WiFi Indoor Localization

James Clark
UMass Lowell
James_Clark@student.uml.edu

Flore Norceide
UMass Lowell
Flore_Norceide@student.uml.edu

Prism Prajapati
UMass Lowell
Prism_Prajapati@student.uml.edu

Oskar Barrera
UMass Lowell
oskar_barreradazzo@student.uml.edu

***Abstract*—This project aims to examine the differences between different Machine Learning Algorithms in yielding a model for predicting classification of some attribute based on the values of the other attributes. In this case of the project, the predicted attribute is the indoor localization of different routers based on their signal. The data set contains 8 different columns. The first seven correspond to the signal from the seven router signals, and the last column contains the classification values with each number representing a specific room.**

I. Introduction

In a world of ever-increasing surveillance, it can be useful to have known methods from which the exact location of an entity can be determined, in a bound region. For instance, having the ability to accurately identify the physical position of a prisoner in a secure building with minimal margin of error gives confidence that a hostile target could be located and subdued without a violent altercation. Another unexpected attack is in the field of economics, particularly in dynamically targeted advertising tailored to a particular individual based on their local position relative to some well-established way point. In general, wireless sensors are not limited to a radio signal strength. More advanced wireless sensors provide attribute information which can be used, with an additional temporal dimension, to draw inferences about the environment that can be used to make new types of classification predictions [1].

In any classification application, where the independent attribute vector represents sensor readings, the goal is to develop a model that yields the most accurate classification of unseen data instances. The goal of this study is to examine the efficiency of well-established machine learning classification algorithms in predicting one in four possible locations of a signal reading, based on signal measurements of 7 wireless router signal attributes.

To predict the user's location accurately, a definite and consistent model has to be trained and deployed in a tracking or monitoring device. In the paper examined, Wi-Fi signal strength received from various routers in a bounded location were used and a neural network model was trained so that it could further predict the user's location for an unknown tuple set having signal strengths [1].

The goal of our study is to make reproducible comparison in performance of various classification machine learning algorithms, ultimately analyzing the accuracy predicting the classification of new data instances.

The data set is freely available, and can be found at: https://archive.ics.uci.edu/ml/datasets/Wireless+Indoor+Localization

It's original form is a .txt file and was converted to a .csv file and loaded into a Jupyter Notebook for study.

II. Current State of the Art

The authors of the original paper procured data of signal strengths from various routers, mapped them to a user's location and considered this mapping as a classification data set. A neural network was trained using the weights obtained by the proposed fuzzy hybrid of Particle Swarm Optimization and Gravitational Search Algorithm (FPSOGSA), The non-fuzzy PSOGSA algorithm is intended to eliminate bias by considering a set of candidate solutions at each step of the training process, and to eliminate the possibility of becoming trapped in a local minimum during training. Even with employing the PSOGSA optimization, training the neural network can be known to overfit the data. The study attempts to overcome this by introducing a fuzzy component to the calculations made in the particle swarm optimization, which can help the abstracted particles used in training calculations explore new search spaces where they would not otherwise be able to do. [1].

The data is clean, easy to access, and manipulate, and does not require extensive pre-processing. There are no incomplete or missing data, and no outliers or noisy data instances. The pre-processing requirements were minimal. To be specific: the independent attribute vectors were originally negative integer values and converted to positive for readability and usability. (The generation and training of the various machine learning model types examined will generally be unphased when it comes to using positive vs. negative independent attribute vectors.) Further pre-processing involved converting the .txt format to .csv and, in some cases, normalization of the independent attribute vector and conversion of the target variable to a categorical attribute vector.

III. Approach

The goal of this study is to explore the operational and performance aspects of various classification machine learning

algorithms with our dataset. Each team member was tasked with implementing their own model construction and training, along with measuring performance, of a particular classification machine learning algorithm. In each case, python code was assembled to generate a classification model, and measure performance in classifying data instances. The following sections detail each team member's individual results. The final section makes an inter-model comparison of performance and attempts to draw inferences regarding the comparison of the different classification machine learning algorithms.

IV. Overview of Data

The data set is a collection of instances of WiFi radio signal sensor measurements, where the measurement is performed in one of four rooms. The data set is represented with eight columns. The first seven columns represent the WiFi signal strengths of the seven respective routers. The last column represents the independent attribute, which is one of four discrete locations from which the signal strength measurements are made. The first few lines of the data are shown in Figure 1 below.

	Sensor1	Sensor2	Sensor3	Sensor4	Sensor5	Sensor6	Sensor7	Room
0	64	56	61	66	71	82	81	1
1	68	57	61	65	71	85	85	1
2	63	60	60	67	76	85	84	1
3	61	60	68	62	77	90	80	1
4	63	65	60	63	77	81	87	1

Fig. 1. The first few lines of the data file

Figure 2 shows some statistical metadata about the distribution of the data instances' attribute vector components.

	Sensor1	Sensor2	Sensor3	Sensor4	Sensor5	Sensor6	Sensor7	Room
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	52.330500	55.623500	54.964000	53.566500	62.640500	80.985000	81.726500	2.500000
std	11.321677	3.417688	5.316186	11.471982	9.105093	6.516672	6.519812	1.118314
min	10.000000	45.000000	40.000000	11.000000	36.000000	61.000000	63.000000	1.000000
25%	46.000000	53.000000	51.000000	46.000000	56.000000	77.000000	78.000000	1.750000
50%	55.000000	56.000000	55.000000	56.000000	64.000000	82.000000	83.000000	2.500000
75%	61.000000	58.000000	58.000000	63.000000	69.000000	86.000000	87.000000	3.250000
max	74.000000	74.000000	73.000000	77.000000	89.000000	97.000000	98.000000	4.000000

Fig. 2. The overall properties of the file

V. Breakdown of Contents

- Neural Network Algorithm developed by Clark
- Naive Bayes and Decision Tree Classifier Algorithms developed by Norceide
- kNN Algorithm developed by Barrera
- Random Forest Algorithm developed by Prajapati

VI. Neural Network, James Clark

In the context of the indoor localization, many experimental approaches have been employed to in attempt to construct an optimal model. Classical approaches, such as KNN, SVM, or Naïve bayes classifier, are known to involve complex tuning of parameters, large computational resource and time requirements, and have not the best accuracy when it comes to predicting classification for new data instances [2]. Deep neural networks have been proven efficient, but may not scale well. It is proven that a Deep Neural Network for WiFi indoor localization can yield an accuracy of about 85%, and take a negligible amount of time to train [3].

Artificial Neural Networks are often considered to be non-intuitive when it comes to understanding their training and overall computational process. In practice, neural networks have proven to be invaluable in classification and forecasting applications. Neural networks perform well for classification problems where the attribute vector components may not be linearly separable. However, classical training methods such as Back Propagation impose a bottleneck on the feasible application of neural networks.

In the paper used as base for this project, a study was conducted to examine the effect of the particle swarm optimization and global search algorithm (PSOGSA) on the convergence to the global optimum in training. The algorithm is intended to eliminate bias by considering a set of candidate solutions at each step of the training process, and to to eliminate the possibility of becoming trapped in a local minima during training. Even with employing the PSOGSA optimization, training the neural network can be known to overfit the data. It attempts to overcome this by introducing a fuzzy component to the calculations made in the particle swarm optimization, which can help the abstracted particles used in the optimization calculation explore new search spaces where they would not otherwise be able to do [1].

Another study in the indoor localization classification based on WiFi signal data was performed with, the intent of examining the efficiency of a CNN coupled with Gaussian process regression in predicting position. The study proved that the accuracy of the model can be increased with the additional convolution layers in the network, and the Gaussian process regression. However, it does add significant processing overhead in that the feature extraction kernels must be carefully crafted, to feed to the inner layers of the network [13].

In this study, three traditional feed-forward artificial feed-forward neural network models were constructed, trained, and analyzed to gain insight into overall performance of the model [12]. The inputs to the network are the seven wireless router signal strengths, and the output is the discrete classification of one of four rooms where the signal readings are measured. The study examines the effect of network structure and training optimizations on the accuracy and training aspects of the neural network. Later, we compared the overall performance of the neural network models to the other classification algorithms that were studied by the remaining team members.

Additionally, for each of the constructed networks, there is a classification report and corresponding ROC curves [11] for each of the four discrete target values. Figures below show the performance metrics and visualization for the first network model constructed.

Three different neural networks were constructed and examined. The characteristics of all the models can be seen from the screen-capture showing the Python code used in constructing the model. See figure 3 below:

```
#build the neural network
model = Sequential()
#network 1

#model.add(Dense(64, input_dim=7, kernel_initializer='random_normal', activation='relu'))
#model.add(Dense(4, activation='sigmoid'))
#model.add(Dense(4, activation='sigmoid'))

#network 2

#model.add(Dense(64, input_dim=7, kernel_initializer='random_normal', activation='relu'))
#model.add(Dense(32, activation='relu'))
#model.add(Dense(16, activation='relu'))
#model.add(Dense(8, activation='relu'))
#model.add(Dense(4, activation='sigmoid'))


#network 3

model.add(Dense(64, input_dim=7, kernel_initializer='random_normal', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(4, activation='sigmoid'))

model.summary()
model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=0.0003), metrics=['mse', 'mae',
history = model.fit(X_train, Y_train, epochs=75, batch_size=100, verbose=1, validation_split=0.2)
```

Fig. 3. Networks Characteristics

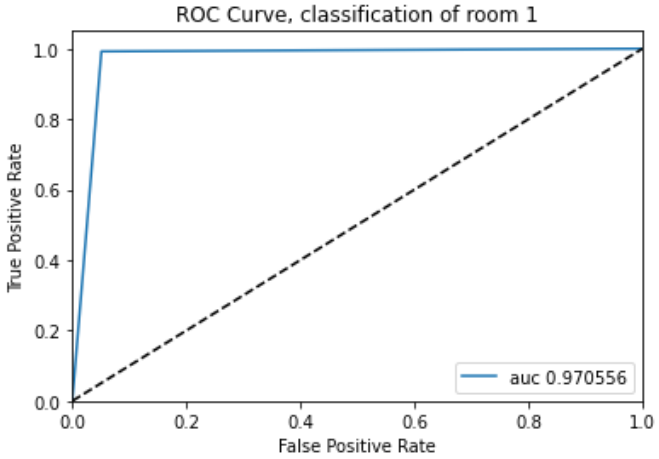

Fig. 4. ROC Curve of Room 1 for NN 1

	precision	recall	f1-score	support
0.0	1.00	0.95	0.97	372
1.0	0.87	0.99	0.93	128
accuracy			0.96	500
macro avg	0.93	0.97	0.95	500
weighted avg	0.96	0.96	0.96	500

Fig. 5. Classification report of Room 1 for NN 1

The neural network study examines the training convergence, accuracy, and bias of three neural network models.

The first model is a shallow network, with 1 hidden layer and no dropout. A training of 75 epochs, using a batch size

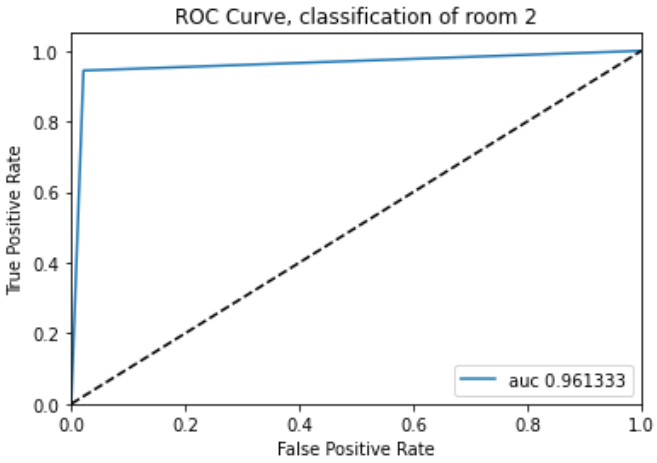

Fig. 6. ROC Curve of Room 2 for NN 1

	precision	recall	f1-score	support
0.0	0.98	0.98	0.98	375
1.0	0.94	0.94	0.94	125
accuracy			0.97	500
macro avg	0.96	0.96	0.96	500
weighted avg	0.97	0.97	0.97	500

Fig. 7. Classification report of Room 2 for NN 1

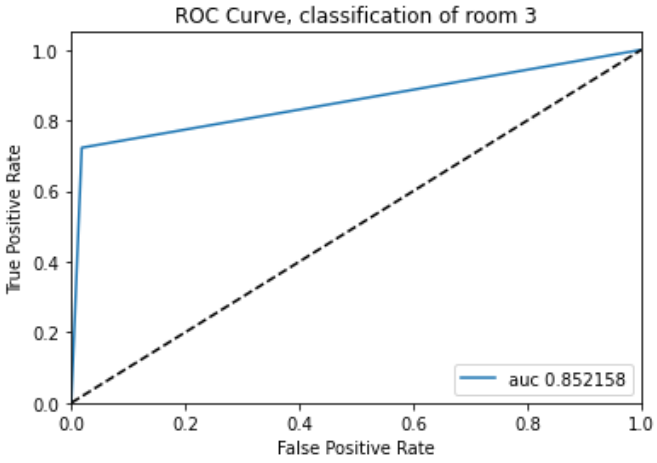

Fig. 8. ROC Curve of Room 3 for NN 1

	precision	recall	f1-score	support
0.0	0.92	0.98	0.95	381
1.0	0.92	0.72	0.81	119
accuracy			0.92	500
macro avg	0.92	0.85	0.88	500
weighted avg	0.92	0.92	0.92	500

Fig. 9. Classification report of Room 3 for NN 1

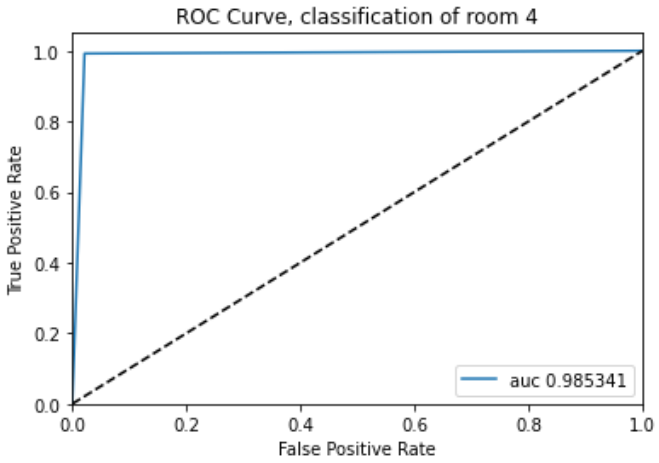

Fig. 10. ROC Curve of Room 4 for NN 1

	precision	recall	f1-score	support
0.0	1.00	0.98	0.99	372
1.0	0.94	0.99	0.97	128
accuracy			0.98	500
macro avg	0.97	0.99	0.98	500
weighted avg	0.98	0.98	0.98	500

Fig. 11. Classification report of Room 4 for NN 1

of 100, yields an accuracy of about 95% in the validation set. There is little difference in the accuracy of the training data and the validation data, indicating no imposing overfitting or underfitting of the training data. The accuracy vs epoch plot shows a logarithmic increase of accuracy with respect to the epoch, suggesting a relatively normal training process. This is to be expected, considering the model has only 1 hidden layer with no dropout.

The second model is a deeper network, identical to the first model but with 4 hidden layers instead of 1. The training of 75 epochs and a batch size of 100 yields an accuracy of about 93% in the validation set. Similar to the first model, the second model shows little difference in accuracy between the training data and validation data. This would point to the fact that there is no indication of overfitting or underfitting of the training data. The accuracy vs. epoch plot shows a sharp increase in accuracy in the first few epochs, followed by a period of almost no increase in accuracy, and then a sudden convergence toward the global optimum. This can be attributed to training process with respect to the network structure, in that the training optimizer has some initial hiccups where it could hit a local optimum, and require some epochs of training before it can break out of the local optimum and continue to converge toward the global optimum. The convergence to 93% accuracy suggests that the deeper network converges to a comparable accuracy rate to model 1.

The final model is identical to the second model, but with dropout added between the input and first hidden layer, and dropout added between the last hidden layer and the output layer. The same training of 75 epochs with a batch size of 100 yielded an accuracy of about 83%. This model shows that the model is almost 15% more accurate in predicting the validation data than the training data. This means that the model has underfit the training data, that can be attributed to the dropout added. The accuracy vs epoch graph shows a logarithmic increase of model accuracy vs training epoch, indicating a normalized training process resulting from the dropout added.

Figures 12 and 13 respectively show the visual representation of comparing the accuracy and model loss of the train and validation parameters.

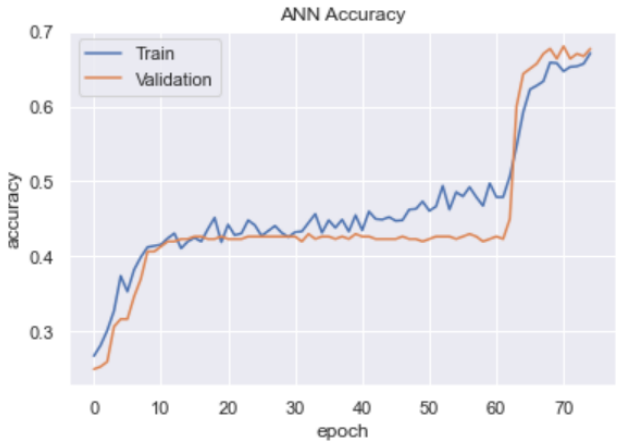

Fig. 12. ANN accuracy of the Train versus the Validation

Fig. 13. Model loss representation of Train versus Validation

VII. Bayes Classifier, Flore Norceide

Many techniques have been introduced to increase the accuracy of the localization system. Bayesian learning techniques are considered much accurate for localization but still there are some issues including zero probability and good accuracy. Some researchers have used a unique weighting

technique called improved multinomial Naive Bayes technique for localization. It provides better accuracy,resolves zero probability issue caused due to data incompleteness. It also somehow tackles with naïve Bayes issue of independencies that according to Navies Bayes all the features are independent of each other [4].

In term of my approach, instead of using the improved method used in the paper cited above, I looked at the Bayesian algorithm. To perform the localization in my case, the Bayes rule is used to find posterior probability. Meaning that given some received signal strength value what the probability of being in some location is. Naïve Bayes algorithm has conditional independence assumption for received signal strength values which the aforementioned paper does not.

I partitioned the data so that the sensor value would be stored in one variable and the classification values would be stored in another. Once that partition is done, the data was separated into a 20-80% test-train split.

Once the values were set, I did a simple Gaussian Naive Bayes fit for my model. I then ran a prediction and created a confusion matrix to observe the accuracy of the model. The resulting confusion matrix is shown in the figure below.

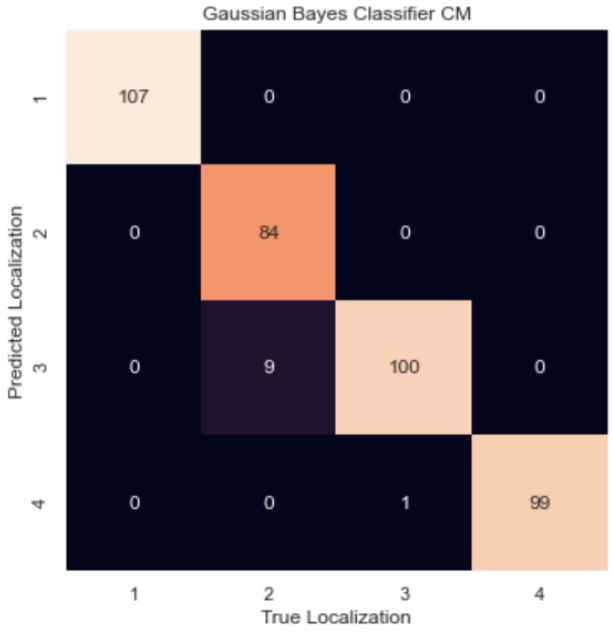

Fig. 14. The confusion matrix from the Gaussian Bayes model

A easier way to visualize the performance of the model is to look at the classification report. Figure 15 below shows the generated report.

VIII. Decision Tree Classifier, Flore Norceide

The second model that I decided to take a look at was the Decision Tree Classifier.

Decision Trees are versatile Machine Learning algorithms that can that can perform classification tasks. They are powerful algorithms, capable of fitting complex datasets [6].

	precision	recall	f1-score	support
1	1.00	1.00	1.00	107
2	1.00	0.90	0.95	93
3	0.92	0.99	0.95	101
4	0.99	1.00	0.99	99
accuracy			0.97	400
macro avg	0.98	0.97	0.97	400
weighted avg	0.98	0.97	0.97	400

Fig. 15. The classification report from the Gaussian Bayes model

The intuition behind Decision Trees is that you use the dataset features to create yes/no questions and continually split the dataset until you isolate all data points belonging to each class.

The ideal tree is the smallest tree possible, i.e. with fewer splits, that can accurately classify all data points. In the model I trained, I experienced with different depth of trees and as stated above, indeed the smaller tree provided better accuracy.

When picking the split, the algorithm tries to divide the dataset into the smallest subset possible. Just like any other Machine Learning algorithm, the goal is to minimize the loss function as much as possible. One way to do so, is to find a loss function that compares the class distribution before and after the split, like Gini Impurity and Entropy, which is examined in the model built [5].

Although the Decision Tree seems like a very simple algorithm, it comes with various advantages that make it worthwhile considering when doing a classification.

- The tree can be easily interpreted as it can be visualized
- There is no pre-processing of the data required
- The algorithm is easily able to handle different types of data.

In this section, I train, do predictions on the data and visualize different decision trees.

The initial model I trained was a decision tree classifier where I did not specify the maximum depth. As a result, it generated a decision tree with multiple levels. When analyzing the results, I noticed the accuracy was a bit low and i adjusted my parameters and specified a maximum depth of five.

The initial decision tree generated using 'graphviz' is below in figure 1.

Once I realized what the optimal maximum depth was, i created a new model and trained it with a depth of 5. The tree that was generated is shown in figure 17

With that new model generated, the accuracy was much higher as can be seen from figure 18 below:

The confusion matrix shows the number of true versus predictable values guessed. Figure 19 shows our generated confusion matrix for the decision tree.

I also wanted to take a look at the actual count of values that were correctly classified using our model. For that reason, I decided to create two histogram on the same axis. One of them represented the count of predicted classes and the other

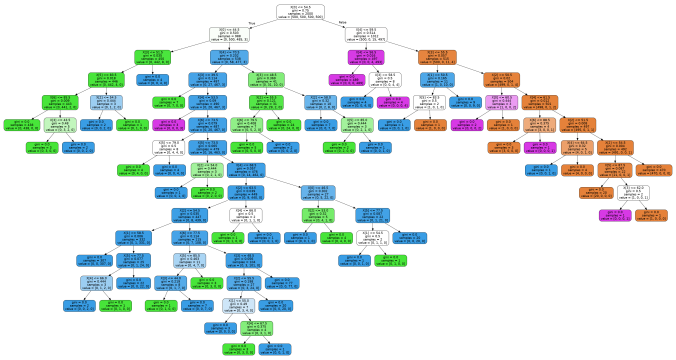

Fig. 16. The Decision Tree using the default depth

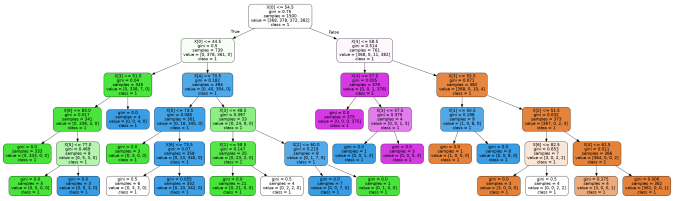

Fig. 17. The Decision Tree using depth five

	precision	recall	f1-score	support
1	0.99	0.98	0.99	133
2	0.93	0.97	0.95	116
3	0.96	0.90	0.93	136
4	0.97	0.99	0.98	115
accuracy			0.96	500
macro avg	0.96	0.96	0.96	500
weighted avg	0.96	0.96	0.96	500

Accuracy: 0.962

Classification Report:

Fig. 18. Classification report of Decision Tree

was the classification from our test data. Looking at figure 20, we can see that a very high count of values were identified.

IX. KNearestNeighbor Classifier, Oskar Barrera

The KNN (K-NearestNeighbor) classification algorithm should be considered as the simplest classification algorithm in machine learning. Before sorting, the KNN classifier will read a large amount of sample data with sort labels as reference data for sorting. When classifying samples with unknown categories, it shall calculate the difference between the current sample and all reference samples; The size of the difference is measured by the distance of the data points in the multidimensional space of the sample characteristics, That is, if the two sample points are closer in the multidimensional space of their characteristic data, the smaller the difference between them, the more likely these two sample points belong to the same category [7].

In practical use, the objective is to obtain the useful location classification of some types of sensor reading. This research is able to evaluate the efficiency of the KNN classification algorithm and other types of automatic learning to predict the location of the sensor reader in an indoor environment, based on the 7 attributes of the wireless router signal.

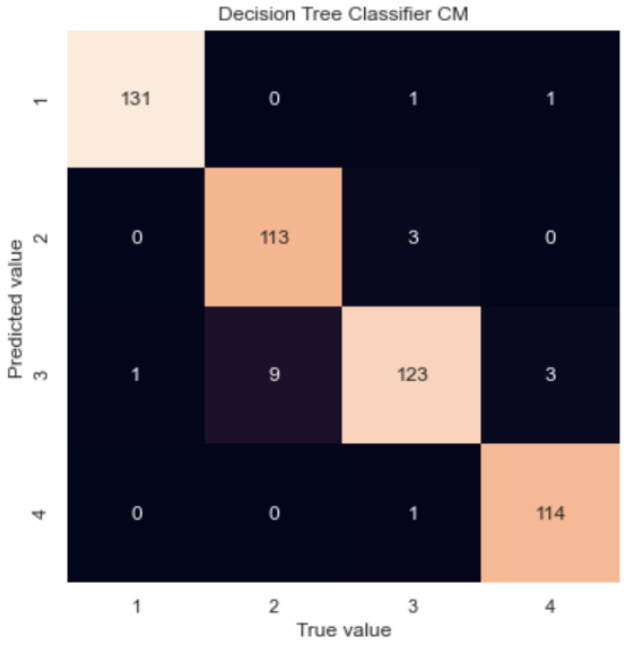

Fig. 19. Confusion matrix of Decision Tree

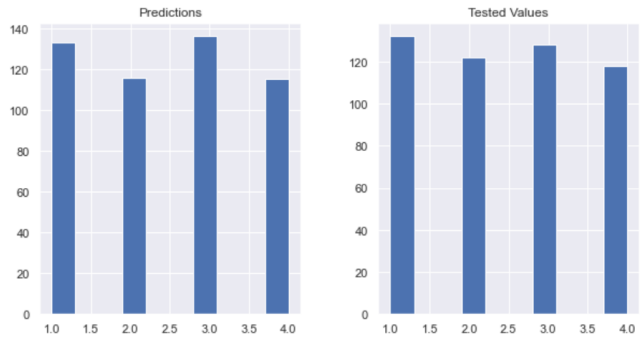

Fig. 20. Comparing predicted and test values

Figure 21 below shows the histogram distribution of each of the different columns in the data. It is not related to model construction but serves as a good way to visualize the distribution of each variable.

The value of our KNN classifier is defined by giving it the value of K in 7 and this yields a pretty high accuracy as can be seen from figure 22.

The accuracy obtained is 99% for the training set and 98% for the test set, and we use the KNeighborsClassifier of SciKit Learn since our labels are discrete values. Then the model is precise and the accuracy is confirmed by seeing the Matrix Confusion and the Report on the set of tests, which details the successes and failures. Refer to figure 23:

The F1 score obtained is 98%, giving an excellent result. We then try to modify the model by looking for the best value of k since the value n neighbors=7 was previously assigned as the value of “k” with good results.

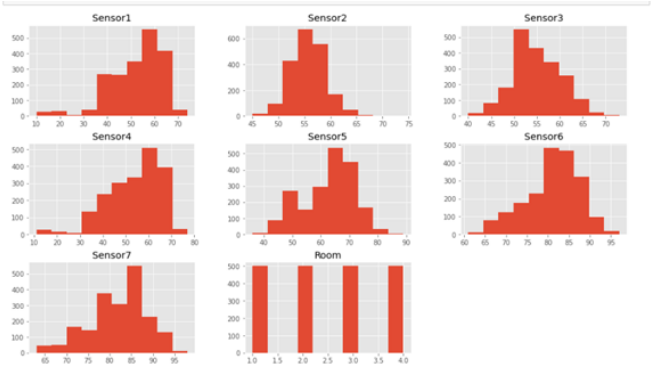

Fig. 21. The Decision Tree using depth five

```
n_neighbors = 7

knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.99
Accuracy of K-NN classifier on test set: 0.98
```

Fig. 22. The Decision Tree using depth five

```
[[131   0   1   0]
 [  0 113   9   0]
 [  0   1 126   1]
 [  0   0   0 118]]
              precision    recall  f1-score   support

           1       1.00      0.99      1.00       132
           2       0.99      0.93      0.96       122
           3       0.93      0.98      0.95       128
           4       0.99      1.00      1.00       118

    accuracy                           0.98       500
   macro avg       0.98      0.98      0.98       500
weighted avg       0.98      0.98      0.98       500

[[131   0   1   0]
 [  0 113   9   0]
 [  0   1 126   1]
 [  0   0   0 118]]
              precision    recall  f1-score   support

           1       1.00      0.99      1.00       132
           2       0.99      0.93      0.96       122
           3       0.93      0.98      0.95       128
           4       0.99      1.00      1.00       118

    accuracy                           0.98       500
   macro avg       0.98      0.98      0.98       500
weighted avg       0.98      0.98      0.98       500
```

Fig. 23. The Decision Tree using depth five

In the graph we see that with values k = 7 to k = 14 is where greater precision is achieved.

X. Random Forest Classifier, Prism Prajapati

One of the most popular algorithm in machine learning is random forest that is used in both classification and regression.

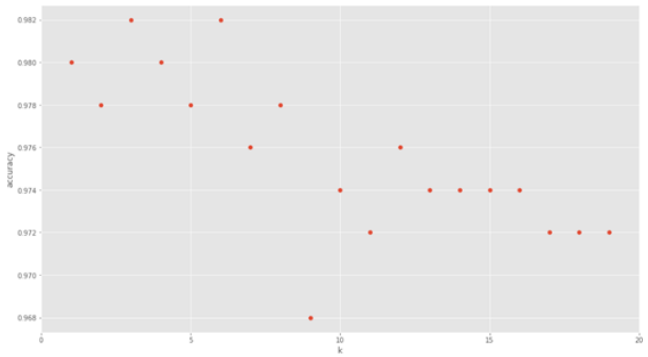

Fig. 24. The Decision Tree using depth five

The ability to perform both tasks makes it unique, and enhances its wide-spread usage across a myriad of applications. It also assures high accuracy most of the time, making it one of the most sought-after classification algorithms. Random Forests are comprised of decision tree. Random forest is a supervised machine learning algorithm that is used widely in classification and regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. Random forest algorithm can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems[prismref1].

Our data set has 2000 rows and 8 columns. This data set contains Wi-Fi signal strength observed by using 7 different Wi-Fi devices on a smartphone which can collect data from the indoor space. For this particular situation, the use of Random Forest tree to predict the room number where the devices are located by detecting different Wi-Fi sensors.

Using the Random Forest tree, we were able to predict 98% of accuracy. Our model is precise and accurate after analyzing the output from the matrix confusion and the report from the set of tests.The accuracy for the generated model can be seen from 25 showing a screenshot from the Jupyter Notebook.

```
# Importing accuracy score
from sklearn.metrics import accuracy_score
accuracy_score(t_test, rf_Predicted)
```

```
0.984
```

Fig. 25. Accuracy of random forest

The classification report and confusion matrix of the model showing the details on the test success and failure rate can be seen in fig. 26 and fig. 27 respectively.

Our study examines various classification type machine learning algorithms in arriving at a model which can accurately predict a discrete location value based on an independent attribute vector representing WiFi radio signal strengths. The

	precision	recall	f1-score	support
1	0.99	0.99	0.99	123
2	0.99	0.98	0.99	132
3	0.98	0.96	0.97	121
4	0.97	1.00	0.98	124
accuracy			0.98	500
macro avg	0.98	0.98	0.98	500
weighted avg	0.98	0.98	0.98	500

Fig. 26. Classification report of random forest

algorithm types that were studied include Naïve Bayes Classifier, Decision Tree Classifier, k-Nearest-Neighbor Classifier, Random Forest Classifier, and Artificial Neural Network. An inter-model comparison of accuracy shows similar results in all algorithms, in terms of overall accuracy convergence. This most likely indicates a high bias attained resulting from overfitting the data set. Had we allocated data instanced to be used solely as validation data, or used cross-validation methods for training, a possibility is that the overall accuracy of the various models would have stark differences from what we attained by our training methods. In the realm of our data set, and the accuracy metric obtained for each algorithm, it is evident that most classification style machine learning algorithms can perform comparatively well. This can be attributed to the lower dimensional relationships in our data set. The models that were constructed may not scale well to larger data sets, or data sets with higher dimensional relationships. However, in the context of our dataset, the models are fittingly accurate.

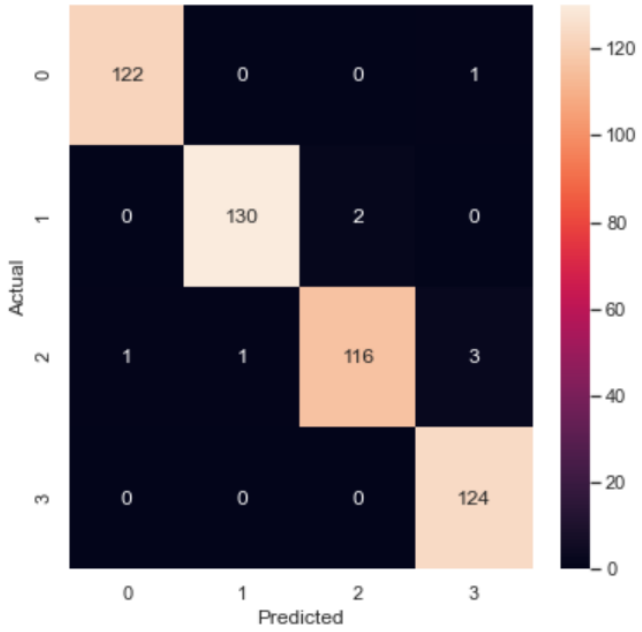

Fig. 27. Confusion Matrix of random forest

References

[1] Rohra, J., Perumal, B., Narayanan, S., Thakur, P. and Bhatt, R., 2017. User Localization in an Indoor Environment Using Fuzzy Hybrid of Particle Swarm Optimization Gravitational Search Algorithm with Neural Networks. [ebook] Available at: ¡https://link.springer.com/chapter/10.1007/978-981-10-3322-3_27¿ [Accessed 10 February 2022].

[2] X. Song et al., "A Novel Convolutional Neural Network Based Indoor Localization Framework With WiFi Fingerprinting," in IEEE Access, vol. 7, pp. 110698-110709, 2019, doi: 10.1109/ACCESS.2019.2933921.

[3] BelMannoubi, S. and Touati, H., 2019. Deep Neural Networks for Indoor Localization Using WiFi Fingerprints. [ebook] Available at: ¡https://doi.org/10.1007/978-3-030-22885-9_21¿ [Accessed 18 April 2022].

[4] Sohail, A., Aziz Ul Haq, M., Kamboh, H., Akram, U. and Iram, H., 2017. Indoor Localization Using Improved Multinomial Naïve Bayes Technique. [ebook] Available at: ¡https://link.springer.com/chapter/10.1007/978-3-319-60834-1_32¿ [Accessed 20 April 2022].

[5] Bento, C., 2021. Decision Tree Classifier explained in real-life: picking a vacation destination. [online] Towards Data Science. Available at: ¡https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575¿ [Accessed 19 April 2022].

[6] Géron, A., 2020. Hands-on machine learning with Scikit-Learn, Keras and TensorFlow. Beijing: O'Reilly.

[7] datagy. 2022. K-Nearest Neighbor (KNN) Algorithm in Python • datagy. [online] Available at: ¡https://datagy.io/python-knn/¿ [Accessed 22 April 2022].

[8] E. R., S., 2021. Random Forest — Introduction to Random Forest Algorithm. [online] Analytics Vidhya. Available at: ¡https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/: :text=Random%20forest%20is%20a%20Supervised,average%20in%20case%20of%20regression¿ [Accessed 21 April 2022].

[9] Rohra, Jayant Perumal, Boominathan J.N., Swathi Thakur, Priya Bhatt, Rajen. (2017). User Localization in an Indoor Environment Using Fuzzy Hybrid of Particle Swarm Optimization Gravitational Search Algorithm with Neural Networks. 10.1007/978-981-10-3322-3_27.

[10] BelMannoubi, S., Touati, H. (2019). Deep Neural Networks for Indoor Localization Using WiFi Fingerprints. In: Renault, É., Boumerdassi, S., Leghris, C., Bouzefrane, S. (eds) Mobile, Secure, and Programmable Networking. MSPN 2019. Lecture Notes in Computer Science(), vol 11557. Springer, Cham. https://doi.org/10.1007/978-3-030-22885-9_21

[11] PiQuestions. 2019. Python: ROC for multiclass classification. [online] Available at: ¡https://pyquestions.com/roc-for-multiclass-classification¿ [Accessed 16 April 2022].

[12] tutorials, P., 2020. Keras: Regression-based neural networks — DataScience+. [online] Datascienceplus.com. Available at: ¡https://datascienceplus.com/keras-regression-based-neural-networks/¿ [Accessed 18 April 2022].

[13] Zhang G, Wang P, Chen H, Zhang L. Wireless Indoor Localization Using Convolutional Neural Network and Gaussian Process Regression. Sensors (Basel). 2019;19(11):2508. Published 2019 May 31. doi:10.3390/s19112508

APPENDIX

XI. TASK DISTRIBUTION AMONG TEAM MEMBERS

A. James Clark

- Data acquisition
- Neural Network Modeling and generated writing for said section
- Analyzed cross-model similarities and differences
- Conclusion and Introduction

B. Flore Norceide

- Naive Bayes Modeling and generated writing for said section
- Decision Tree Modeling and generated writing for said section
- State of the art, approach, overview
- Report Translation to LaTex
- Creating notebook and incorporating all codes
- Wrote readme file included with code submission

C. Oskar Barrera

- kNN Modeling and generated writing for said section
- Clustering research for classification
- Linear and Polynomial regression research

D. Prism Prajapati

- Random Forrest Modeling and generated writing for said section
- Presentation Slides Overview