EE 599 Mathematical Pattern Recognition Project

WIRELESS INDOOR LOCALIZATION

Akshay Kenchappa Manjunath
kakshay@usc.edu

May 8, 2020

Data Set used: Wireless Indoor Localization data set.

1. Abstract

The Wireless Indoor Localization problem comprises of using WiFi signal strengths to determine the indoor location of a user. The dataset consists of measured signal strength in dB at seven Wifi Routers (WS1-WS7) and four user locations. We employ the Wireless Indoor Localization data set to predict the location of a user who is in one of four rooms based on the WiFi signal strength received from seven routers. The dataset was divided into two sets as training data and validation data with 80% of the data kept as training data and 20% as validation data. The data is first cleaned and pre-processed by standardizing the samples. The pre-processed data is trained on different classification models to classify the samples. There were two approaches to the problem. In the first approach, we employ the given data consisting of seven input features corresponding to the seven WiFi routers on different classification algorithms to classify the data. In the second approach, we first introduce new features generated as nonlinear functions of the given seven features (nonlinear mapping). A total of 452 features are generated with regard to physical meaning of the features like mean and standard deviation of the signal power, minimum and maximum values of signal power and z-scaling. The best 40 features are then selected through feature ranking with recursive feature elimination(RFE). A total of seven classification techniques were employed in both the approaches. The Naïve Bayes Classifier is used as the baseline system in each approach. Logistic Regression, Support Vector Machine(SVM), XGBoost, K-Nearest Neighbors, Random Forest and Multi-Layer Perceptron(MLP) classifier models are implemented in each approach. For each of these models, different hyper parameters are applied using GridSearchCV and the best hyper parameters are selected. The performance of these classifiers are compared by analyzing metrics such as training accuracy, validation accuracy and mean cross validation accuracy. Results were visualized using the confusion matrix for each classifier result. The model with the highest mean cross validation accuracy in each approach was chosen as the final system to classify the test-set samples. In the first approach, the K-Nearest Neighbors classifier gave the best mean cross validation accuracy and achieved a test accuracy of 0.9825. Similarly, in the second approach, the Multi-Layer Perceptron classifier gave the best mean cross validation accuracy and achieved a test accuracy of 0.9575. In conclusion, the first approach produced the best result on the test data with the K-Nearest Neighbors classifier. K-Nearest Neighbors classifier was the best suited model to predict the location of the user. Intuitively we can attribute this to the fact that a router can be considered as a point where the signal range of the router constitutes a circle. As the K-Nearest Neighbors classifier plots the decision boundary in the form of a circle, it is the best predictor for the location of a user.

2. Introduction

In this project we develop a pattern recognition system to efficiently classify a real world dataset. The project report has been structured in the following manner. At first I have defined the problem statement and goals. The next section presents the details of my approach and implementation. Lastly a comparative study has been performed on the classification results obtained by running each classifier on the given real world data set.

2.1 Problem Statement and Goals

The Wireless Indoor Localization data set consists of WiFi signals from an indoor WiFi system that has 7 WiFi routers at various locations. Our goal with this dataset is to predict the location of a user who is in one of four rooms based on the WiFi signal strength received from the 7 routers. The dataset consists of measured signal strength (dB, integers) at 7 wireless sensors (WS1-WS7) and the user location. Thus, there are 7 input variables (features) and 4 classes (user locations).

The primary goal was to find the best classification model with the best hyper parameters which achieved the highest mean cross validation accuracy. This model is then used to classify the test data set. To extend the problem statement, a second approach was undertaken wherein new features were introduced by non- linear mapping of the given input features. The dimensionality of the feature space was reduced by selecting the best 40 features through feature ranking with recursive feature elimination. Here again we find the best classification model with the best hyper parameters which achieved the highest mean cross validation accuracy. This model is then used to classify the test data. We analyze the accuracy results of the final system in both approaches to draw conclusion on which approach produced the best result overall.

3. Approach and Implementation

Feature Engineering and feature dimensionality reduction is first performed and then standardization process is followed. This approach was taken to retain the scale of the input features while generating new features.

3.1. Feature engineering

Feature engineering is not applicable to the first approach. In the first approach, no new features were developed. The dataset consisting of the given seven input features was used directly to train the models.

The second approach consisted of increasing the feature dimensionality space by deriving physical meaning out of the given input features. New features were introduced by as nonlinear functions of the given seven features (nonlinear mapping). The features were generated as follows:

- Mean and standard deviation of the features. The mean and standard deviation of the signal strengths received from the seven routers was calculated for each parameter (each row of the data set). The original data consists of signal strengths in decibels(dB). Here I first converted all the samples from decibel values to linear values by employing the following formula:

  $$\text{Signal Power} = 10 \text{ ^ } (\text{Signal Power(dB)}/20)$$

  Then I converted the values back to decibel scale using the formula:

  $$\text{Signal power(dB)} = 20 * \log_{10}(\text{Signal Power})$$

  This is a nonlinear mapping and the singular value is guaranteed to be non-zero. If we instead use the arithmetic average of the signal strengths in dB, then the singular value after increasing the dimensionality will have a zero.

- Z-score [1] (Standard score) of the input features. Simply put, a z-score gives you an idea of how far from the mean a data point is. But more technically it's a measure of how many standard deviations below or above the population mean a raw score is. The standard score is the number of standard deviations by which the value of a raw score (i.e., an observed value or data point) is above or below the mean value of what is being observed or measured. The z-score was calculated using the following formula [2]:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of the samples and σ is the standard deviation.

- Minimum and Maximum feature in each row of the data set. Here the features with maximum and minimum values in each row corresponding to signal strength is determined. To be clear, the features which correspond to the maximum and minimum value out of the given 7 input features in each row and not their values are determined.
- Calculated the sum (Maximum + Minimum value) using each of the above maximum and minimum features.
- Then built on these features by doing group by maximum value and group by minimum value and then compute the mean.
- Further built on these features by merging the above generated features with the original features.
- Calculated the distance between each data point and the average for every maximum value, minimum value and their sum.

A total of 452 features were generated from the 7 input features.

3.2 Feature dimensionality adjustment

Feature dimensionality adjustment is not applicable to the first approach. In the first approach, no changes to the dimensionality was introduced to the data. The dataset consisting of the given seven input features was used directly to train the models.

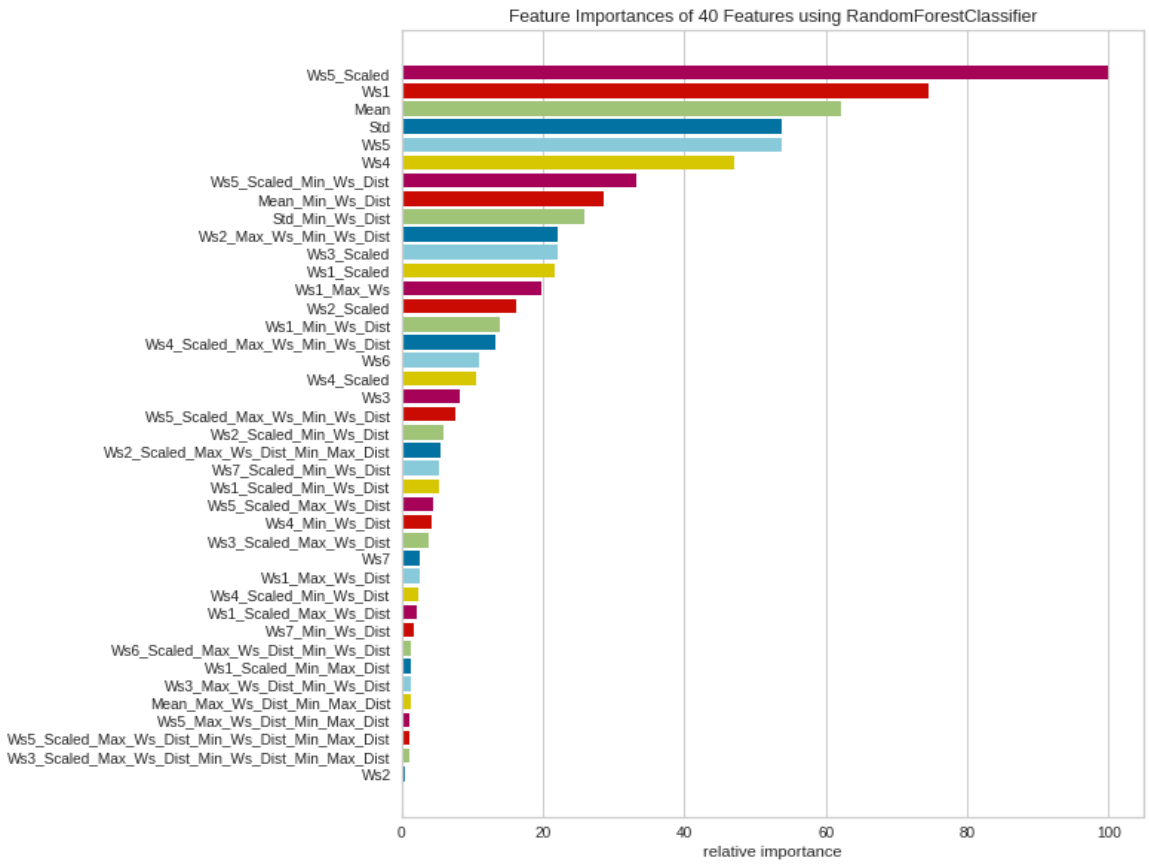
After introducing new features in the second approach, the dimensionality of the feature space was reduced by feature selection. The idea here is to select the most prominent features among the generated features which helps to differentiate the best data points from the others. Here the best 40 features are selected by performing feature ranking with recursive feature elimination (RFE). Recursive feature elimination [3] is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained

on the initial set of features and the importance of each feature is obtained. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached [4].

The estimator model that I chose to implement this was the XGBoost Classifier. I tried various estimators such as SVM and Random Forest models. However, the gradient boosting model gave the best results. The number of best features to be selected was chosen as 40 after trying out multiple values ranging from 25 to 50 wherein selecting the best 40 features produced the best results.

To visualize the relative importance of the features, I developed a visualizer [5] which utilizes the FeatureImportances attribute to rank and plot relative importances of the features. A Random Forest classifier was used as the estimator. A figure showing the features ranked according to the explained variance each feature contributes to the model was created. Here the features are plotted against their relative importance, that is the percent importance of the most important feature.



3.3 Preprocessing

The given dataset is evenly balanced among all the classes consisting of 500 data points for each location (class). Hence there is no issue of unbalanced data and no requirement to perform data balancing for the classes.

In the first approach, we are using the given input features. All features are integer-valued and there are no missing values. Here we do not deal with any NaN values in the data. However, in the second approach after introducing new features by nonlinear mapping of input features, we might deal with NaN values. We fill the NA/NaN values [6] with value 0 before performing feature selection so as to not include these values.

3.3.1 Standardization

Standardization of a dataset is a common requirement for many machine learning estimators. It is a preprocessing technique which is used to scale the input data. Here the features are standardized by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as:

$$z = (x - u) / s$$

where u is the mean of the training samples or and s is the standard deviation of the training samples.

We perform standardization [7] of the data set in both approaches. In the second approach standardization is performed after performing feature engineering and feature selection. This approach was taken to retain the scale of the input features while generating new features.

We perform standardization so that classifier models do not behave badly if the individual features do not look like standard normally distributed data. The models were initially trained on non-standardized data which yielded very poor result in terms of both validation accuracy and mean cross validation accuracy. Hence standardization was done to improve the performance metrics of the models.

3.4 Dataset Usage

The following procedures were followed in the use of the dataset.

3.4.1 Feature Engineering

Feature engineering is not applicable to the first approach. In the second approach the entire training data and test data was used to generate 452 new features from the original seven input features.

3.4.2 Feature Selection

Feature engineering is not applicable to the first approach. In the second approach the best 40 features out of the 452 generated features was selected.

3.4.3 Training and Validation Split

For both the approaches, I have used random sampling to divide the given training dataset into the training and validation data [8]. This is done to split the training data set for two different purposes namely training and validation. The training subset is used for building the model. The validation subset is for using the model on unknown data to evaluate the performance of the model.

The split was carried out by specifying a validation data size of 20% and training data size of 80 %. So the split is 80:20 with respect to training data and validation data.

3.4.4 Standardization

The training data and validation data was scaled by standardization procedure. For the final model, the test data was also standardized. The test data was only used on the final systems selected in both approaches.

3.4.5 Cross Validation

Cross validation on the training data was performed during the grid search with the number of folds specified as 5 (cv=5) uniformly throughout both the approaches.

To obtain the cross validation score [9], the number of folds considered was kept uniformly constant in both approaches with a value of 5(cv=5). The entire training data set was used for cross validation.

3.5 Training and Classification

3.5.1 Training Algorithm

Both the approaches used the same training algorithm for all the classifier models. For training the different classifiers, a grid search algorithm is used. Grid search [10] is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. Here an exhaustive search is carried out over certain specified parameter values with a range of values for each parameter for a classifier model. A cross validation process is performed in order to determine the hyper parameter value set which provides the best accuracy levels. When fitting the model with the training data, all the possible combinations of parameter values are evaluated and the best combination is retained.

The list of parameters for each model was chosen by heuristics. Experimentation with the parameter values were carried out before arriving at the list of values specified which gave rise to better performance. We then use the best set of hyper parameter values chosen in the grid search to train the models. The performance of the selected hyper-parameters and trained model is then measured on a dedicated evaluation set that was not used during the model selection step.

3.5.2 Classifiers

Seven different classifiers were used to fit to the training data and to analyze which model produced the best results. The seven classifiers used are:

- Naïve Bayes Classifier
- Logistic Regression Classifier
- Support Vector Machine(SVM) Classifier
- XGBoost Classifier
- K-Nearest Neighbors(KNN) Classifier
- Random Forest Classifier
- Multi-layer Perceptron(MLP) classifier

1) Naïve Bayes Classifier

The first classifier which I implemented was Naïve Bayes Classifier [11]. I used the Gaussian Naïve Bayes Classifier as the baseline model for classification in both the approaches. This model is based on the Bayes' theorem. In the Gaussian Naïve Bayes algorithm, the likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The model implementation is straightforward and simple. The Naïve Bayes model does not have any hyper-parameters to tune. Hence we do not perform a grid search.

A random classifier was also implemented as a baseline in which the random classifier will randomly pick a class label.

2) Logistic Regression Classifier

The second classifier which I implemented was the Logistic Regression Classifier [12] . The training algorithm used here is the multiclass classification method. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

The model implementation consisted hyperparameter tuning with a grid search in both the approaches. Here only the parameter 'C' which is the inverse of regularization strength is tuned. A large C can lead to an overfit model, while a small C can lead to an underfit model. The list of values for C was chosen by heuristics. The following values were used for tuning:
C = [0.001,0.01, 0.1, 1, 10, 50, 100, 200]
A grid search was carried out to find the best value of C using a cross validation value of 5. The best value of C found is as follows:

Parameter	First Approach	Second Approach
C	50	0.1

Then the model was trained using the best C value found. The penalty parameter used was l_2 and the 'lbfgs' solver. These were chosen as they were found to converge faster. The multinomial class parameter was selected as it implements a a true multinomial logistic regression model rather than the default one-vs-rest method.

3) Support Vector Machine(SVM) Classifier

The third classifier which I implemented was Support Vector Machine(SVM) classifier [13]. SVM chooses the decision boundary that maximizes the distance from the nearest data points of all the classes. It finds the most optimal decision boundary. Since we are dealing with non-linearly separable data, I used the Gaussian kernel (rbf kernel) for the SVM Classifier.

The model implementation consisted hyperparameter tuning with a grid search in both the approaches. Here the parameters C and gamma were tuned. The parameters list specified were as follows:
C = [0.01, 0.1, 1, 10, 50, 100, 200] and
gamma = [0.001, 0.01, 0.1, 1, 10]. The list of values was chosen by heuristics. Grid search was implemented with a cross validation value of 5. The best parameter values found is as follows:

Parameter	First Approach	Second Approach
C	50	50
Gamma	0.1	0.001

Then the model was trained using the best values found for C and gamma.

4) XGBoost Classifier

The fourth classifier which I implemented was the XGBoost Classifier [14]. The XGBoost stands for Extreme Gradient Boosting and it is a boosting algorithm based on Gradient Boosting Machines. XGboost applies regularization technique to reduce overfitting, and it is one of the differences from the gradient boosting. Another advantage of XGBoost over classical gradient boosting is that it is fast in execution speed.

The model implementation consisted hyperparameter tuning with a grid search in both the approaches. Here the parameters learning rate, number of trees and maximum depth of the tree were tuned.
The parameters list specified were as follows:
Learning rates = [0.01,0.05,0.1,0.15,0.2]
Number of trees = [10, 50, 100, 200]
Maximum depth = [1,2,3, 4.....,20]
The list of values was chosen by heuristics. Grid search was implemented with a fivefold cross validation.
The best parameter values found is as follows:

Parameter	First Approach	Second Approach
Learning rate	0.2	0.15
Number of trees	200	100
Maximum depth	1	1

Then the model was trained using the best values found for each.

5) K-Nearest Neighbors Classifier

The fifth classifier which I implemented was K-Nearest Neighbors Classifier [15]. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the

new point, and predict the label from these. A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

The model implementation consisted hyperparameter tuning with a grid search in both the approaches. Here the parameters number of neighbors and weight function were tuned. The parameters list specified were as follows:
k = [1,2,3,4,5,6,7,8,9,10]
weights = ['uniform','distance']
The list of values was chosen by heuristics. Grid search was implemented with a fivefold cross validation. The best parameter values found is as follows:

Parameter	First Approach	Second Approach
k	4	3
Weight function	'distance'	'Uniform'

Then the model was trained using the best values found for each.

6) Random Forest Classifier

The sixth classifier which I implemented was the Random Forest Classifier [16]. Random forest is an ensemble method which create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.

The model implementation consisted hyperparameter tuning with a grid search in both the approaches. Here the parameters number of trees in the forest and maximum depth of the tree were tuned. The parameters list specified were as follows:
Maximum depth = [1,2,3, 4.....,20]
number of trees = [10, 50, 100, 200]
The list of values was chosen by heuristics. Grid search was implemented with a fivefold cross validation.

The best parameter values found is as follows:

Parameter	First Approach	Second Approach
Number of trees	50	200
Maximum depth	10	6

Then the model was trained using the best values found for each.

7) Multi-Layer Perceptron(MLP) Classifier

Multi-layer Perceptron [17] is a class of feedforward artificial neural network(ANN). An MLP [18] consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP uses backpropagation for training.

The model implementation consisted hyperparameter tuning with a grid search in both the approaches. Here the parameters alpha (L2 penalty parameter) and maximum number of iterations were tuned. The parameters list specified were as follows:
alpha = [10.0 ^ -(1), 10.0 ^ -(2) ……., 10.0 ^ -(10)]
maximum number of iterations = [800,1000,1500]
The list of values was chosen by heuristics. Grid search was implemented with a fivefold cross validation. The best parameter values found is as follows:

Parameter	First Approach	Second Approach
alpha	0.01	0.001
Maximum iterations	800	800

Then the model was trained using the best values found for each.

4. Results

4.1 Performance Metrics

Training accuracy and Validation accuracy metrics are computed on the model using the best hyperparameter values. The mean cross validation accuracy is computed for each of the model using the entire training data set along with the standard deviation. A Confusion Matrix is computed to evaluate the accuracy of a classification using the prediction on the validation data. The confusion matrix is visualized.

The results obtained from both the approaches are summarized as follows:

4.1.1. Approach 1

The Accuracy of random classifier was found to be 0.26875.

The performance comparison of the different models is tabulated and presented below.

Classifier	Training Accuracy	Validation Accuracy	Mean cross-validation accuracy	Standard deviation
Naïve Bayes(Baseline)	0.983593	0.990625	0.984375	0.005229
Logistic Regression	0.983593	0.990625	0.97875	0.007234
Support Vector Machine(RBF Kernel)	0.994531	0.975	0.980625	0.007756
XGBoost	0.996093	0.990625	0.984375	0.003423
K-Nearest Neighbours	1.0	0.9875	0.986875	0.0075
Random Forest	0.998437	0.990625	0.9825	0.005077
Multi-layer Perceptron	0.990625	0.99375	0.984375	0.006846

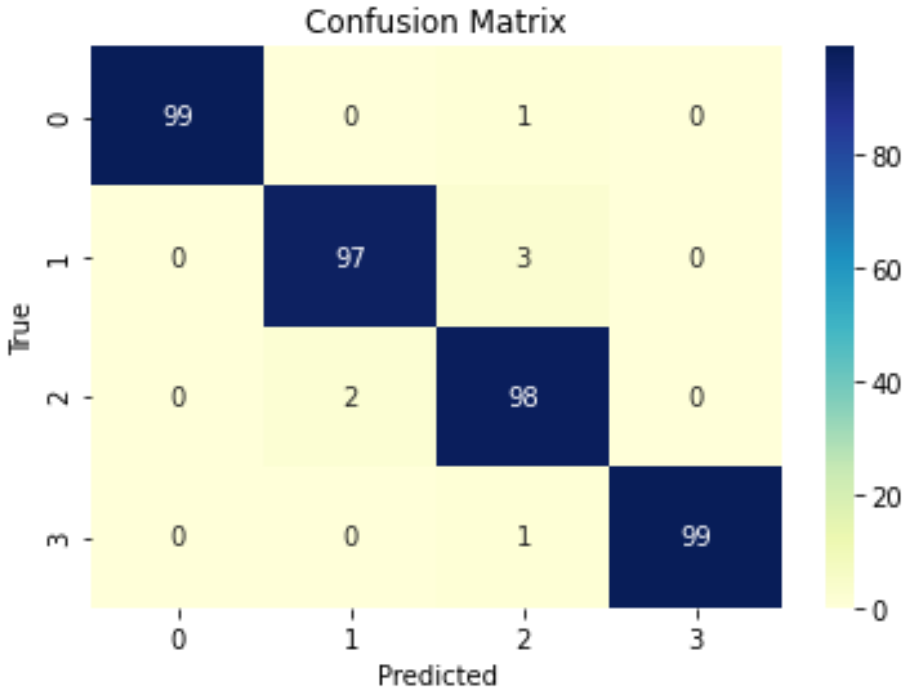
From the results shown above we see that each of the models achieved near 100% training and validation accuracies. Clearly standardization of training data

has produced better results than not standardizing the data. The results are summarized as follows:

1) The K-Nearest Neighbors Classifier achieved the highest training accuracy of 1.0.
2) The MLP Classifier achieved the highest validation accuracy of 0.99375.
3) The K-Nearest Neighbors Classifier achieved the highest mean cross validation accuracy of 0.985875.

Our metric to choose the best model as the final system to classify the test-set samples is to choose the model with the highest mean cross validation accuracy. This reason to choose this metric is because although accuracy scores for classification are useful to determine the optimal model, real world data is often distributed unevenly, meaning that the fitted model is likely to perform better on some sections of the data than on others. Hence analyzing the mean cross validation scores is a better metric.

We choose the K-Nearest Neighbors model as the final system to classify test data as it achieved the highest mean cross validation accuracy. The test accuracy for our chosen K-Nearest Neighbors model is 0.9825. The confusion matrix for this is shown below.



4.1.2 Approach 2

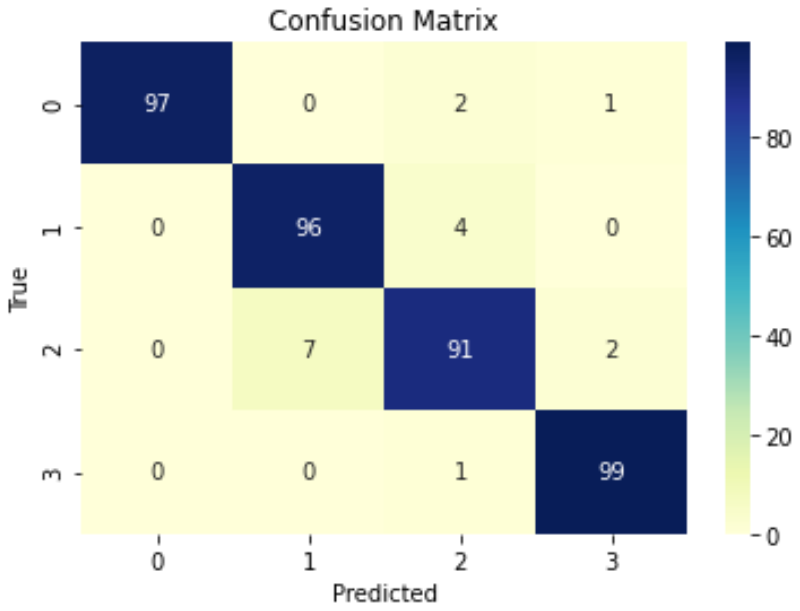
The performance comparison of the different models is tabulated and presented below.

Classifier	Training Accuracy	Validation Accuracy	Mean cross-validation accuracy	Standard deviation
Naïve Bayes(Baseline)	0.967968	0.975	0.97	0.005448
Logistic Regression	0.984375	0.990625	0.98375	0.006373
Support Vector Machine(RBF Kernel)	0.985937	0.990625	0.98437	0.007126
XGBoost	0.992187	0.990625	0.98437	0.005929
K-Nearest Neighbours	0.990625	0.978125	0.975	0.010825
Random Forest	0.991406	0.98125	0.9825	0.003186
Multi-layer Perceptron	1.0	0.996875	0.985	0.003644

The results are summarized as follows:

1) The MLP Classifier achieved the highest training accuracy of 1.0
2) The MLP Classifier achieved the highest validation accuracy of 0.996875
3) The MLP classifier achieved the highest mean cross validation accuracy of 0. 985.

Our metric to choose the best model as the final system to classify the test-set samples is to choose the model with the highest mean cross validation accuracy. We choose the Multi-Layer Perceptron(MLP) model as the final system to classify test data as it achieved the highest mean cross validation accuracy. The test accuracy for our chosen KNN model is 0.9575. The confusion matrix for this is shown below.



5. Conclusion

The first approach produced the best result on the test data with the K-Nearest Neighbors classifier. The second approach produced a poor result with the MLP classifier. This result was surprising as the performance metrics like training accuracy, validation accuracy and mean cross validation accuracy were very high.

K-Nearest Neighbors classifier was the best suited model to predict the location of the user. Intuitively we can attribute this to the fact that a router can be considered as a point where the signal range of the router constitutes a circle. As the K-Nearest Neighbors classifier plots the decision boundary in the form of a circle, it is the best predictor for the location of a user.

References

[1] https://www.statisticshowto.com/probability-and-statistics/z-score/

[2] https://en.wikipedia.org/wiki/Standard_score

[3] https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[5] https://www.scikit-yb.org/en/latest/api/model_selection/importances.html

[6]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html

[7]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[8]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[9]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

[10]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[11] https://scikit-learn.org/stable/modules/naive_bayes.html

[12]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[13] https://scikit-learn.org/stable/modules/svm.html

[14] https://xgboost.readthedocs.io/en/latest/python/python_api.html

[15]https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[16] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[17] https://en.wikipedia.org/wiki/Multilayer_perceptron

[18]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html