**IOT Analytics Wi-Fi Locationing**

Jelly P. Edwards

**Intro:**

IOT Analytics was contacted by a new client that wanted us to help analyze their new Wi-Fi Locationing system. This client is developing a new program that would track a person’s location via “Wi-Fi fingerprinting” in an interior space. Wi-Fi fingerprinting uses the signals from the Wi-Fi hotspots to determine a person’s location, which is similar to how GPS uses satellite signals for location. This client provided a large database of their Wi-Fi fingerprints for a three-building university campus. We were asked to evaluate this data set and to provide a recommendation to the client based on machine learning models.

**Data**

The client provided a dataset that had roughly 19,938 observations with 529 features. Of the 529 features, 520 contained the wireless access points (WAP). The other 9 features were user location and user information, as well as time stamp. I merged the location features into a single unique identifier called USERLOCATION. I set this feature to be a factor data type since it is categorical data. Due to USERLOCATION being a concatenation of the FLOOR, BUILDINGID, SPACEID, and RELATIVEPOSTION features, I removed these from the sample data sets due to redundancy. This would also make computation time improve. I also removed LONGITUDE and LATITUDE since the purpose of this task was to predict via WAP data points, and the LONGITUDE and LATITUDE affected the model’s ability to predict and did not produce results that were desired. The machine learning models would score extremely high with those two features, skewing the ability to test the capabilities of the WAP features.

This USERLOCATION feature is the dependent variable that we were asked to predict based off the WAP data. Due to the large size of the data set, I sampled the data by BUILDINGID to ease the computation needed to process the data. There were three different building in totals, with building 0 at 5249 observations, building 1 at 5196 observations and building 2 at 9492 observations. Previously, I tried created a randomized sample of 5000 observations from the original data set, but this caused issues later down the line with the machine learning models due to small data points for the categorical dependent variable. There were no missing data points or nulls in this data set.

**Cross-Validation with Machine Learning Models**

I decided to perform some cross-validation on three different algorithms to explore which would be best suited for the sample data. The three algorithms I used were RandomForest (RF), C5.0, and K-nearest neighbor (KNN). These are three prominent algorithms for classification machine learning, and since the dependent variable was a factor data type, I thought it best to focus on using algorithms that would work best with identifying discrete output variables. When it comes to choosing algorithms, I personally like RandomForest as I always find it robust and more accurate than other algorithms. However, that robustness also means that it has a long computation time – and since the data set was large, I thought it best to look for other algorithms that would provide good predictions but not take as long. C5.0 is similar to RandomForest, as it also is a decision tree classifier. However, it isn’t as robust as RandomForest, but it proved to take a shorter computation time. KNN is an algorithm that assumes that similar points exist near each other and uses feature similarity for predicting. It is a more simplistic machine learning algorithm that is good for classification purposes, and it uses less computation time.

I split the sample data sets by 75%/25%, with 75% for Training the models and 25% for Testing. I ran the three algorithms on all Training data sets with 10-fold cross validation, and the results of these can be seen in *Figure 1*. This figure provides both the Accuracy and Kappa scores for each model. Accuracy is metric that shows the percentage of correctly classified cases out of all possible cases. Kappa is a metric that compares the observed accuracy that were classified by the model vs the expected accuracy. For our case, Kappa is the most important metric as it considers random chance and is useful for a dataset that is unbalanced. When comparing the Accuracy and Kappa scores, it was clear that RF proved to be the best algorithm. It scored the highest in both Accuracy and Kappa for all data sets. KNN showed the least promise, as it scored lower than the other two algorithms.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Building | RF  Accuracy | RF  Kappa | C5.0  Accuracy | C5.0  Kappa | KNN  Accuracy | KNN  Kappa |
| 0 | 0.85 | 0.85 | 0.79 | 0.79 | 0.65 | 0.65 |
| 1 | 0.95 | 0.95 | 0.88 | 0.88 | 0.80 | 0.80 |
| 2 | 0.88 | 0.88 | 0.80 | 0.80 | 0.76 | 0.76 |

***Figure 1.*** *10-Fold Cross Validation Algorithm Tests for the 3 Buildings (Training Data)*

**Evaluation of Prediction Models with Testing Data**

After running the cross-validation tests, I used the models on the Testing data sets to evaluate their performance. To evaluate the performance of each model, I used caret’s postResample function. This function evaluates the overall agreement rate and Kappa across a provided prediction matrix. The values for the three models can be seen in *Figure 2.* From this figure, it is clear that once again, RF has the highest accuracy and kappa values out of all three models. However, C5.0 is very close in Accuracy and Kappa as well.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Building | RF  Accuracy | RF  Kappa | C5.0  Accuracy | C5.0  Kappa | KNN  Accuracy | KNN  Kappa |
| 0 | 0.97 | 0.97 | 0.96 | 0.96 | 0.74 | 0.74 |
| 1 | 0.99 | 0.99 | 0.96 | 0.96 | 0.87 | 0.87 |
| 2 | 0.98 | 0.98 | 0.93 | 0.93 | 0.85 | 0.85 |

***Figure 2.*** *Evaluating Models after Prediction (Testing Data)*

To further evaluate these prediction models, I used the resamples() function in caret. This function compares the models by their resampling distributions and provides a detailed list of their performance. I have provided pictures of the resample metrics for each sample data set below in *Figure 3, Figure 4,* and *Figure 5.* From these metrics, we can see RF consistently has high Accuracy and Kappa throughout each resample for all three building data sets.

*Text

Description automatically generated*

***Figure 3.*** *Respective Metrics for Each Model for Building 0 Data Set*

*Text

Description automatically generated*

***Figure 4.*** *Respective Metrics for Each Model for Building 1 Data Set*

*Text

Description automatically generated with medium confidence*

***Figure 5.*** *Respective Metrics for Each Model for Building 2 Data Set*

**Algorithm Recommendation for Predictions**

Originally, based on postResample function, I was going to recommend the C5.0 algorithm. It was clear from that evaluation that RF had the highest accuracy and kappa rate, but C5.0 was only slightly behind. C5.0 scored above 0.90 for both Accuracy and Kappa, showing that it provides good predictions. While RF was the highest, it had one downside: computation time. This algorithm took a very long time to run, and I had to adjust core usage with RStudio to speed this processing time. Even still, it took around an hour or so to run the query. C5.0, however, had a short processing time of around 10 minutes or so. This made me think that it was more promising between the two.

However, I am glad I also evaluated these models via the resamples() function. This gave further insight to the performance of each model, making it more visible to me that RF is really the best choice. This model had consistent high Accuracy and Kappa values throughout each resample, while C5.0 does not. For Building 0 data set, the kappa score for RF was on average 0.85, while the average for C5.0 was 0.80. For Building 1, the kappa score for RF was on average 0.95, while C5.0 was 0.88. For Building 2, the average kappa score for RF was 0.88, while C5.0 was 0.80. This leads me to believe that RF is the best algorithm to set this data set, as long as the user has the time to run the entire query.

**Other Recommendations**

There are currently a lot of resources and research for indoor locationing. There are many companies that are investigating this topic, and I took some time to research further into it. The dataset provided her was based on Wi-Fi Fingerprinting, but this isn’t the only technique that can be used for a Wi-Fi positioning system. Another system is simply a signal strength-based technique called RSSI localization. Technically, Fingerprinting is a form of RSSI localization as it is based on the signal strength between a client device and several Wi-Fi Access points. However, Fingerprinting requires the strength to be recorded in a database, which is used during an online tracking phase where it compares the active location to that of the recorded one. Users can try using a more active Wi-Fi tracking technique and use trilateration to calculate the estimated client device position to the known access points. However, it is said that the more active RSSI localization is less accurate than Fingerprinting. A positive of this technique is that it does not get affected by environmental changes like Fingerprinting would.

There are also other forms of indoor positioning that do not involve Wi-Fi. There are currently several companies that are performing Bluetooth indoor positioning. This may be another option that the client can try for their indoor positioning system. One negative about this technique, however, is that Bluetooth is short-range and doesn’t provide as wide of range as the Wi-Fi positioning does. If there is a possibility to use both to cover all bases, that may be best.