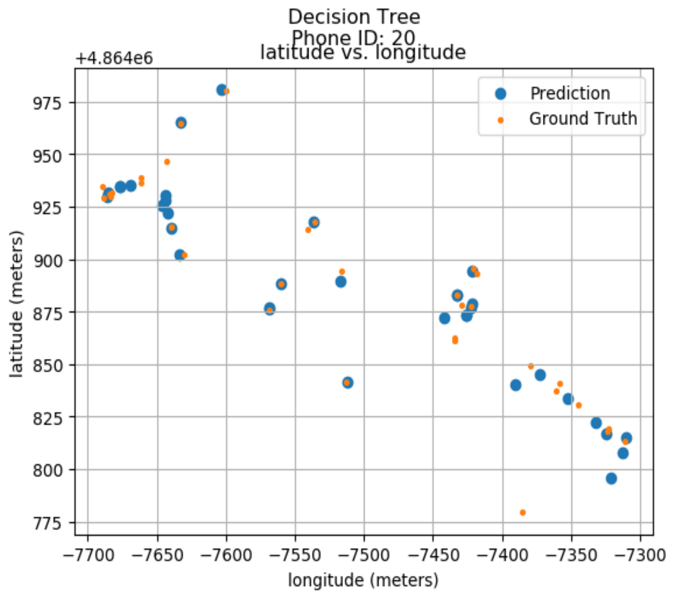

Fig. 10. DT Prediction - Latitude vs. Longitude

given in this paper. The first and most obvious improvement would be to find or create a better dataset for this problem. From here, a temporal localization approach could be used if reliable temporal data is available such as in Hong et. al's particle filter approach [4]. Another approach would be to localize the individual WAPs and optimize their locations by minimizing the error between a set of ground truth coordinates and the predicted coordinates generated from an intensity versus distance fit.

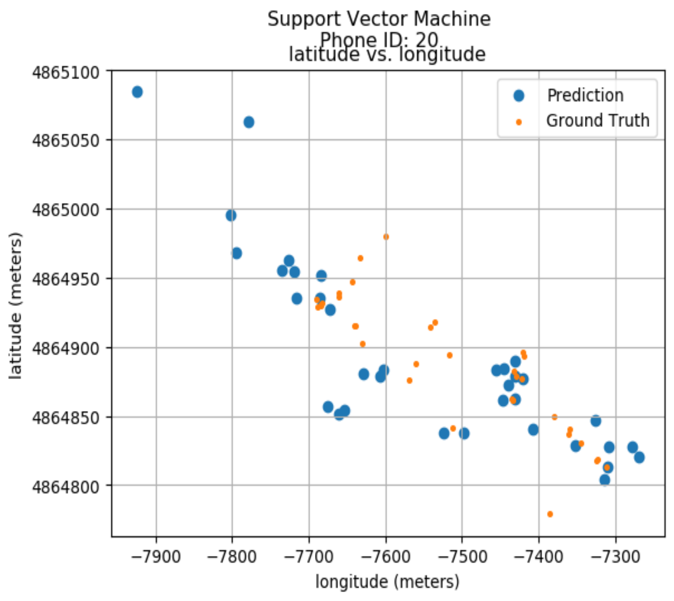

Fig. 11. SVM Prediction - Latitude vs. Longitude

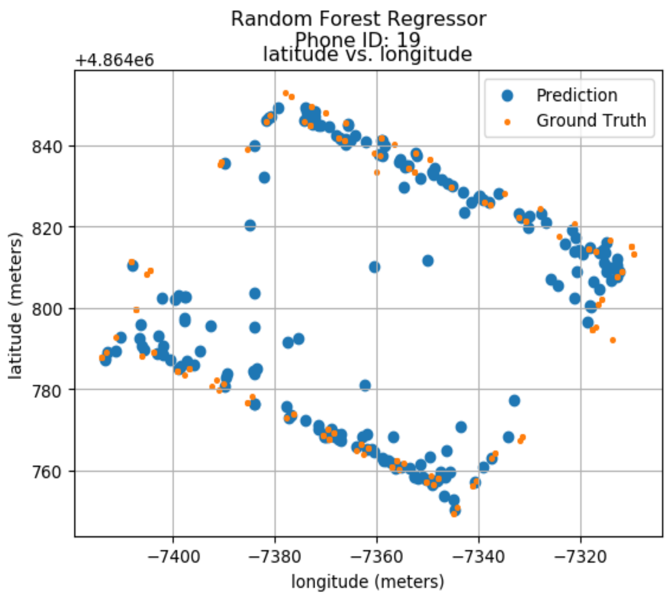

Fig. 13. RF Prediction - Latitude vs. Longitude

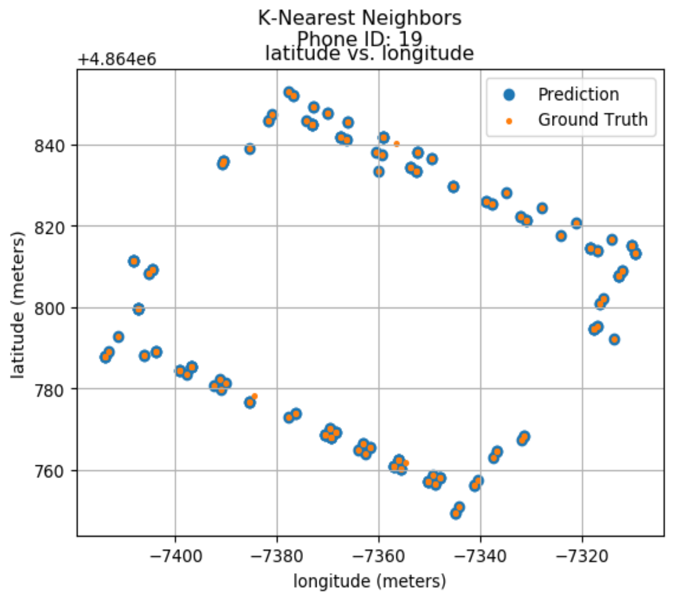

Fig. 12. KNN Prediction - Latitude vs. Longitude

VII. Contributions

Ryan contributed to the final report, the code template for the project, the analysis plots, parameter tuning, the K-Nearest Neighbors model, the Random Forest model, and the Support Vector Machine model. Matt contributed to the poster, the mid-project presentation, the code template for the project, parameter tuning, the Random Forest model, and the Decision Tree model. So contributed to the final report, the mid project presentation, the code template, parameter tuning, the Decision Tree model, and a Neural Network model that was not used do to not being compatible with the relatively small dataset used. Sophia contributed to the final report, the Support Vector Machine model, the Decision Tree model, and some of the result metrics.

References

[1] J. Torres-Sospedra et al., ”UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems,” 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, 2014, pp. 261-270. doi: 10.1109/IPIN.2014.7275492

[2] Joaqun Torres-Sospedra Raul Montoliu Adolfo Martinez Joaquin Huerta. UJI - Institute of New Imaging Technologies, Universitat Jaume I, Avda. Vicente Sos Baynat S/N, 12071, Castelln, Spain. UPV - Departamento de Sistemas Informticos y Computacin, Universitat Politcnica de Valncia, Valencia, Spain. https://archive.ics.uci.edu/ml/datasets/ujiindoorloc

[3] R. Montoliu, J. Blom, and D. Gatica-Perez, Discovering places of interest in everyday life from smartphone data, Multimedia Tools and Applications, vol. 62, no. 1, pp. 179207, 2013.

[4] Hong, Feng Zhang, Yongtuo Zhang, Zhao Wei, Meiyu Feng, Yuan Guo, Zhongwen. (2014). WaP: Indoor localization and tracking using WiFi-Assisted Particle filter. Proceedings - Conference on Local Computer Networks, LCN. 210-217. 10.1109/LCN.2014.6925774.

[5] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12. 2825–2830. 2011.

[6] EvAAL. EvAAL-ETRI on-site and off-site Indoor Localization Competition In conjunction with IPIN 2015 Banff, Alberta, Canada. 2015. http://evaal.aaloa.org/2015/competition-home

[7] R Berkvens, M Weyn, H Peremans. Mean Mutual Information of Probabilistic Wi-Fi Localization. 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN) 1-4. 2015

[8] Ho Jin Ju, Min Su Lee, Chan Gook Park, Soyeon Lee, Sangjoon Park. Advanced Heuristic Drift Elimination for indoor pedestrian navigation. 2014 International Conference on Indoor Positioning and Indoor Navigation(IPIN). 2014

[9] Sinem Bozkurt, Gulin Elibol, Serkan Gunal, Ugur Yayan. A Comparative Study on Machine Learning Algorithms for Indoor Positioning. 2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA). 2015

[10] M. Ali Aydin, Ahmet Sertbas, Tulin Atmaca. A Comparative Analysis of N-Nearest Neighbors (N3) and Binned Nearest Neighbors (BNN) Algorithmsfor Indoor Localization. Computer Networks. CN 2017. Communications in Computer and Information Science, vol 718. 2017

[11] Nowicki M., Wietrzykowski J. Low-Effort Place Recognition with WiFi Fingerprints Using Deep Learning. Automation 2017, ICA 2017, Advances in Intelligent Systems and Computing, vol 550. 2017

[12] Micha Nowicki, Jan Wietrzykowski, Piotr Skrzypczyski. Adopting Learning-based Visual Localization Methods for Indoor Positioning with WiFiFingerprints. Learning Applications for Intelligent Autonomous Robots. 2018