Segmenting and classifying GPS traces to identify types of transportation

Machine Learning – WS 2011/2012

Sebastian Nagel, BSc – 03630886 – SEBASTIAN.NAGEL@TUM.DE –TECHNICAL UNIVERSITY OF MUNICH

Abstract

A lot of research on traffic behavior of individuals and statistics on means of transportation benefits from accurate and near-term GPS data. The MobiTracker project researches techniques on how to identify types of transportation of a user's GPS trace automatically by applying machine learning algorithms This paper discusses filtering algorithms and segmentation techniques, including Dynamic Bayesian Networks, especially Semi-Markov models. As for classification, advanced feature extraction algorithms using OpenStreetMaps data and an informed approach to training a LibSVM Support Vector Machine in contrast to the traditional approach of instrumenting third party Data Mining Engines are used. By using the new approach a better classification result is achieved with an overall accuracy of 89.66% compared to 86.72%. The proposed approach improves the identification results of nearly all types of transportation and the suggested advanced segmentation algorithm promises an even more flexible recognition of track segments.

1. Introduction

The MobiTracker project researches the possibility to identify types of transportation of a user's GPS trace automatically by applying different machine learning algorithms. If GPS track data can be processed in a convenient way, this procedure could help different institutions or governments to improve accuracy and actuality of their research on transportation behavior of individuals. The discussed types of transportation currently contain classes BIKE, BOAT, BUS, CAR, COACH, TRAIN, TRAM and WALK.

The general procedure to identify transportation types can be broken down into two phases: Segmentation and Classification. First in the segmentation (or recognition) phase, the raw GPS traces are preprocessed to filter out invalid measurements or data with no information (e.g. due to no movement – "point clouds"). Then a segmentation algorithm is applied to divide the continuous GPS track into segments. In phase two these segments shall be classified according to a set of features, which have to be extracted first. Figure 1 illustrates the whole process and shows all relevant steps during training (violet) and prediction (blue) where edited data means labeled data.

In the following chapters both phases and their substeps are discussed. Especially capabilities of Dynamic Bayesian Networks (DBNs) for segmentation and the newly implemented (informed) approach of a Support Vector Machine for classification are explained.

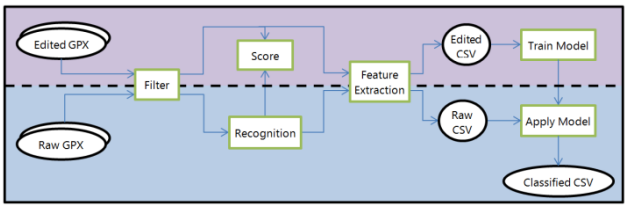

Figure 1 Overview of the process

1.1 GPS Traces / Data structures

GPS traces, or also called tracks, usually contain a big number of track points (e.g. one for every second) and can be broken down into segments which typically group points semantically.

The gathered GPS data has been collected from a variety of different tracking devices and data standardization concerns were made. Luckily nearly all GPS trackers conform to a standard introduced by the National Marine Electronics Association (NMEA) originally intended only for maritime navigation. The NMEA 0183 format consists of different record types, containing information on position, time, magnetic deflection, satellite information etc. Unfortunately not all of the used devices support all aforementioned records.

The GPS Exchange Format (GPX) is optimized for storing and exchanging GPS information between applications or web-services. GPX is a XML based format and therefore very convenient to read by humans

or XML-Parsers. Because of the wide support for GPX in trackers and editing software it is also used in this project.

Track segments are stored as comma separated values along the process, both as intermediate and finally classified results. LibSVM uses a special storage format which is documented in the readme files available with the software under `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

2. Segmentation / Recognition

GPS Tracks have to be segmented in order to create semantically connected groups of track points which resemble parts with a single type of transportation. These segments can then be labeled by the user, which result in the training dataset for classification.

This chapter copes with techniques of finding the correct start and end points of such segments. To score different segmentation algorithms, labeled tracks can be matched with automatically segmented data resulting in a comparable accuracy value similar to the standard machine learning procedure.

Note: The terms segmentation and recognition are used synonymously throughout the paper.

2.1 Filtering

Before we apply segmentation algorithms on the GPS data, several filtering steps are undertaken to improve the data quality and reduce the computational effort in the following steps. Three filters have been developed to eliminate data with low entropy, for example during phases of lost GPS signal and/or longer stays at the same place (e.g. in houses). These "clouds" of track point data can be filtered out efficiently with such filters as seen in Figure 2 and Figure 3.

In this project a filter for course changes, minimal distance between track points and a filter for correcting invalid data due to lost signals have been implemented.

2.2 Traditional method

The most obvious way to recognize track segments is by having an algorithm find certain "stop points" where the tracker didn't move for a certain time and therefore a set minimal distance is not covered during a defined time frame. The traditional method is based on this idea and extends in observing "future" and "past" points around candidate track points and calculating average velocities over these sets. The change from "past" to "future" average velocity gets calculated and if it exceeds a set threshold, a new segment is found. Algorithm 1 tries to give an outline on the procedure.

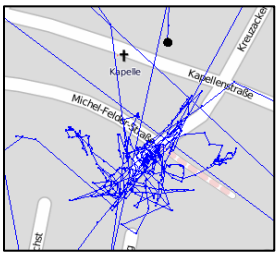

Figure 2 Point cloud

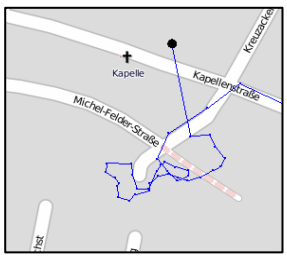

Figure 3 Filtered point cloud

Algorithm 1 AdvancedRecognition

```
params: p, f, futureSecs,
        minChangePercent, minMovement
input: points
 segment := new segment
 for each point in points
   avgPast := avg velocity of p past points
   avgFuture := avg velocity of f future points
   maxDistanceFuture := maximal covered distance in
                        the next futureSecs seconds

   change := abs(avgPast - avgFuture)
   higher := max(avgPast, avgFuture)

   if (change > higher * minChangePercent
        or maxDistanceFuture < minMovement)

     if (distanceCovered(segment) >= minMovement)
       accept segment
     else
       discard segment
     end if

     segment := new segment
   end if
   add point to segment
 end for
return all accepted segments
```

2.3 Dynamic Bayesian Networks

The problem with the traditional approach is that it's parameters have to be configured by hand and can only be evaluated in trial and error. When a good score is achieved on the training data, it is unclear how well the set thresholds will perform on unknown data. As this deterministic approach is somewhat inflexible and is one of the key points to improve, other solutions have to be found.

A very promising approach is the use of Dynamic Bayesian Networks (DBNs). They have been introduced in (Murphy, 2002) and generalize Hidden Markov Models (HMM) and Kalman Filters, all modelling state sequences by reasoning on emitted observations of these states. The markovian assumption, on which all these models base, says that state transitions only base on the previous state and every state emits some kind of observable data.

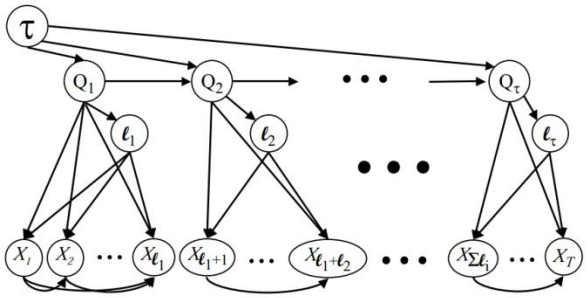

Figure 4 State sequence SMM

In terms of GPS track segmentation one particular subset of DBNs are of interest: Semi-Markov Models also called segment models. These models differ from typical HMMs or Kalman Filters by allowing for multiple observations emitted by the same state. Figure 4 shows a resulting state sequence of such a Semi-Markov Model where $\mathcal{Q}$ are the states, emitting segments of observations $\mathcal{X}$ with varying length ℓ. Based on this approach different research had been done by (Duong et. al., 2005), (Liao et. al., 2006) and (Shi et. al., 2008).

Modeling: One thinkable modeling approach to SMMs could be to model calculated velocities from the GPS position and time measurements as observations $\mathcal{X}$, while using the different types of transportation as underlying states $\mathcal{Q}$ similar to the lowest layer in the hierarchical model presented in (Liao et. al., 2006).

Training: The overall likelihood of the Semi-Markov Model will be given by

$$P(y_{1:T}) = \sum_{\tau} \sum_{q_{1:\tau}} \sum_{l_{1:\tau}} \prod_{i=1}^{l_i} P(\tau)P(q_i|q_{i-1},\tau)P(l_i|q_i)P(y_{s_i:e_i}|q_i,l_i)$$

where τ is the number of segments, l_i the segment length, and s_i and e_i are the start end ending times of segment i.

A particular segment is then described by

$$P(y_{1:l}|Q_t = q, l) = \prod_{t=1}^{l} P(y_t|Q_t = q)$$

when $s_i = 1$ and $e_i = l$ with iid. observations.

As SMMs are a subset of DBNs, existing learning and inference algorithms can be applied to learn the parameters (Murphy, 2002). In the end we are interested in computing the probability of a state at time t given the observations $y_{1:T}$: $P(Q_t = q|y_{1:T})$

Even though the DBN approach has not been implemented in this project, it promises a flexible modeling technique for finding segment borders if one looks at results of aforementioned research work.

3. Classification

When the GPS track is segmented, classification techniques can be applied to predict the type of transportation for each segment. In order to train a classifier for this task, features describing the segment have to be extracted first. With the featured and labeled data, different types classifiers can be trained and used for classification of new data.

This chapter elaborates on feature extraction techniques, as well as the (traditional) approach taken so far and a better, more informed way how to train classifiers. Both approaches use cross validation for scoring the performance of the model.

3.1 Feature extraction

In the MobiTracker project three types or levels of features were identified:

- **Level 1** features are values which can be directly calculated from the track point data of a segment, such as average and maximal velocity, covered distance or maximal acceleration.
- **Level 2** features are calculated from GIS data (e.g. Google Maps or OpenStreetMaps). Such properties include percentage on roads in general, on railways, on motorways and so on, as well as number of bus stops, tram stops etc.
- **Level 3** features would be information from public transportation services like bus schedules. These types of features are not yet implemented and only ideas for future development.

Currently Level 1 and 2 features are implemented and used for classification. Where the GIS features are extracted by a color coded algorithm working on OpenStreetMaps tile data. This extraction is not flawless, but the easiest to implement. Possibly improved feature quality could be achieved with a geocoded feature extraction algorithm.

3.2 Traditional method

Up to now classification models had been trained by including RapidMiner (Mierswa et. al., 2006) as a third party application into the project application. Experiment execution layouts of this Data Mining Engine for K-Nearest Neighbors, Naïve Bayes, Decision Tree and Neural Net models had been preconfigured for later use. Then during training and classification the appropriate experiment file was executed.

As only default or automatic parameters of the model were used, this approach was sort of a “BlackBox” way of implementing a classification algorithm. A good result could only be achieved by trying blindly different sets of features and values of the limited configuration options. Luckily the RapidMiner environment does a very good job in finding hyperparameters automatically.

3.3 Support Vector Machines

Besides the different nature of Support Vector Machines (SVM) compared to the already implemented classifiers, this strategy resembles a more informed way of approaching a classification problem. There exist good introductory guides (Burges, 1998) and a variety of off-the-shelf libraries for implementing like LibSVM (Chung, 2011) which was used in this project.

A support vector machine is a kernel based method which tries to maximize the margin between two classes. For linearly separable data this is straight-forward. Whereas, in the case of non-linearly separable data, slack variables ξ_i are introduced to allow for misclassification (weak margin). The objective function to optimize is then given by:

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_i \xi_i$$

where C balances a harder / softer margin and is therefore the regularization parameter.

As not all classification problems are approximated best with linearly, kernels are used. They emerge from an interesting observation made during optimization: the feature vector $\boldsymbol{x}$ never has to be evaluated directly, but only the dot product as in the dual optimization problem:

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j}$$

This allows the application of kernels, which describe the similarity of two feature vectors $\boldsymbol{x}$ by calculating a real value for the product:

$$K(\boldsymbol{a}, \boldsymbol{b}) = \phi(\boldsymbol{a}) \cdot \phi(\boldsymbol{b})$$

where $\phi(\boldsymbol{x})$ is a mapping function. This function maps the feature vector $\boldsymbol{x}$ from the input space into a higher dimensional feature space, where the data is more likely to be linearly separable. The most frequently used (and in LibSVM implemented) kernels are the linear, polynomial, Radial Basis Function and sigmoidal kernel.

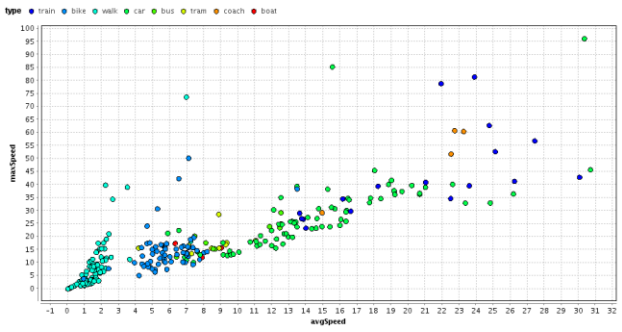

Figure 6 Plot of avgSpeed and maxSpeed features

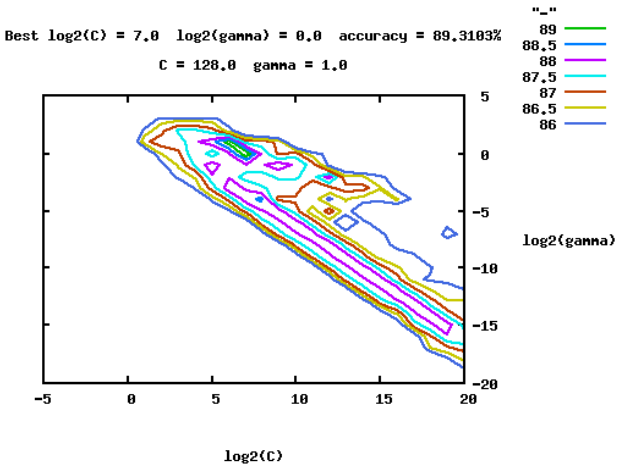

Figure 7 Coarse grid search

To implement multiclass classification, most commonly the one-versus-rest method is used with SVMs where always one class has to be separated from the remaining dataset.

Training: To start off with a good approach in classification with SVMs (and in general) is to have a look at the data. By knowing the significance of some features, possible outliers and missing values, one can improve his strategy significantly. Figure 6 shows a 2D plot of the two most significant features of this dataset. It can easily be seen that the WALK (cyan) class can be separated quite easily, whereas TRAIN (dark blue) and CAR (green) overlap.

As also stated in (Hsu, 2003), feature scaling increases training quality significantly. This is currently done by simply analyzing the range of each feature and normalizing it to the range $[0, 1]$. It is important to apply the same feature scales on training and prediction data to ensure correct scaling of the feature values.

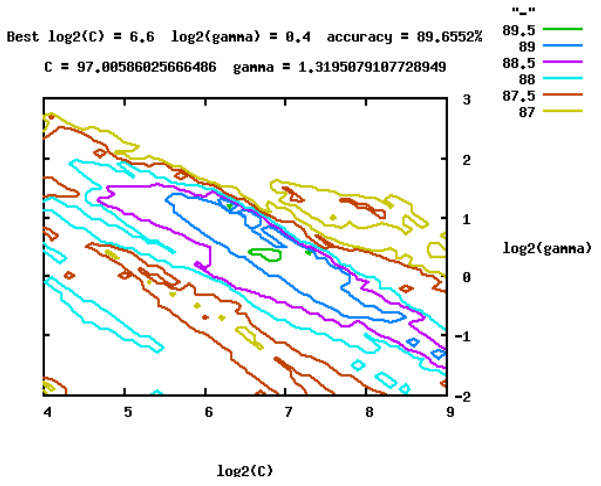

Figure 5 Fine grid search

In terms of kernel selection, a RBF kernel has been chosen because it only needs two parameters (C, γ) and still is very flexible. In fact it could map the input space to an infinitely high features space. A good starting point in parameter selection is a neutral $C = 1$ and a $\gamma = 1/m$ with m being the number of features. In case of the RBF kernel, even grid search is a feasible way to find good hyperparameters. For this dataset two grid searches had been conducted, one with a coarse range (Figure 7) and one with a fine grid (Figure 5). The finally selected parameters are $C = 97$ and $\gamma = 1.395$.

4. Results

In both approaches the models were trained on 294 labeled track segments (290 after removing examples with missing values). By using 5-fold cross validation, the training could be validated. As RapidMiner uses slightly different features scaling and due to shuffling in cross validation the values are not 100% comparable. But a comparison $\pm 1\%$ is realistic as this is roughly the standard deviation one gets in multiple runs of cross validation with the same parameters.

The presented results are only from training with level 1 and 2 features because training only on level 1 features usually leads to a lower performance of the model.

Table 1 Accuracy and Precision of models

Model	Accuracy	Precision Car	Precision Bike
Naïve Bayes	47.60%	71.43%	40.86%
K-NN	59.82%	56.47%	40.00%
Decision Tree	86.09%	81.61%	88.71%
Neural Net	86.72%	83.53%	86.15%
SVM	89.66%	85.71%	89.23%

Table 1 states the achieved accuracy of the four models of the traditional method (Naïve Bayes, K-NN, Decision Tree, and Neural Net) along with the result of the newly implemented SVM. Even though the training data did not contain many examples on classes like BOAT and COACH (5 and 4 respectively), the Support Vector Machine performs with a respectable overall accuracy of 89.66%. The second and third column contain precision values of the classes CAR and BIKE which had next to WALK the most training examples. The power of the level 2 feature extraction algorithm can be seen in class TRAIN which can be classified very accurately even though only 18 training examples were provides. Table 2 shows a summary of precision values for the Neural Net and SVM models, where the full confusion matrices of all models can be found in Appendix B.

Table 2 Class precision of Neural Net and SVM

Class	Precision NeuralNet	Precision SVM
Walk	96.08%	95.19%
Train	95.24%	94.44%
Bike	88.89%	89.23%
Car	88.75%	85.71%
Tram	70%	71.43%
Bus	11.11%	50%
Boat	40%	83.33%
Coach	0%	100%

All trained classification models, as well as the filtering and segmentation algorithms can be tested on new data subjectively by using a user interface which visualizes the track data on a OpenStreetMaps map. The GPS track, its segments and points can be examined after each step of the whole procedure, allowing to test different parameters on algorithms or to see the classification result of the trained model on the currently loaded GPS track (as seen in Figure 8).

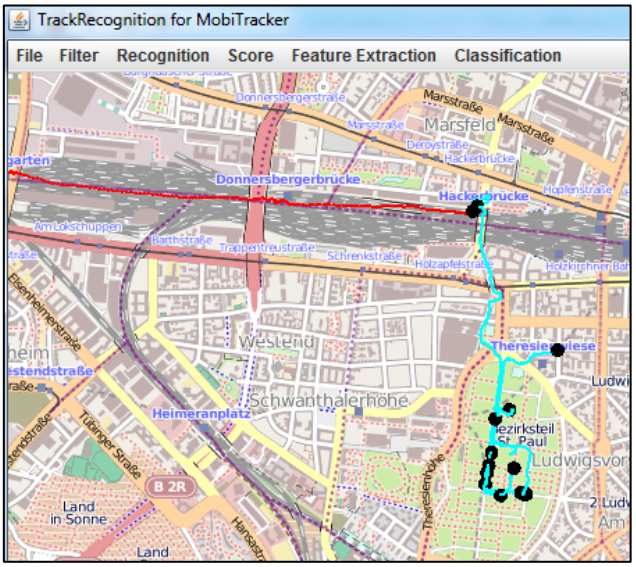

Figure 8 MobiTrackerSingle user interface for testing

5. Conclusion

The proposed process of segmentation and classification along with filter and feature extraction algorithms turns out to be very effective and could be realized in a productive system. Most of the defined types of transportation can be identified with sufficient precision. However, there are classes like BUS and COACH which

are currently quite hard to distinguish and the scored 100% in Table 2 can be credited to a fortunate folding because the model was only trained on 2 examples of class COACH. Hence, the models could perform even better on some classes when they are trained with more examples or a more accurate feature extraction algorithm (geocoded) is used to gather GIS features.

Although the segmentation algorithms could only be validated with a custom scoring technique without any reference, the subjective testing gives reason to say that good segmentation results can be achieved with the currently implemented methods. But there is still room for improvement in terms of flexibility by using Dynamical Bayesian Networks, especially segment models or Semi-Markov models.

References

Chang Chih-Chung and Lin Chih-Jen, *LIBSVM : a library for support vector machines*. ACM Transactions on Intelligent Systems and Technology, 2011. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

Murphy K. P. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

Murphy K. P. *Hidden semi-Markov models (HSMMs).* unpublished notes, 2002

Duong T.V., Bui H.B., Phung D.Q., and Venkatesh S. Activity Recognition and Abnormality Detection with the Switching Hidden Semi-Markov Model. *Proceedings of IEEE Conf. Computer Vision and Pattern Recognition*, vol. I, pp. 838-845, June 2005.

Liao L., Patterson D.J., Fox D., and Kautz H. Learning and inferring transportation routines. *Artifitial Intelligence*, 2006.

Shi Q., Wang L., Cheng L., and Smola A.. Human action segmentation and recognition using discriminative semi-markov models. *International Journal of Computer Vision*, 2008.

Mierswa I., Wurst M., Klinkenberg R., Scholz M. and Euler T. YALE: Rapid Prototyping for Complex Data Mining Tasks. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006. `http://rapid-i.com/`

Burges C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.

Hsu C.-W., Chang C.-C., and Lin C.-J. A practical guide to support vector classication. Technical report, Department of Computer Science, National Taiwan University, 2003.

Appendix A: Guide to MobiTracker

Two applications have been developed to train and apply the proposed algorithms on GPS data. MobiTrackerBatch is the interface to train classifiers on a dataset, as well as apply a trained model to predict classes. The MobiTrackerSingle application can be used to apply all filter, recognition and classification algorithms on a single GPS trace in order to validate and test the trained models.

It is recommended to execute the MobiTracker .jar files from a console, because of their experimental nature, lots of the status messages are only printed into the standard output. Additionally it is recommended to adapt the memory limits of the java application. To do this, one has to call the java application files with: `java -Xms512m -Xmx1024m –jar MobiTrackerBatch.jar`

The MobiTrackerSingle user interface can be seen in Figure 8 whereas the MobiTrackerBatch GUI resembles the process graph depicted in Figure 1. It also contains several tabs to configure the executed algorithms. The color coding of the MobiTrackerSingle user interface is stated in Table 3.

Table 3 Color codes of MobiTrackerSingle GUI

COLOR	TYPE OF TRANSPORTATION
BLACK	NOT CLASSIFIED
BLUE	CAR
RED	TRAIN
GREEN	BIKE
CYAN	WALK
YELLOW	BOAT
PINK	TRAM
GRAY	COACH

Note: All required libraries have been packed into the jar archives, so that the application does not have any dependencies except the configuration and image files in the folders config and img respectively.

Appendix B: Detailed result figures

This appendix contains the full results in form of confusion matrices and accuracy values of all trained models (RapidMiner and LibSVM models). All models were trained on 294 examples of which 102 are of class WALK, 80 CAR, 63 BIKE, 21 TRAIN, 10 TRAM, 9 BUS, 5 BOAT and 4 COACH. The confusion matrix of the SVM model only contains 290 examples because records with missing values are not included.

Naïve Bayes:

Table View Plot View

accuracy: 47.60% +/- 6.57% (mikro: 47.62%)

	true train	true bike	true walk	true car	true bus	true tram	true coach	true boat	class precision
pred. train	18	1	2	0	0	0	0	0	85.71%
pred. bike	1	38	47	4	3	0	0	0	40.86%
pred. walk	2	1	44	0	0	1	0	0	91.67%
pred. car	0	4	1	20	0	1	2	0	71.43%
pred. bus	0	19	4	47	6	1	0	0	7.79%
pred. tram	0	0	4	2	0	7	0	0	53.85%
pred. coach	0	0	0	7	0	0	2	0	22.22%
pred. boat	0	0	0	0	0	0	0	5	100.00%
class recall	85.71%	60.32%	43.14%	25.00%	66.67%	70.00%	50.00%	100.00%	

K-Nearest Neighbors:

Table View Plot View

accuracy: 59.82% +/- 7.50% (mikro: 59.86%)

	true train	true bike	true walk	true car	true bus	true tram	true coach	true boat	class precision
pred. train	7	1	1	5	0	0	3	0	41.18%
pred. bike	7	32	8	20	6	6	0	1	40.00%
pred. walk	2	9	89	7	0	3	0	2	79.46%
pred. car	5	21	4	48	3	1	1	2	56.47%
pred. bus	0	0	0	0	0	0	0	0	0.00%
pred. tram	0	0	0	0	0	0	0	0	0.00%
pred. coach	0	0	0	0	0	0	0	0	0.00%
pred. boat	0	0	0	0	0	0	0	0	0.00%
class recall	33.33%	50.79%	87.25%	60.00%	0.00%	0.00%	0.00%	0.00%	

Decision Tree:

Table View Plot View

accuracy: 86.09% +/- 4.53% (mikro: 86.05%)

	true train	true bike	true walk	true car	true bus	true tram	true coach	true boat	class precision
pred. train	20	1	2	0	0	0	0	0	86.96%
pred. bike	0	55	1	2	3	0	0	1	88.71%
pred. walk	0	2	97	0	0	0	0	2	96.04%
pred. car	1	2	1	71	5	4	3	0	81.61%
pred. bus	0	2	0	3	1	0	0	0	16.67%
pred. tram	0	0	0	1	0	6	0	0	85.71%
pred. coach	0	0	0	3	0	0	1	0	25.00%
pred. boat	0	1	1	0	0	0	0	2	50.00%
class recall	95.24%	87.30%	95.10%	88.75%	11.11%	60.00%	25.00%	40.00%	

Neural Net:

Table View Plot View

accuracy: 86.72% +/- 4.19% (mikro: 86.73%)

	true train	true bike	true walk	true car	true bus	true tram	true coach	true boat	class precision
pred. train	20	0	0	1	0	0	0	0	95.24%
pred. bike	1	56	1	3	3	0	0	1	86.15%
pred. walk	0	3	98	1	0	1	0	0	95.15%
pred. car	0	3	0	71	5	1	4	1	83.53%
pred. bus	0	0	0	1	1	1	0	0	33.33%
pred. tram	0	1	2	2	0	7	0	0	58.33%
pred. coach	0	0	0	1	0	0	0	1	0.00%
pred. boat	0	0	1	0	0	0	0	2	66.67%
class recall	95.24%	88.89%	96.08%	88.75%	11.11%	70.00%	0.00%	40.00%	

Support Vector Machine:

Cross Validation Accuracy = **89.65517241379311%**

Confusion matrix:

true:	**car**	**bus**	**train**	**bike**	**walk**	**boat**	**tram**	**coach**
car	72	4	0	2	0	0	4	2
bus	2	2	0	0	0	0	0	0
train	0	0	17	1	0	0	0	0
bike	3	3	0	58	1	0	0	0
walk	2	0	0	2	99	0	1	0
boat	0	0	0	0	1	5	0	0
tram	1	0	0	0	1	0	5	0
coach	0	0	0	0	0	0	0	2