Eva Lipska

C3-T3

**Evaluate Techniques for Wifi Locationing**

- Internal Report -

**Goal of the project**

Evaluation of the application of machine learning techniques to the problem of indoor locationing via WiFi fingerprinting.

**Machine learning process**

Learning task driving the choice of algorithms: classification

Comparison of the models produced by following algorithms: k-Nearest Neighbors, Naïve Bayes, C5.0, Random Forest.

1. **Data collection**

Database of WAP signals for a multi-building industrial campus with a location (building, floor, and location ID) associated with each WAP attribute.

Data description: 19937 observations, 529 attributes.

1. **Data exploration and preparation**

Data quality: data was clean (no messy data, no missing values)

Sample: building # 0 (5249 observations)

Data was recoded to conform to the classification problem. New, dependent variable LOCATION (class: factor) was created, consisting of FLOOR and SPACEID attributes.

Unnecessary data was eliminated: e.g. features with zero-variance, variables causing overfitting (previously included in LOCATION attribute).

New dataset included only the dependent variable LOCATION and wifi fingerprints.

1. **Model training**

Selection of the appropriate algorithm for classification problems: k-Nearest Neighbors, Naïve Bayes, C5.0 and Random Forest.

Data slicing:

Training set: 3996 observations (75%)

Testing set: 1253 observations (25%)

200 predictors (only WiFi fingerprints)

Dependent variable LOCATION including 256 classes

10 fold cross validation with 3 repeats

***k-Nearest Neighbors*:** well-suited for classification tasks; works well when the relations between features and target classes are numerous and complicated. Classifies unlabeled examples by assigning them the class of similar neighbors using distance.

Preprocessing: usually features are transformed to a standard range prior to applying the KNN algorithm, as distance formula is highly dependent on how features are measured. It prevents domination by the features with much larger rage of values. There is no need for preprocessing because all of the predictor attributes have the same range of possible values.

**Naïve Bayes algorithm:** estimates likelihood of a potential outcome; information from numerous attributes is considered simultaneously in order to estimate the overall probability of an outcome. While many algorithms ignore features that have weak effect, Bayesian methods utilize all the available evidence to subtly change the predictions.

**C.05 decision tree algorithm:** all-purpose classifier, that handles numeric or nominal features as well as missing data; excludes unimportant features; reasonable defaults; post-prune tree strategy; human readable format of the structure; easy to over- and underfit (which may be addressed through adjusting some parameters).

1. **Model evaluation**

At the beginning Accuracy and Kappa Scores (comparing an Observed Accuracy with an Expected Accuracy) were evaluated based on testing data, using the default model parameters. Both metrics were used to select the optimal model using the largest accuracy and kappa values.

**k-Nearest Neighbors:** k = 5; Accuracy = 0.55; Kappa = 0.55

Low accuracy: the algorithm was struggling to identify the class boundaries. Explanation: lack of clear distinctions among classes, similar class type is not fairly homogenous.

**Naïve Bayes:** Accuracy = 0.42; Kappa = 0.42

Even though the algorithm has a lot of strengths (effectiveness, working well with very large numbers and examples), in this case it didn’t perform well.

Possible explanation: Naïve Bayes uses frequency tables to learn the data and each feature must be categorical (wifi data is numeric) in order to create the combinations of class and feature values comprising of the matrix.

**C5.0:** Accuracy = 0.71; Kappa = 0.70

After seeing that the C5.0 algorithm obtained the highest accuracy of the models tested, an additional tree-based model, Random Forest was selected for further testing.

Random Forest reached the highest accuracy.

**Random Forest**: Accuracy = 0.76; Kappa = 0.76

1. **Model improvement**

After initial evaluation and the choice of the best performing algorithm, to augment the performance of the selected model, additional parameters should be adjusted (tuning). For example, through adding the parameter *tuneLength*.

For Random Forest, the search may be executed with the use of three different searching strategies for the *mtry* parameter: default, random, and grid, which may improve the performance.

Different sampling methods should be also considered, as well as removing more attributes.

1. **Final evaluation**

Selection of the Random Forest algorithm was based on its accuracy level, however alternative performance measures derived from the confusion matrix were developed and analyzed.

On average, additional measurements presented pretty good performance, indicating that the model should be a subject to a further improvement.

Meaningful sensitivity (true positive rate): 77 %

Very high specificity (true negative rate): 100 %

Good precision (positive predictive value): 79 %

Recall: 77 %

F - measure: 77 %

**Conclusions**

Further improvements to the selected model (Random Forest) are necessary as at this stage it doesn’t appear to be performing well and can’t be deployed for its intended task.

Improving model’s performance may include:

- removing additional attributes,

- tuning algorithm parameters,

- considering a different sampling method.