David Piggott
Fitzwilliam College
dhp26

Computer Science Tripos Part II

Inferring Transportation Mode using Smartphone Sensor Data

May 16, 2011

Proforma

Name	**David Piggott**
College:	**Fitzwilliam College**
Project Title:	**Inferring Transportation Mode using Smartphone Sensor Data**
Examination:	**Part II Computer Science Tripos, July 2011**
Word Count:	**11, 610 (not including appendices, figures, or tables)**
Project Originator:	David Piggott
Supervisors:	Simon Hay and Mattias Linnap

Original Aims of the Project

To write a data logging application and use it to build up a data-set of smartphone sensor traces labelled with transportation mode ground truths. Using this data-set a number of transportation mode classifiers will be created and evaluated to determine which algorithms produce the most accurate inferences. As an extension a smartphone application for personal transport energy metering may be produced.

Work Completed

An application that records GPS location and satellite, accelerometer, light, orientation and magnetic field data was created and published on the Android Market. Six volunteers helped to collect 3000 km (100 hours/200 journeys) of data spanning eight cities in four countries.

A framework that generates features from the sensor traces was implemented. The framework builds and tests classifiers using subsets of the full feature set and evaluates them, providing a number of information retrieval metrics as well as insightful distribution and misclassification visualisations.

Special Difficulties

None.

Declaration of Originality

I, David Piggott of Fitzwilliam college, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signature:

Date:

Contents

List of Figures

List of Tables

Glossary

API Application Programming Interface. 5, 8, 9, 18, 25, 27

Azimuth The angle between the magnetic north direction and the axis passing through the conventional smartphone speaker and microphone locations. 31

DCT Discrete Cosine Transform. 13

DHMM Discrete Hidden Markov Model. 7, 8, 12, 39

DHT Discrete Hartley Transform. 13

DST Discrete Sine Transform. 13

DT Decision Tree. 7

EC2 Amazon Elastic Compute Cloud. 20

F-Measure The harmonic mean of the precision and recall metrics. 28

FFT Fast Fourier Transform. 4, 7, 13, 14, 25, 29

FP False Positive. 28

GPS Global Positioning System. 2, 4, 7–10, 15, 17–20, 23, 27, 29–33, 39, 40

GPX GPs eXchange format. 18

HCR Heading Change Rate. 7, 17

HTTP Hyper Text Transfer Protocol. 19

IDE Integrated Development Environment. 20

IO Input Output. 19

JVM Java Virtual Machine. 20, 21

KML Keyhole Markup Language. 18

kNN k-Nearest Neighbour. 7, 29–31

MAT Memory Analyzer Tool. 20

OS Operating System. 2, 25

OSM OpenStreetMap. 12, 18, 19, 39

PRN Pseudo-Random Name. 9

Rag From Wikipedia[1]: "University Rag societies are student-run charitable fund-raising organisations that are widespread in the United Kingdom and Ireland". 11

recall The number of segments correctly inferred by the classifier as belonging to the class in question divided by the total number of segments in the data-set actually belonging to that class. The ideal value is 1. 1, 2, 40

RF Radio Frequency. 32

ROC Receiver Operating Characteristic. 28

SD Secure Digital. 25

SDK Source Development Kit. 6

SIM Subscriber Identity Module. 25

SNR Signal to Noise Ratio. 2, 9, 15, 32

SR Stop Rate. 7

SSH Secure SHell. 12

TP True Positive. 39

UI User Interface. 22

UROP Undergraduate Research Opportunities Program. 40

USB Univseral Serial Bus. 11, 12

VCR Velocity Change Rate. 7

VDS Virtual Dedicated Server. 12

XML eXtensible Markup Language. 9

[1] `http://en.wikipedia.org/wiki/Rag_student_society`

Chapter 1

Introduction

The goal of the project was to develop and compare a number of algorithms for inferring what type of transport an individual is using, based only on data collected using their smartphone's sensors. In this context the term algorithm refers to the whole process of generating features from sensor data and then classifying instances of those features into transportation categories.

My preliminary research and testing showed that, in general, the choice of machine learning algorithm at the classification step makes little difference in terms of overall accuracy. Informed by this, I focused my efforts on developing and testing features for different sensor types. The most accurate classifier produced by the project has recall[1] equal to 0.978.

Figure 1.1 shows the main application view when idle. This application was, among other things, used as a tool for verifying data-set correctness and exploring different feature definitions.

1.1 Original Ideas for Classification Features

In addition to Global Positioning System (GPS) location and accelerometer data, the Android platform provides orientation data (derived from the accelerometer), light level data, magnetic field strength data and GPS satellite data. In preparation, I read a number of papers and did not find any which tested classifiers derived from light level, magnetic field strength or GPS satellite data sources. Consequently, I thought it would be interesting to develop such classifiers myself. When used to produce appropriate features of my own definition, some of these sources turn out to enable respectable inferences.

Classifiers using only orientation data features have recalls of up to 0.959. Other features such as GPS metadata (number of visible satellites and mean signal Signal to Noise Ratio (SNR)) also prove to be useful. Light level data turns out to be quite useless, primarily because the rate at which light level events are generated by the Android Operating System (OS) on the devices used is a few per hour, so there is very little data for classifiers to train or infer with.

It is tempting to hypothesise that the mean magnetic field strength may be a useful

[1] Metrics are defined fully in the evaluation chapter; for now, knowledge that this ranges from zero to one and that a value of one is desirable should suffice.

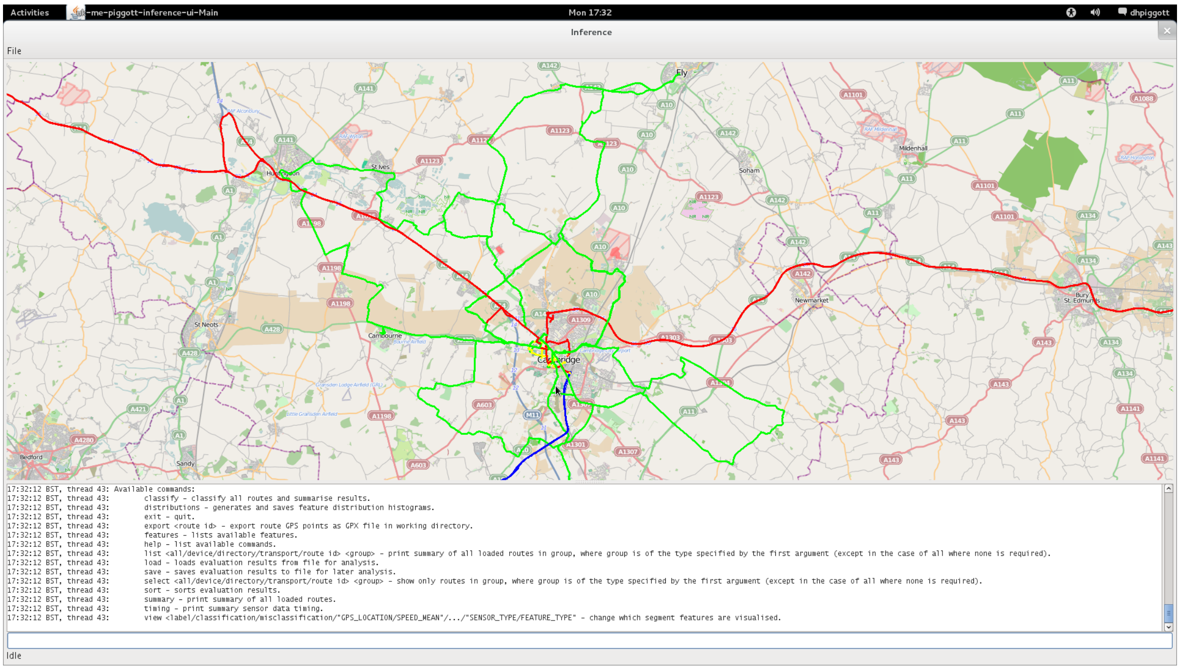


Figure 1.1: Inference Framework main view showing the data-set ground truths (green: bike, orange: bus, red: car, blue: train, yellow: walk). Newer routes are rendered on top of older routes.

feature on the basis that enclosed vehicles (cars, buses and trains) would act as Faraday cages, while cycling and walking would not. This cannot be the case, however, as Faraday cages can only shield against dynamic fields; static fields will pass. Surprisingly, I found that using only the mean field strength, a classifier with recall of 0.683 can be created. When other magnetic field strength features are added this reaches 0.715. This may be explained by local magnetic field variation across vehicle types due perhaps to the presence of electric motors on trains, or to the magnetisation of ferrous vehicle components.

1.2 Summary of Results

A summary of the findings in terms of the inference accuracies possible with different data sources is shown in figure 1.2.

1.3 Motivation

There are numerous ubiquitous computing applications which need inferred knowledge of transportation mode to function; requiring the user to manually input the transportation mode is too much to ask. While more accurate inferences may be possible using external sensors, having these as requirements prohibits widespread adoption.

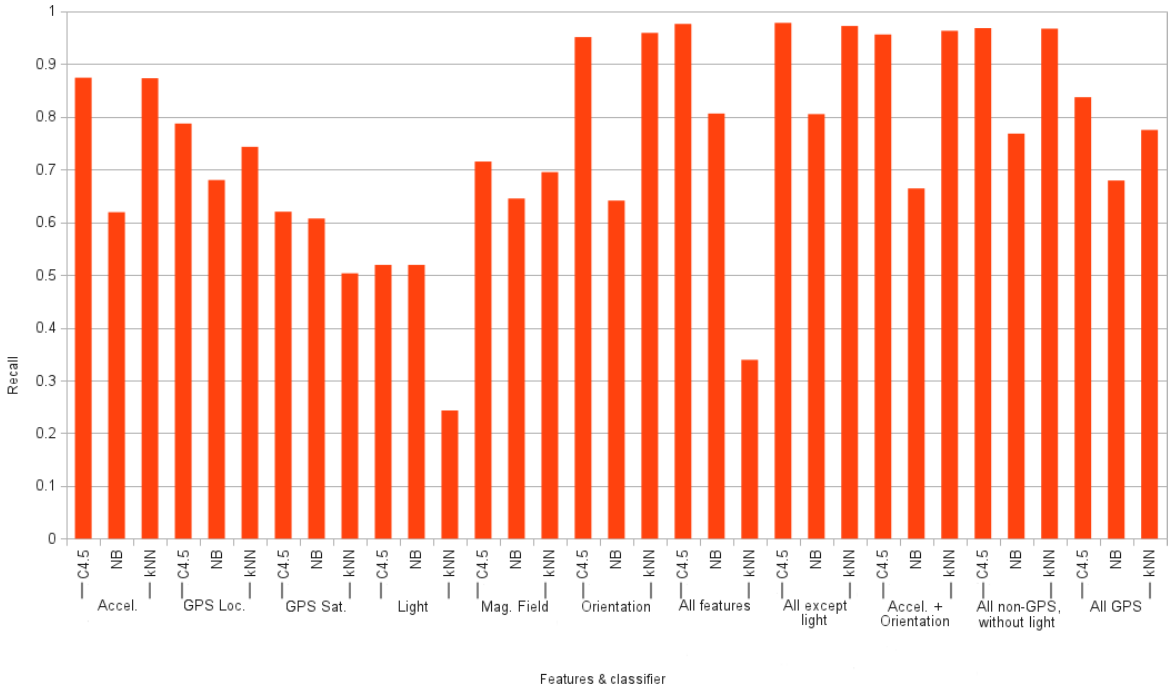


Figure 1.2: Chart comparing inference accuracy of sensor data sources.

1.3.1 Personal energy metering

A personal energy meter records an individual's daily energy usage, due for example, to use of building amenities, transportation, or consumption of goods. The idea is to promote changes in user habits through review of usage. The visualisations provided by the energy metering system help to provide quantitative and qualitative feedback that cannot be obtained by other means. Froehlich et al. [5] demonstrate the potential in such systems by semi-automatically tracking transportation behaviours and providing primarily qualitative feedback.

If the mode of transportation is known along with the number of people using it then all that must be done to obtain an estimate of the individual's carbon footprint is to multiply the distance travelled by the average CO_2 produced per mile for that mode of transport and divide by the vehicle occupancy.

1.3.2 Real-time traffic reporting and routing

Another application is that of an automated and crowd sourced congestion level inference system. It would work by having road users run a smartphone application that anonymously reports average speeds over a short interval, along with the road location that the measurement was taken on. A central system would maintain estimates of national traffic concentrations. Such data would be available in a form such that internet enabled satellite navigation devices can use the data to reroute based on real-time estimates of the road network congestion/link states, in a similar fashion to systems such as the internet itself.

Provided users could be assured of anonymity the idea works in terms of cost and reward; the majority of users would see their average journey times decreased with little to no effort. There is no barrier to adoption as there is no requirement for all road users to participate for it to be useful.

Transportation mode inference fits into this because the system would need to distinguish between pedestrians and cars stuck in traffic so as not to falsely report congestion. The main selling points (convenience and efficiency) of the system would be undermined by requiring users to manually input changes of transportation mode. Knowledge of transportation mode would also be needed if the system were to be extended to include reporting of estimated arrival times of public transport vehicles. For example, if the transportation mode is identified as bus then the system would go ahead and infer what bus route and journey time the rider is on and use this with location data to report estimated arrival and departure times for remaining stops on the route.

The benefit of such a scheme has been demonstrated by Thiagarajan et al. [6]. A simulation of the Chicago transit network found that expected wait times would be reduced by 2 minutes with as little as 5% of transit riders actually crowd-sourcing data for the system. With 20% of users sourcing data, the mean wait time would be reduced from 9 down to 3 minutes.

1.3.3 General purpose context awareness

The final use case is the more general context awareness one. This includes smartphone behaviours such as automatically switching to loudspeaker mode when the phone is in a moving car and being used by the driver. Determining whether the phone is in use by the driver may be tricky, but orientation data may help with this; if the phone is oriented with the screen perpendicular to the ground it is likely to be located in a car-mount and thus be the driver's phone.

1.4 Power Consumption

With all of the above applications power usage is a limiting factor that must be considered. GPS sensors in particular are known to reduce battery run times significantly if not used with care. Lu et al. [7] have developed a framework for continuous sensing on both the Apple iPhone and Nokia N95 that exploits patterns in user behaviour to reduce average power consumption through duty cycling and smart admission control. This uses inferences made from low power draw sensor data features (e.g. accelerometer magnitude) to determine when to activate higher power sensors (e.g. GPS) and/or more CPU intensive feature generation (e.g. Fast Fourier Transform (FFT) accelerometer coefficients).

Power consumption is considered in this project by generating and testing classifiers with and without GPS data so that the benefit in terms of accuracy increase can be seen. Whether or not GPS data would then be used in an application would depend on the specific cost-benefit in that scenario. For the personal energy metering application GPS data is required to determine the distance travelled anyway, so there is no benefit to battery run-time in not also using it for the classification.

Chapter 2

Preparation

2.1 Overall Structure

The project consists of two distinct applications. Data-flow between and within these is shown in figure 2.1. Route Tracer is the Android application I developed and used to create the training and test data-set. Inference Framework generates segments, segment features, classifiers, feature distributions and visualisations, and trains and tests the classifiers.

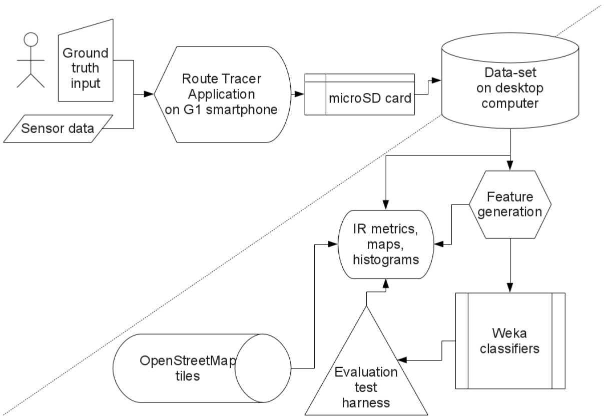


Figure 2.1: Data-flow visualisation.

Separating the construction of the data-set from the classifier exploration made sense for a number of reasons.

- The limited resources available on smartphones would at best have increased run times and at worst halted all progress. As the size of the data-set, number of features and number of classifiers increased, memory churn became an issue even on a machine with 12GiB of memory and necessitated a minor software redesign.

- Debugging smartphone applications is more time consuming due to the decoupling of IDE and target.

- Smartphones constrain the programmer by their Application Programming Interfaces (APIs) - the Android platform requires applications conform to a strict application life-cycle which would have added unnecessary work.
- To inform feature specification and generation I suspected that visualisation would be useful, and this would be best done with the screen size available on desktops.

2.2 Development Model & Schedule

The sooner data collection commenced, the more data would be collected by the end of the project and so rapid development of Route Tracer was necessary. Additionally, development of the Inference Framework could not progress without a small amount of data for testing. For these reasons development of Route Tracer was scheduled to be completed by November 3rd and the following two weeks were allocated to accelerated data collection, for early testing. A waterfall style development process was followed because the requirements were clear, the scale of the application was reasonably small, and the scheduling was important. Route Tracer was in a usable state by October 31st and was finalised on November 6th.

In order to really ensure that Route Tracer was recording useful data and to allow time for corrective measures if necessary, it was also important to get the Inference Framework into a minimal functional state as early as possible. This amounted to creating a stub classifier that mapped segments to transportation modes by testing which of five ranges the segment's average moving speed was in. Due to the scale of the application and number of unknowns, the Inference Framework required something more flexible so an iterative development process was followed.

2.3 Tools Used

For Route Tracer: Android Eclipse Plugin, Android Source Development Kit (SDK)
For the Inference Framework: Eclipse Memory Analyzer Tool, WindowBuilderPro
Common: Eclipse, Git, LaTeX, ObjectAid UML Explorer

2.4 Preparatory Learning

The Eclipse Memory Analyzer Tool, Git, JTransforms, Weka and WindowBuilderPro were all new to me so required learning. I had previous experience with the Android tools, Eclipse and LaTeX.

2.5 Platforms & Programming Languages

The decision to use Android devices was primarily informed by how this would affect data collection. Android was by far the most popular platform among potential volunteers, so choosing it would likely result in a larger data-set. My pre-existing Android knowledge would also minimise development time, allowing more time to be invested in the Inference

Framework. Java was the most appropriate language for the Inference Framework, given that Weka [8] is written in Java; this would ease integration.

2.6 Related Work

2.6.1 Accelerometer and GPS

Reddy et al. [1] use both accelerometer and GPS data to infer transportation mode with an accuracy of 98.8% in the classes {still, walk, run, bike, motor}. A number of classifiers (Naive Bayes, Decision Tree (DT), k-Nearest Neighbour (kNN), Support Vector Machines, Continuous Hidden Markov Model and a Discrete Hidden Markov Model (DHMM) enhanced DT) are tested. The 98.8% result is with the DT-DHMM algorithm, which uses human estimates for the DHMM state transition probabilities. The DHMM stage increases accuracy by 3.3% over the DT system but the DT on its own achieves accuracy of 95.7%; still better than the other classifiers.

Features: variance (accelerometer), energy (accelerometer), sum of FFT coefficients 0.5-10Hz (accelerometer), speed (GPS).

2.6.2 GPS only

Zheng et al. [2], [3] created and tested a number of classifiers, varying the segmentation method (fixed length, fixed duration and change point based), the generated features and the classification algorithm as well as introducing post-inference enhancement techniques using a database of change points combined with a DHMM. Classification in all cases is into classes {walk, car, bus, bike}. The most accurate system has accuracy of 76.2% (precision 51.6%, recall 81.8%).

Features: segment length, mean velocity, top three velocities, top three accelerations, variance of velocity, Heading Change Rate (HCR), Stop Rate (SR), Velocity Change Rate (VCR).

2.6.3 Accelerometer only

Manzoni et al. [4] created a classification system that using only accelerometer data classifies into {bus, metro, walk, bicycle, train, car, still, motorcycle} with an accuracy of 82.15% percent.

Features: 32 FFT coefficients, computed on a window 512 samples long (10.24 seconds), with a window overlap of 50% (5.12 seconds), signal variance.

2.6.4 Summary of related work

Table 2.1 shows a summary of the above described classifiers (for Reddy et al. and Zheng et al. the summaries relate to the most accurate system they created).

algorithm	Reddy et al. [1]	Zheng et al. [2] [3]	Manzoni et al. [4]
sensors	accelerometer, GPS	GPS	accelerometer
structure	fixed length segments → decision tree → discrete hidden markov model	walk/non-walk segments → decision tree → discrete hidden markov model → DHMM graph-based post-processing using change-point data	fixed length segments → decision tree →
accuracy	98.8% of segments correctly identified	76.2% of distance correctly identified	82.15% of segments correctly identified
classes	{still, walk, run, bike, motor}	{walk, car, bus, bike}	{bus, metro, walk, bicycle, train, car, still, motorcycle}
size	6 users / 20 hours of data	65 users / data collected over several months	4 users / "several" hours of data
device	Nokia N95	Independent (numerous stand-alone GPS devices)	Google Nexus One

Table 2.1: Summary of related work.

2.6.5 Findings

From the above described works I take:

- The idea of segmenting routes into fixed duration segments.
- The indication that the C4.5 DT and kNN classifiers are good for transportation mode inference.
- My own interpretation of HCR, VCR and SR as features for GPS location data.
- The indication that spectral coefficients are good accelerometer features.

2.7 Requirements Analysis

2.7.1 Recorded transportation modes

I decided to collect data for the classes {bike, bus, car, train, walk}.

The main factor I considered was popularity; considering the energy metering application, the accuracy would largely be determined by the amount of time the transportation mode is correctly identified[1]. This can only be done if the system is designed to include that transportation mode. Despite this, air travel is not included because smartphones may be switched off or unable to provide the necessary sensor data while in flight. For the energy metering application a better solution may be to mine flight reservations from emails.

Although it may be useful to distinguish between private cars, coaches and taxis, doing so would not be a realistic goal; intuitively there do not appear to be significant differences in the drive-time properties of private cars, coaches and taxis. In contrast, buses are expected to stop more frequently to let passengers on and off. Consequently, private car, coach and taxi journeys are covered by the car class.

2.7.2 Recorded sensor data

Route Tracer logs events from the following sources: {GPS location, GPS satellite, accelerometer, light level, magnetic field, orientation}. Table 2.2 summarises the values recorded for each sensor event type.

With the goal of the project being to compare different features and classifiers for transportation mode inference, the approach here was to be as broad as possible. Acceleration and GPS location are known to be good sources, and I had a number of ideas for how the other sensor data may be useful for inference. Sensor data such as gyroscope information is not recorded because Route Tracer was written to be compatible with the Android 2.1 API upwards. At both the time of writing Route Tracer and the time of writing this dissertation, Android devices with internal gyroscopes remain in the minority.

The idea with GPS satellite events is that a lack of information (in this case a low number of visible satellites or poor signal SNR) is information in itself. Previous experience on trains suggests the majority of the time is spent without a GPS fix (i.e. with no GPS location data available to the classifier) but with meta-data such as the satellite count and SNRs available. This is explained by the fact enclosed vehicles (trains, buses and cars) act as Faraday cages, reducing the amplitude of electromagnetic transmissions (in this case messages from GPS satellites) external to the enclosure. Magnetic field data may provide information through the variance of local magnetic fields across different vehicle types due perhaps to vehicle component magnetisation or the presence of electric motors.

Light level data may be useful if we suppose that bus users listen to music - presumably using their smartphone and perhaps keeping the device in their hand - more than users of other transportation modes. Recorded light levels on buses would be higher than

[1]Actually the cost of that transportation mode must also be considered; even if an individual only uses air travel once in a year it may still dominate all other modes in terms of energy usage.

transportation modes where users keep their phones pocketed. The expected orientation of a phone kept in hand will differ from one kept in a pocket and so the mean orientation[2] may also be useful.

Sensor event type	Value
Accelerometer	Acceleration minus G_x on the x-axis, acceleration minus G_y on the y-axis, acceleration minus G_z on the z-axis, accuracy. All accelerations in ms^{-2}.
GPS location	Longitude and latitude (degrees), altitude, accuracy (meters), speed (ms^{-1}).
GPS satellites	Pseudo-Random Name (PRN) and SNR (per satellite).
Light level	Light level (Lux), accuracy.
Magnetic field	X, Y and Z magnitudes (micro-Tesla), accuracy.
Orientation	From Android's SensorEvent class[a]: "Azimuth, angle between the magnetic north direction and the y-axis, around the z-axis (0 to 359). 0=North, 90=East, 180=South, 270=West Pitch, rotation around x-axis (-180 to 180), with positive values when the z-axis moves toward the y-axis. Roll, rotation around y-axis (-90 to 90), with positive values when the x-axis moves toward the z-axis. For historical reasons the roll angle is positive in the clockwise direction (mathematically speaking, it should be positive in the counter-clockwise direction)."

[a] `http://developer.android.com/reference/android/hardware/SensorEvent.html`

Table 2.2: Summary of recorded data.

Definitions

As specified by Android's SensorEvent class, "the coordinate-system is defined relative to the screen of the phone in its default orientation. The axes are not swapped when the device's screen orientation changes. The X axis is horizontal and points to the right, the Y axis is vertical and points up and the Z axis points towards the outside of the front face of the screen. In this system, coordinates behind the screen have negative Z values".

For accelerometer, light level, magnetic field and orientation events, accuracy is one of {unreliable, low, medium, high}, as specified by Android's SensorManager class[3].

2.7.3 Log format

At approx. 100MiB of information per day, as detailed in table D.1, efficient coding was important. While custom binary would enable the most efficient coding and eXtensible Markup Language (XML) may have been easier to develop readers and writers for, human

[2]The orientation data provided by the Android sensor API is a transform of the accelerometer data but is logged for completeness and to avoid unnecessary work both when programming and at run-time.

[3]`http://developer.android.com/reference/android/hardware/SensorManager.html`

readability was essential to allow quick sanity checking of data. With this in mind I elected to use a custom text format. Each line begins with a millisecond accuracy timestamp followed by labelled columns of data - the contents depending on the sensor type - separated by spaces. An example GPS location event line follows:

```
timestamp: 1298281761000 accuracy: 48.0 longitude: 0.10290026664733887...
           ...latitude: 52.21152126789093 altitude: 53.0 speed: 0.0
```

Gzip compression is used to reduce the storage requirements of this custom text format (the labels enable human readability but also mean there is a huge amount of redundant data). The standard Java `GZIPInputStream` and `GZIPOutputStream` classes are used to achieve approximately ten-fold compression. Files are saved to the device's external storage. There are six files per Route Trace - one per data source - with the following naming scheme:

```
deviceID_$DEVICE_ID_version_{1,2}_starttime_$MILLISECOND_ACCURACY_TIMESTAMP...
  ..._transport_{BIKE,BUS,CAR,TRAIN,WALK}_sensor_{ACCELEROMETER,...
...GPSLOCATION,GPSSATELLITES,LIGHT,MAGNETICFIELD,ORIENTATION}.routetrace.gz
```

2.8 Data Collection

2.8.1 Data-set visualisation and summary

Figure 2.2 and table 2.3 visualise and summarise the data-set. To ensure all routes traces had an associated ground truth I designed Route Tracer so that route tracing was activated by pressing the one of five transport class buttons matching the current vehicle type.

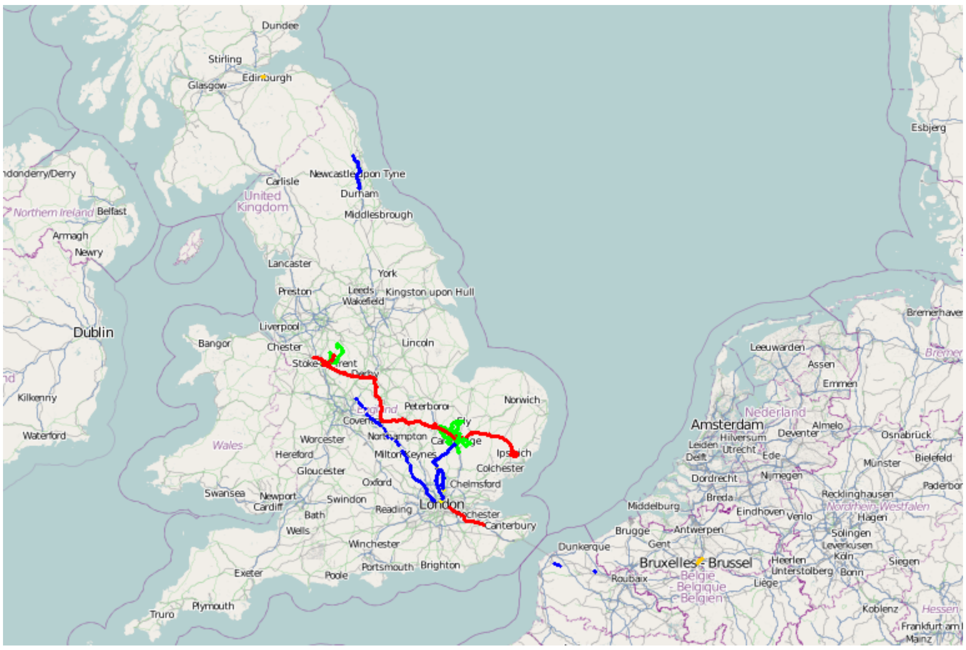


Figure 2.2: Data-set ground truth visualisation (green = bike, orange = bus, red = car, blue = train, yellow = walk). Newer routes are rendered on top of older routes.

	time	distance
devices		
1	87h 25m 01s	2,773.953 km
2	10h 17m 34s	397.238 km
3	01h 47m 18s	146.667 km
volunteers		
volunteer 1	01h 47m 18s	146.667 km
volunteer 2	02h 19m 55s	9.495 km
volunteer 3	88h 07m 04s	2,809.96 km
volunteer 4	01h 35m 32s	89.378 km
volunteer 5	02h 39m 05s	210.646 km
volunteer 6	03h 00m 59s	51.712 km
transport classes		
bike	45h 22m 51s	981.988 km
bus	03h 28m 37s	79.403 km
car	27h 16m 55s	1,611.907 km
train	05h 52m 15s	578.631 km
walk	17h 29m 15s	65.929 km
totals	99h 29m 53s	3,317.858 km

Table 2.3: Summary of data-set.

2.8.2 Volunteers

Volunteers were recruited through college and family connections. A total of six volunteers helped over a six month period to create the data-set which consists of over 3000km of Route Traces, over 200 routes and spans 8 cities in 4 countries. In order to ensure a varied and representative data-set could be created, a G1 smartphone was borrowed from the Computer Laboratory. This meant I was able to provide volunteers without Android smartphones a device to contribute data with.

2.8.3 Power management

For longer routes such as rail journeys with connecting stages, and for the volunteers who took the G1 to Belgium on their Rag Jailbreak, battery run-times would have been an issue. In anticipation of this, I built a universal charger with Univseral Serial Bus (USB) ports as outputs, powered by readily available AA batteries. This is shown in figure 2.3.

(a) Universal charger with USB ports

(b) Data collection on train

Figure 2.3: Universal charger and data collection on train.

2.9 Contingency Measures

Experience suggests that the principle cause of data loss is human rather than machine error, so the approach I took was to revision control everything unless there was good reason not to.

Route Traces are immutable and thus require no revision control which is fortunate given their size, reaching approx. 50MiB for long routes. The Route Trace directory of the project is excluded from revision control; this also helped maintain privacy and anonymity for volunteer contributed Route Traces. The OpenStreetMap (OSM) tile cache and heap-dumps are excluded from revision control and backups due to their size and reproducibility.

2.9.1 Revision control

I chose to use Git for revision control because its distributed model means that the whole project history is stored in the local working project directory as well as at whichever remote repositories the project is pushed to. This is beneficial in several ways, partly because it removes the need to separately backup the repository as would be required with centralised systems such as Subversion (the repository itself being almost as important as the current head). As well as a local Git repository, regular pushes were made to a remote Git repository held on a Virtual Dedicated Server (VDS) located in Maidenhead. The VDS itself is backed up by the service provider.

2.9.2 Backups

Daily backups of the working directory were made to the VDS using a cron job that runs rsync over Secure SHell (SSH). This protects files yet to be committed to Git and unversioned files such as the data-set itself. Manual off-line backup of the working directory (including the contained Git repository) was periodically made to a USB flash drive. This protects against the failure of the local machine and VDS and against human error - e.g. erasure of root file-systems - or a malicious third party. This backup was also performed using rsync.

2.10 Limitations of Data-set

From a high level view there were two directions I could have taken with logging strategy and ground truth labelling; online and offline labelling.

With online labelling as implemented in Route Tracer, users must press a button on the device every-time they change transportation mode and do so as close to the transition time as practical. This has the benefit of not requiring later recall of the day's events. The disadvantage is that occasionally volunteers forget to change the transportation label until several minutes after a transition, resulting in inaccurate ground truths and so reducing the accuracy of any results drawn from the data-set.

With offline labelling Route Tracer would have continually logged events on the device. A supporting desktop application would then allow volunteers to visualise their day's activities on a map and retrospectively add ground truth labels, using the map as a recall aid. The advantages would be a possible increase in ground truth accuracies and more data would be recorded per day. However, most smartphones - including the G1 - would only allow a maximum of approx. six hours of route traces on battery charge. It would be unreasonable to require volunteers to maintain battery charge over the course of the day. Additionally, the start of data collection would have been delayed by the requirement that the offline labelling tool also be completed and ready for use.

The type of labelling used has further reaching implications than the demands placed on volunteers. With online labelled routes inference techniques are restricted to memory-less segment based schemes. As a result I am unable in the work that follows to test algorithms which work by splitting continuous data streams into walk and non-walk segments as done by [2] and [3] or algorithms which work using DHMM techniques to enhance segment inferences based on transition probabilities ([1], [2] and [3]). Instead, all of my work focuses on determining the best algorithms and data sources for single segment inference.

Opting for the online labelling approach allowed the project to be broken down into more loosely coupled components than an offline labelling scheme would have allowed. This reduced the risk involved in scheduling, helping to ensure that the project was completed on time.

Chapter 3

Implementation

3.1 Libraries Used

Figure 3.1 shows how JTransforms and Weka fit into the Inference Framework.

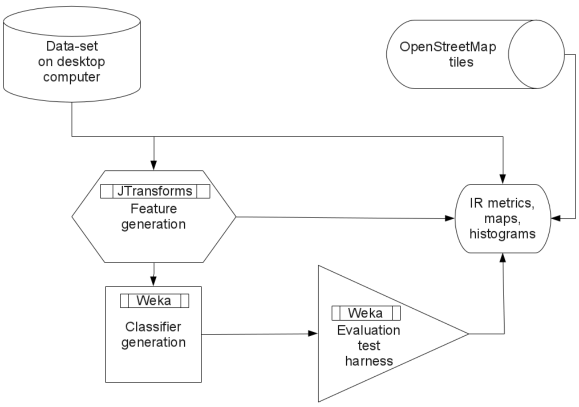


Figure 3.1: Inference Framework data-flow and library usage visualisation.

3.1.1 jline

In its first stages the Inference Frameworks's user interface was command line only. I used the jline[1] library to provide the system with tabbed command completion and history to decrease the amount of time spent typing while testing, thus increasing overall productivity.

3.1.2 JTransforms

To generate sensor features derived from spectral data, the incoming time domain data are transformed into the frequency domain using the JTransforms library[2] which includes

[1] `http://jline.sourceforge.net/`

[2] `http://sites.google.com/site/piotrwendykier/software/jtransforms`

FFT, Discrete Sine Transform (DST), Discrete Cosine Transform (DCT) and Discrete Hartley Transform (DHT) algorithms.

3.1.3 Weka

Weka [8] provides a number of machine learning algorithms as well as an evaluation system that I was able to integrate into the Inference Framework. This meant that I did not have to reimplement algorithms such as the C4.5 Decision Tree [9] and was able to spend more time exploring data sources and features.

3.2 Segment Features

3.2.1 Accelerometer, magnetic field and light level features

The same features are generated for accelerometer, magnetic field and light level data sources. Table 3.1 shows their definitions.

Because of the way Android's sensor event system works, sensor events are not produced at a fixed sampling rate. This complicates computing FFTs, which require the input data to have constant spacing between samples. One way to work around this would have been to interpolate between events to generate constant sample rate event streams. Doing this in Route Tracer was ruled out because implementing and validating an interpolation system would have delayed the start of data collection. Moreover, information would be lost and so further exploration would not be possible.

Manzoni et al. [4] encountered this same problem due to using an Android phone and successfully implemented an interpolation system. I opted for a different approach; I computed the variance across all events for a route and found that although sampling rates vary slightly, they are approximately constant. Over a ten second segment the number of events will be approximately 500[3]. Using this fact we can generate reasonably useful features by taking the FFT of these 500 events and merging the output bins into five larger groups. The act of merging the output bins reduces the error introduced by the fact that for segments that differ in the number of events they contain, the unmerged output bins actually correspond to different frequencies. The trade-off is that there are fewer features for the classifiers to use.

[3] 10 seconds at 50 events per second; the mean time period between events is 20 ms.

Feature	Definition
Low range spectral power (n = 1) *Low-mid range spectral power (n = 2)* *Mid-range spectral power (n = 3)* *Mid-high range spectral power (n = 4)* *High range spectral power (n = 5)*	The sum of FFT coefficients in the bins $\frac{n-1}{6} \times totalbins$ to $\frac{n}{6} \times totalbins$
Minimum magnitude Maximum magnitude	The {smallest,largest} magnitude in the segment, where for accelerometer and magnetic field events this is the Euclidean norm of the {x, y, z} event values and for light level events it is simply the light level.
Mean magnitude Magnitude variance	The {mean,variance} of the magnitudes of all events in the segment, where for accelerometer and magnetic field events this is the Euclidian norm of the {x, y, z} event values and for light level events it is simply the light level.

Table 3.1: Summary of spectral features. Emphasised features differ significantly from definitions in the literature. None of the reviewed literature implement magnetic field or light level features.

3.2.2 GPS location, GPS satellite and orientation features

Table 3.2, 3.3 and 3.4 show summaries of the orientation, GPS location and GPS satellite features.

Feature	Definition
X mean Y mean Z mean	The sum across all orientation events in the segment of the {X,Y,Z} orientation values, divided by the number of orientation events in the segment.
X variance Y variance Z variance	The variance of the {X,Y,Z} orientation values across all orientation events in the segment.

Table 3.2: Summary of orientation features. None of the reviewed literature implement orientation features.

Feature	Definition
Heading change rate	The number of times the bearing[a] between two location events is greater than 20°, divided by the number of events in the segment.
Velocity change rate	The number of times the speed between two location events differs by more than $2ms^{-1}$, divided by the number of events in the segment.
Stop rate	The number of location events for which the speed is less than $1ms^{-1}$, divided by the number of events in the segment.
Minimum speed Maximum speed	The {smallest,largest} speed of any location event in the segment.
Mean speed Speed variance	The {mean,variance} of the speeds of all location events in the segment.
Total distance	The sum of the great circle distances[a] between all consecutive pairs of location events in the segment.
Increasing altitude mean *Decreasing altitude mean*	The sum of altitude changes between all consecutive location event pairs for which altitude was {gained,lost} divided by the number of location event pairs between which altitude was {gained,lost}.
Maximum altitude gain *Maximum altitude loss*	The maximum {gain,loss} of altitude between any consecutive pair of location events in the segment.

[a] Computed using code taken from `http://android.git.kernel.org/?p=platform/frameworks/base.git;a=blob_plain;f=location/java/android/location/Location.java` which is in turn based on `http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf`.

Table 3.3: Summary of GPS location features. Emphasis indicates features that are either not present in the literature reviewed or that may differ from definitions in the literature.

Feature	Definition
Mean satellite count	The sum across all satellite events in the segment of the number of visible satellites in that event, divided by the number of satellite events in the segment.
Mean SNR	The sum of satellite SNR values across all satellite events in the segment, divided by the number of satellite events in the segment.

Table 3.4: Summary of GPS satellite features. None of the reviewed literature implement GPS satellite features.

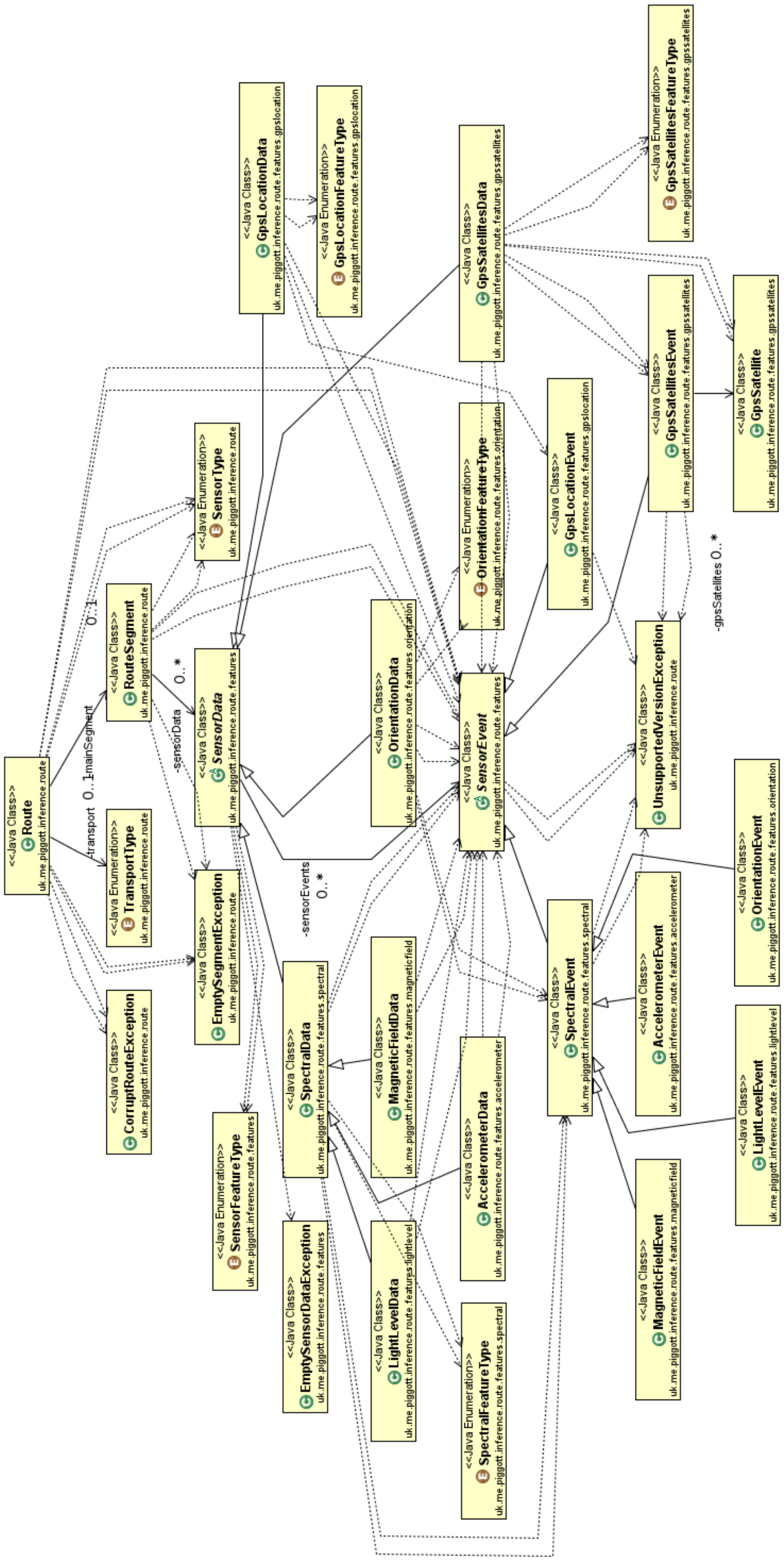


Figure 3.2: Inference Framework route and features class diagram.

3.3 Route Parsing and Representation in Memory

Figure 3.2 shows the class structure of route and feature areas of the Inference Framework. The key classes involved with representing routes in memory are described below.

3.3.1 SensorEvent

SensorEvent is an abstract class which contains data common to all SensorEvent implementations; in this case just the timestamp of the event. Implementations include additional information, such as lists of GPS satellites.

3.3.2 SensorData

SensorData is an abstract class which contains a single SortedMap from timestamps to SensorEvents. Implementations override the abstract `generateSensorFeatures()` method to compute features (e.g. HCR) by iterating through SensorEvents. It provides functionality common to all implementations, such as a method that computes a number of statistics on the timing of SensorEvents. Implementations of SensorData are constructed from a SortedMap of SensorEvents.

3.3.3 RouteSegment

A RouteSegment is a view onto six series (one per SensorEvent type) of contiguous SensorEvents. These SensorEvent series are held by six SensorData implementations. RouteSegments are constructed from six SortedMaps of SensorEvents.

3.3.4 Route

The Route constructor reads in the six `routetrace.gz` files associated with the route and constructs SensorEvent objects from the lines in these files. Which SensorEvent implementation is used is determined by which `routetrace.gz` file is being read. The Route constructor then constructs a main RouteSegment from these SensorEvents and keeps a reference to this, along with information such as the transportation mode ground truth. Route provides a method for obtaining RouteSegments of any duration. It uses the `subMap(T fromKey, T toKey)` method of the SortedMap interface to construct RouteSegments from SortedMaps that are backed by the SortedMaps of the main RouteSegment, saving memory.

3.4 Generic Feature System

The route feature system is designed such that the test harness is able to generate classifiers by inspecting the features provided by each SensorData implementation.

A single SensorType enumeration acts as the starting point for classifier generation. There is an entry for each sensor type that points to the corresponding SensorData and

SensorEvent implementation classes. Each SensorType entry also points to an enumeration (e.g. GpsLocationFeatureType) of sensor features which enumerates the features provided by that sensor's SensorData implementation.

The alternative to this system would have been to have the test harness look for getter methods matching a certain pattern, which would have made run time errors during development even more frequent.

The generic feature system was added around the same time that support for GPS satellite, light level, magnetic field and orientation data features was added. It became necessary in order to avoid having a large amount of volatile code in the test harness method which generates feature combinations for testing. Continuing to maintain this manually would have resulted in poor code and taken up time unnecessarily.

A much better solution would have been possible if Java were to support static methods in interface definitions or abstract static methods in abstract classes, and supertypes for enumerations.

3.5 Map Visualisation

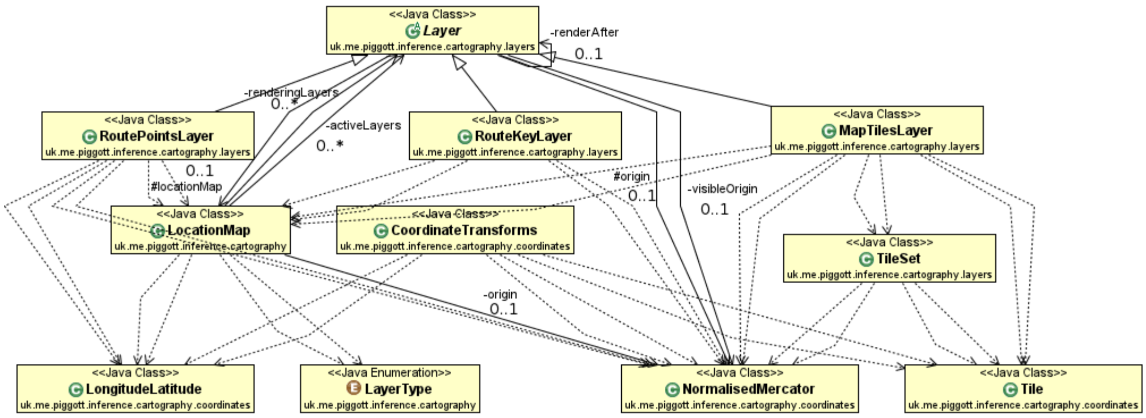


Figure 3.3: Inference Framework cartography components class diagram.

3.5.1 Motivation

Volunteers occasionally made mistakes and would either mislabel entire routes or parts of routes when changing transportation mode (e.g. forgetting to press the walk button until a minute after they'd got off a train). Volunteers were however able to report these mislabellings to me when submitting the route traces, allowing manual correction. Although it is possible with a minimal number of tools to manually edit Route Traces (`gunzip/gzip` to decompress/re-compress and `head/tail` to pull out sub-sequences of sensor events), doing so is not really practical because it is easier to work with locations than time-stamps when trying to locate transition points.

Anticipating that the number of mislabelled routes would over time be large enough that I would save time in the long run by writing a tool to aid relabelling by visualising points on a map and allowing for change point modification this way, I set about doing that by developing a system to overlay routes on OSM[4] tiles. The system would also be useful for gaining insight into the feature definitions, evaluating the classifiers, and debugging. The number of mislabelled routes was not large enough in the end to justify spending time developing a relabelling tool on top of the OSM layers.

I did not immediately decide to develop the mapping system from scratch and had first of all experimented with a number of mapping applications including GPS Prune[5], Google Earth[6] and jTileDownloader[7]. As part of this I added GPs eXchange format (GPX) and Keyhole Markup Language (KML) export functionality to the Inference Framework allowing any route to be saved as a GPX or KML file, enabling import into the above applications. Importing routes into any of these applications would have been a workable solution if the only requirement was sanity checking of incoming data, but even then it would have been a time-consuming process. I did modify the GPS Prune source code in order to partially automate the process, but the resulting solution was somewhat messy and would still only have been suitable for data sanity checking, not general purpose visualisations. This was because GPS Prune does not provide an API and so my modification was fairly tightly coupled to GPS Prune.

The final map visualisation system serves to sanity check data and to provide insightful colour-coded visualisation of any generated feature - such as mean speed by segment - and visualisation of other data such as misclassified segments. This is useful in informing feature design and optimisation.

3.5.2 Caching

OSM tiles are downloaded and cached in accordance with OSM policy: *"Tiles must be cached locally according to the Hyper Text Transfer Protocol (HTTP) Expiry Header, alternatively for a minimum of 7 days. A maximum of 2 download threads are allowed."*

3.5.3 Layer system

The map visualisation architecture is built around a layering system. An abstract `Layer` class serves to reduce code duplication by containing code common to all types. There are three layer types:

- `MapTilesLayer` downloads and stitches together map tiles from OSM.
- `RoutePointsLayer` visualises route segment points based on properties of the segment such as its transportation mode ground truth, or segment features such as mean GPS speed.
- `RouteKeyLayer` is a static overlay in the sense that the location of its contents on screen does not change as the map is panned. Despite this, implementing it as a layer makes sense because code duplication is still minimised this way.

[4] `http://www.openstreetmap.org/`
[5] `http://activityworkshop.net/software/prune/`
[6] `http://www.google.co.uk/intl/en_uk/earth/index.html`
[7] `http://wiki.openstreetmap.org/wiki/JTileDownloader`

3.5.4 Pre-rendering and multi-threading

The LocationMap system features multi-threaded rendering for responsiveness. This prevents the rendering of route points being delayed by the map tile renderer which may spend a lot of time blocked on network Input Output (IO). A large amount of the complexity in this code arises in conforming to the OSM policy that a maximum of two download threads are allowed; the LocationMap system ensures there is never more than one.

An off-screen border area is pre-rendered to improve panning performance: a border equal in width to the size of the visible portion of the map is maintained in order to prevent the user from waiting for tiles to be downloaded and layers to render when panning. Pre-rendering of new layers is triggered when the unrendered area of the currently rendered layers is within half a screen width of being made visible. Pre-rendering layers are not made live until they have been completely rendered, or (in the case of rapid panning or a slow connection) the previously rendered layers become insufficient for the map region being viewed. The visible portion (i.e. the centre) of layers being rendered is always prioritised to improve visible performance in this edge case. The zoom levels one magnification level above and below the current level are always pre-rendered in order to ensure smooth zooming. Unless panning or zooming is done very quickly the rendering process will never be seen.

3.5.5 Coordinate systems

The location map system involves calculations in four different coordinate systems. Although it may have been more efficient to implement these as primitives, in the interests of readability and thus correctness I wrote classes to represent points in these different systems. This way compile time type checking would prevent most mistakes.

- `LongitudeLatitude`: an immutable object constructed from a longitude and latitude, both doubles.

- `NormalisedMercator`: an immutable object constructed from an x and y value, both doubles and both in the interval [0,1]. (0, 0) is the North West extreme of a world map in the Mercator projection and (1, 1) is the South East extreme. The usefulness is that map transformations can be done with simple linear operations, working independently of any zoom level.

- `Tile`: an immutable object constructed from an x, y and zoomLevel, all integers, that represents an OSM tile. Zoom levels are integers in the range 0-18; the range of x and y values depends on the zoom level. The origin (0, 0) is the North West extreme of a world map in the Mercator projection.

- `Point`: the standard Java Point class (not my own implementation).

A `CoordinateTransforms` class provides static methods for conversion to and from all of the above four coordinate systems. Methods which convert a zoom level independent coordinate (`LongitudeLatitude` and `NormalisedMercator`) to a zoom level dependent coordinate (`Tile` and `Point`) also take a zoom level as argument.

Although efficiency gains could be had by maintaining a pool of coordinate objects and using factory methods to recycle those that are no longer in use, testing showed that there were no practical performance problems when naïvely always constructing new coordinate objects. It simply would not have been an efficient use of coding time.

The `LongitudeLatitude`, `NormalisedMercator` and `Tile` constructors clip the input arguments if they are outside the accepted range, e.g. `new NormalisedMercator(0.5, 1.1)` is equivalent to `new NormalisedMercator(0.5, 1)`. This behaviour is desirable because it pushes the detection of edge cases lower down, simplifying the higher level code in `LocationMap` and `Layer` implementations (if exceptions had been used instead, the catch bodies would just implement clipping themselves).

3.6 Memory Limitations

As the size of the data-set and the number of sensor types and associated features increased the run-time of the evaluation continued to increase until it got to a point at which it jumped up from less than an hour to several hours. Realising that this was due to memory churn (brought on at an earlier stage than it would otherwise have been due to the test harness multi-threading), I first sought to reduce it by increasing the Java Virtual Machine (JVM) heap size to 10GiB (this was the maximum practical value given the test machine had a total of 12GiB of physical memory). This worked as a stop-gap fix, but as the data-set and number of implemented features continued to grow the problem of memory churn returned, to the point at which the evaluation could be left for 24 hours and not make any progress.

Renting time on the Amazon Amazon Elastic Compute Cloud (EC2)[8] platform was briefly considered, but knowing that this would bring with it its own set of problems in the form of decreased productivity due to the separation of Integrated Development Environment (IDE) and target, I sought to reduce the memory requirements of the Inference Framework in order to allow it to progress on my local machine. Using the Eclipse Memory Analyzer Tool (MAT)[9], I was able to confirm my suspicion that the majority of heap memory was being used for SensorEvents, with memoized LocationMap layers being the the secondary culprit. As the SensorEvents are not directly needed for evaluation, I was able to refactor the Inference Framework and significantly reduce the run-time of the evaluation, returning it to its original duration of approx. 1 hour.

3.6.1 Route segment memoization

I modified the Route constructor so that it would automatically generate route segments of length 10,000ms and memoize them. Having done this the Route makes its main RouteSegment have all of its SensorData instances (of which there are six, one for each sensor-type) let go of their SensorEvent references[10]. On its own this optimisation does not help, but it paves the way for feature memoization which results in dramatic memory usage reductions.

[8] `http://aws.amazon.com/ec2/`

[9] `http://www.eclipse.org/mat/`

[10] GPS location events are kept in order that map visualisations can be drawn

3.6.2 Feature memoization

All SensorData implementations provide two types of features; timing and value features. The timing features are inherited from the abstract SensorData class that they extend while the value features are specific to the sensor type. When SensorData objects are constructed they also generate all implementation defined features. By doing this, the Route constructor can now safely clear each memoized 10,000ms RouteSegment's SensorData objects of their SensorEvents, allowing the garbage collector to reclaim that memory. Now each serialized Route object is also many times smaller, which enables loading speed-ups.

3.7 Optimisations

3.7.1 Map layer memoization

Once rendered, layers for zoom levels other than the current zoom level are memoized using Java SoftReferences so that a subsequent return to that zoom level can be done using the existing renders (provided the map region to be viewed is contained within the existing render). SoftReferences are used because each layer, being represented as a BufferedImage, is fairly resource heavy. Since they are not a necessity it makes sense to allow the JVM to garbage collect them when necessary.

3.7.2 Test harness multi-threading

The cross validation of Weka's classifiers is fairly time consuming when performed on the number of feature sets and classifiers this project tests, and with a data-set of the size used in this project. To combat this the test harness spawns a pool of worker threads equal in size to the number of available processors and uses these to perform the evaluation, resulting in a significant decrease in the run-time.

3.7.3 Data-set loading, caching and multi-threading

Loading in every route in the data-set from disk each time the Inference Framework was launched took a significant amount of time, and as the data-set was growing in parallel with the Inference Framework this time would only increase. As each addition to the code would require several launches of the Inference Framework for testing, this would have wasted a lot of time. To combat this, I implemented a caching system that works by making routes serializable. When a route is loaded for the first time, it is also serialized and written out to disk alongside the six sensor files, with the name format:

```
deviceID_$DEVICE_ID_version_{1,2}_starttime_$NANOSECOND_ACCURACY_TIMESTAMP...
        ..._transport_{BIKE,BUS,CAR,TRAIN,WALK}.routetrace.cache
```

When first implemented, the first run with this caching took 165s to read in the then 400MiB data-set (due to the cache write-back), and subsequent loads took 55s. Prior to this it took 80s to load in the data-set. The size on disk of cached routes was the same order of magnitude as their non cached forms, which is expected; the speed-up from caching results from not repeating the parsing stage of reading in event lines to construct sensor event objects.

Some time after implementing the caching system I further improved the route loading stage through multi-threading, that is, creating a pool of worker threads equal in size to the number of available processors and having them pull jobs from a sorted set of prefixes until no more remain. The prefixes are strings representing the route traces that had been found in the scanning stage, and the reason for sorting them is so that routes are assigned route id numbers in chronological order, helping human comprehension of the route listing.

The switch to multi-threaded route loading resulted in a speed-up from 100 to 35 seconds (the data-set now consisted of 286[11] routes). The speed-up results from the fact that with the single threaded approach the route loading was not limited by disk access speeds but by the rate at which the JVM could allocate memory. Using multiple threads helps approach the limit of disk access speeds - the single threaded rate of 8MiB/s suggested further improvements could be made - by increasing the rate of memory allocation. Table 3.5 shows the overall result of this work.

Optimisation	Speed-up
Caching	1.45x
Multi-threading	2.85x
Segment and feature memoization, SensorEvent garbage collection	11.6x
Total speed-up	47.9x

Table 3.5: Route loading speed-ups.

I hypothesised that an additional speed-up could be gained by changing the SensorEvent system to not have each SensorEvent be a new instance but instead have a singleton SensorEvent that would represent all sensor events using a number of arrays. A SensorEvent instance in such a system would just be an array index (returned by the singleton SensorEvent when a new event is constructed) which is then given to SensorEvent accessor methods which wrap access to the arrays. The speed-up from using a system like this would result from a reduction in the number of objects created from $O(n)$ to $O(1)$ (in terms of the number of sensor events n). I went ahead and implemented this alternative system and found that it did not work so well because:

1. Knowing how many sensor events would be created in advance of constructing the singleton SensorEvent proved difficult, and using resizeable data structures instead of arrays would largely defeat the point.

2. If the singleton pattern were to be used in conjunction with caching it would require the cache to be ignored and regenerated each time a route was added to the data-set, limiting its usefulness.

[11]This includes corrupt routes and routes less than 30 seconds in length; the data-set size reported elsewhere is smaller because these are filtered out before the evaluation runs.

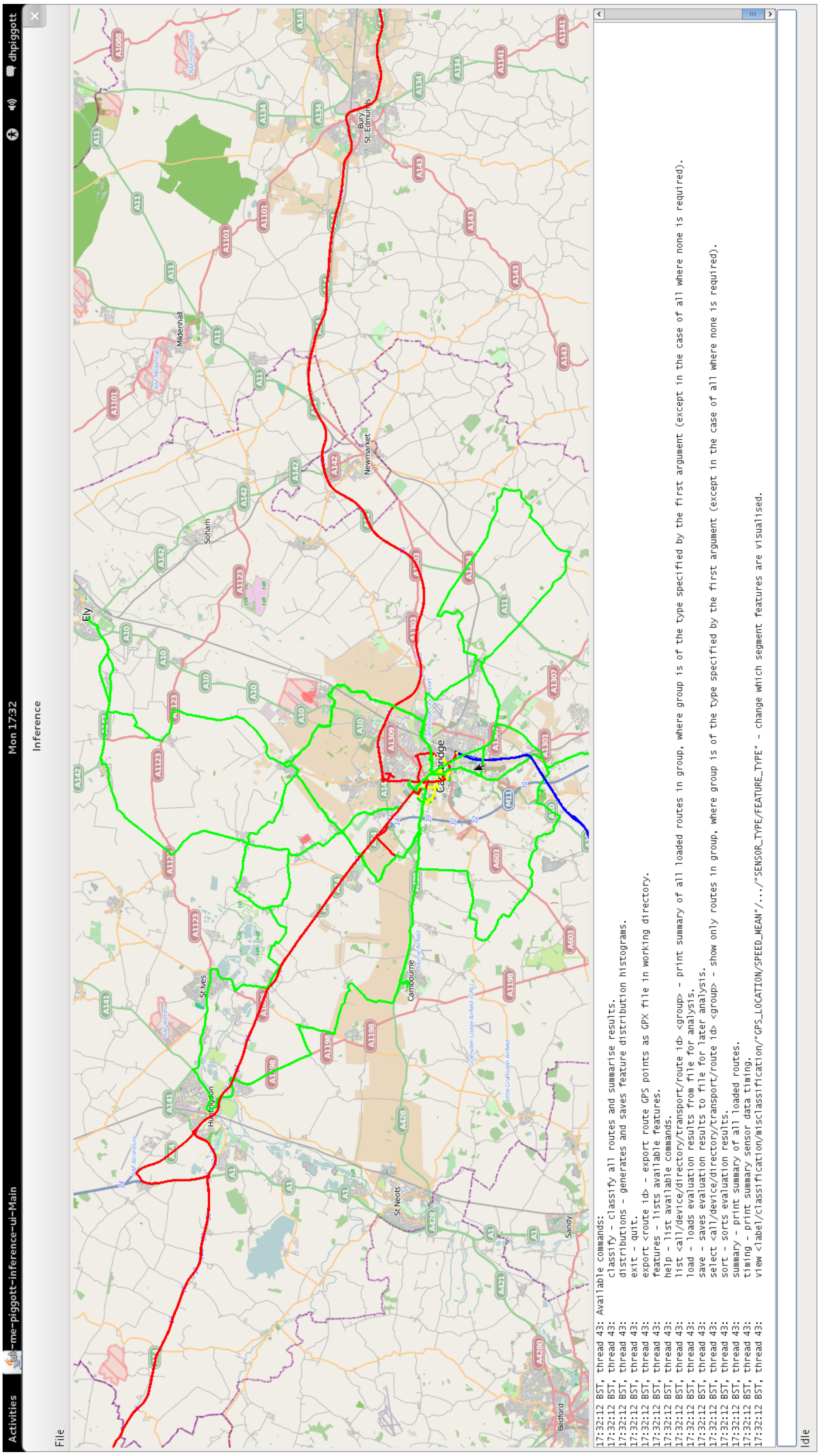


Figure 3.4: Inference Framework main view showing the data-set ground truths (green: bike, orange: bus, red: car, blue: train, yellow: walk). Newer routes are rendered on top of older routes.

3.8 Graphical User Interface

The Model View Controller pattern is used at a high level for the user interface. Figure 3.5 shows the structure of the top level and User Interface (UI) components of Inference Framework.

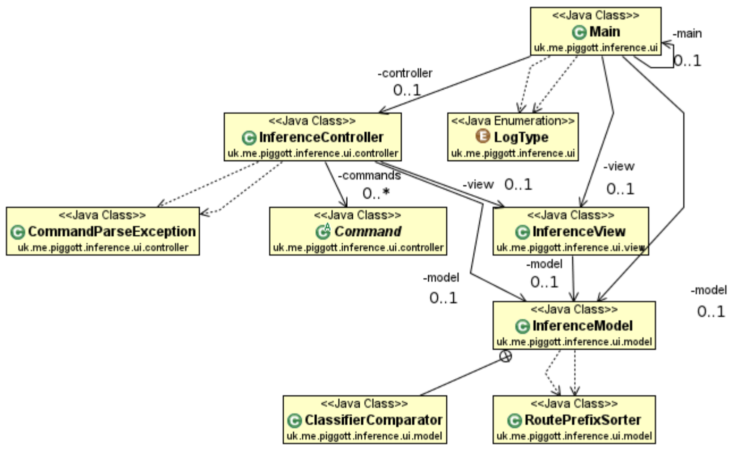


Figure 3.5: Inference Framework user interface class diagram.

3.8.1 Inference Model

InferenceModel contains methods for finding and loading routes from their on disk representations, using the multi-threading and caching techniques described previously. It also provides and maintains mappings from device groups to routes, directories to routes, transport types to routes, and route ids to routes; these are used by the controller to efficiently select subsets of the loaded routes for viewing and when summarising the data-set. A selected routes collection which can be set using a select command contains the routes to be used by commands which work on routes (e.g. evaluation and visualisation commands). InferenceModel also contains the high level method which initialises the evaluation.

3.8.2 Inference View

Figure 3.4 shows a full view of the Inference View layout, which was constructed with the help of WindowBuilderPro[12]. InferenceView does not contain any application logic, instead providing only methods to register action and event listeners with the various components it contains. It provides a split pane view, with the top pane containing a view onto the LocationMap (which is drawn using LocationMap's `drawImage(Graphics graphics)` method). The bottom pane contains a text area and text field which together enable console style interaction; commands are typed into the text field and output is displayed in the text area. Opting for primarily text based interaction - with the exception of panning and zooming the map - benefited both development time and the

[12] `http://code.google.com/javadevtools/wbpro/`

speed with which the Inference Framework can be used, especially once command history functionality was added back[13].

The observer/observable pattern is employed by the LocationMap and InferenceView classes; LocationMap is Observable and the InferenceView registers with it for updates. The LocationMap marks itself as changed whenever more data is drawn to a visible and rendering layer (e.g. when a tile completes downloading, or a route segment completes rendering). Combined with the fact that rendering is asynchronous this ensures the Inference Framework feels responsive. When the data-set has been loaded or when the visible route selection changes the map zoom level and origin are updated so that the visible routes are centred on screen. This is done via a call to a `recentreMap` method in LocationMap which iterates through all GPS location points in the current route selection, updating its record of the minimum and maximum longitudes and latitudes in the visible route selection as well as the upper and lower limits of the feature selected for visualisation. Private helper methods `calculateInitialOrigin` and `calculateInitialZoomLevel` help to achieve centering of the routes on screen.

3.8.3 Inference Controller

InferenceController registers callback methods with InferenceView that handle command input, command history scrolling and mouse movement corresponding to map zooming and panning. The command system fits the Command pattern[14] and is built around an abstract Command class which has an abstract `run` method. The InferenceController constructor puts a number of anonymous inner classes which extend Command into a map from Strings to Commands, where the String is the name of the command. Implementing commands as inner classes prevented a lot of otherwise redundant accessor methods being needed in InferenceModel. When a command is entered it is retrieved from the map and its run method is called with the command line as argument.

[13]jline was not used because tabbed completion would no longer have been beneficial to productivity, and it was less work to implement a command history from scratch than to reintegrate jline.

[14]`http://en.wikipedia.org/wiki/Command_pattern`

3.9 Route Tracer Implementation

Route Tracer is the Android application used by volunteers to record test data.

3.9.1 Activity and Service Compartmentalisation

Route Tracer is split into a back-end part which handles sensor and location events and logs them to the Secure Digital (SD) card, which is implemented as an Android Service, and a front-end user interface which allows changing of preferences, live viewing of sensor data, and controls the creation of Route Traces. The front-end consists of a number of Android Activities.

Aside from being good practice in terms of code readability and modularity this was important because the Android Activity life-cycle permits Activities to be destroyed by the OS at any time in order to free up resources for foreground Activities. Doing the logging in a Service ensures it is not interrupted when the system load is high.

3.9.2 Route trace submission

When a volunteer had a collection of `routetrace.gz` files ready to be added to the data-set the files were either emailed to me or returned with the G1. Having Route Tracer automatically upload Route Traces would be inappropriate as the G1 was used Subscriber Identity Module (SIM)-free the majority of the time, and volunteers using their own phones might not appreciate an application using bandwidth unnecessarily.

3.9.3 Timing problems

sensor	mean (ms)	minimum (ms)	maximum (ms)
accelerometer	21.196	3.284	166.002
gps location	1,000	1,000	1,000
gps satellites	-	-	-
light level	-	-	-
magnetic field	21.098	3.409	162.154
orientation	22.588	3.746	163.772

Table 3.6: Mean values of sensor event time period properties for a typical route.

The way that the Android sensor and location APIs work is problematic for the generation of some sensor features as Android provides no guarantees on the rate at which events are generated; as a result, sampling rates are not consistent. This is particularly problematic for spectral features which require FFT transforms of sensor value magnitudes. The timing statistics across all sensor types, shown in table 3.6, illustrate this.

There was some appeal to developing Route Tracer in a way such that it would hide the the fact that sampling rates are not constant in that it would make the correctness of feature generation code easier to reason about. However, I elected not to do this as it would only increase the complexity of Route Tracer and increase its run-time overheads, while not actually increasing the amount of information contained in the Route Traces. Route Tracer instead works by doing only the essential job of logging sensor events at whatever rate they are generated by the OS.

3.9.4 Android Market release

I published Route Tracer on the Android Market on December 17th; by this point it had several weeks of use and so was known - to some degree - to be correct. This made distribution to volunteers who chose to use their own Android smartphones easier. Figure 3.6 b shows the Route Tracer Market Page.

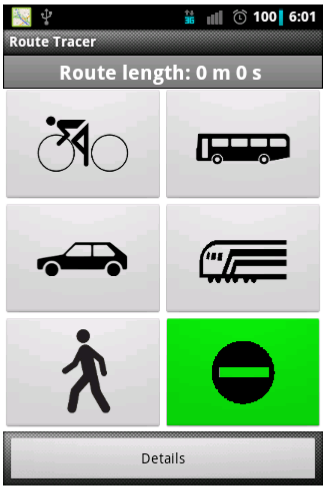
A minor update was released on December 18th adding backward compatibility with Android 2.1; at that time and at present a significant fraction of devices run Android 2.1[15]. At the time of writing this dissertation, Route Tracer has been downloaded 2,390 times, and there are 357 currently active installs.

3.9.5 User interface

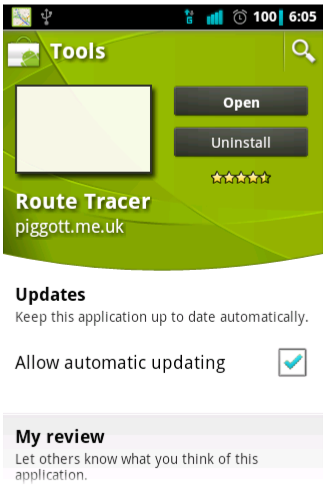
With the goal of collecting as much data as possible and with as accurate labelling as possible it was clear that Route Tracer would best achieve this by being as simple to use and as minimalist as possible in its design. This I achieved by having the main Activity be an arrangement of six buttons, with icons indicating which mode of transportation they initiate logging for (which button is pressed determines only the labelling of the `routetrace.gz` files produced).

A smaller button brings up the live sensor data view Activity (used primarily during development for debugging but left in for interested users). The main Activity menu button provides access to the Preferences activity and an About page, showing device and application version information. The main Activity works in both portrait and landscape orientations by defining two LinearLayouts for the button arrangement. The Android platform takes care of switching between these on an orientation change. Figures 3.6 a, c, d and e shows these four Activities.

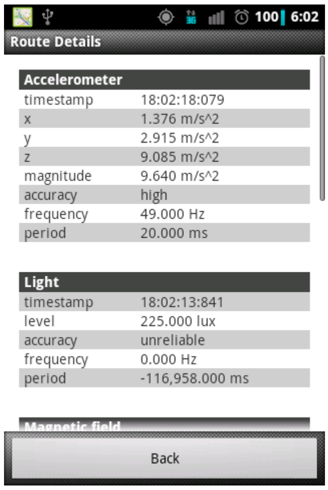
[15] `http://developer.android.com/resources/dashboard/platform-versions.html`



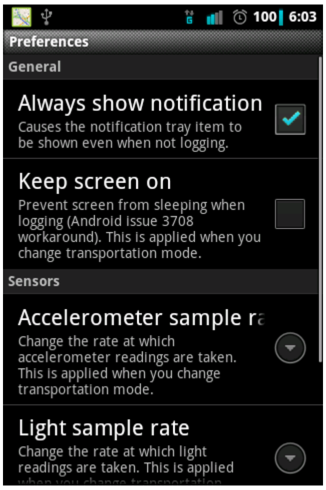
(a) Main Activity



(b) Route Tracer Market Page



(c) Details Activity



(d) Preferences Activity

(e) About Activity

Figure 3.6: Route Tracer sub activities

3.10 Technical Issues

3.10.1 Sensor API problems

Within days of beginning Route Tracer development it became apparent that a large number of Android devices suffer from a bug[16] that prevents delivery of sensor events to applications when the device's screen is switched off. Early builds of Route Tracer contained a hook that would cause a distinctive sound to be played when the device was in free-fall that enabled quick testing of devices to determine whether they suffered from the bug.

My own phone - an HTC Hero - suffers from the bug. I implemented in Route Tracer the work around suggested at Stack Overflow[17] but it didn't help and no matter which Android version I installed on the HTC Hero it wouldn't provide sensor events with the screen off. Conversely, no matter which Android version I installed on the G1, it would always provide sensor events with the screen off. I flashed CyanogenMod[18] 6.1 onto the G1 so that more recent Android features could be used.

I put an alternative work around for the bug in Route Tracer, in the form of a preference that can be set to make Route Tracer maintain a power lock when logging data. This prevents the screen from switching off by time-out. However, the screen will still switch off if the user presses the device's power button. This meant volunteers with affected devices would need to ensure they kept their phone's screen on or the only data that would be recorded would be GPS location and satellite data. Asking this of volunteers would be unrealistic; they inevitably do switch devices off, both through accidentally knocking power buttons and by forgetting not to switch the device off.

The preliminary volunteers I had lined up to create the data-set all had phones affected by the Android sensor API problem but I was able to rotate the G1 amongst the volunteers and still produce a reasonable data-set.

[16] `http://code.google.com/p/android/issues/detail?id=3708`
[17] `http://stackoverflow.com/questions/2143102/`
[18] `http://www.cyanogenmod.com/`

Chapter 4

Evaluation

4.1 Performance Metrics & Terminology

The Inference Framework makes use of Weka's Evaluation class to compute a number of information retrieval metrics for each classifier. These metrics are defined below.

Recall: The number of segments correctly inferred by the classifier as belonging to the class in question divided by the total number of segments in the data-set actually belonging to that class. The ideal value is 1.

Precision/true positive rate: The number of segments correctly inferred by the classifier as belonging to the class in question divided by the total number of segments inferred as belonging to the class in question. The ideal value is 1.

F-Measure: The harmonic mean of precision and recall, i.e. $F = \frac{2 \cdot precision \cdot recall}{precision + recall}$. The ideal value is 1.

False Positive (FP) rate: The number of segments incorrectly inferred by the classifier as belonging to the class in question divided by the total number of segments inferred as belonging to the class in question. The ideal value is 0.

Receiver Operating Characteristic (ROC) area: The area under the curve ((0, 0) (FP rate, true positive rate), (1, 1)). The ideal value is 1.

I have chosen to use recall as the primary metric for accuracy when comparing classifiers because I consider it to be the closest match to the human intuition of accuracy in this context. Moreover, as classifiers can only label each instance as belonging to one class there is no need to introduce another metric to measure error rates; misclassifications are evident as reduced recall.

4.2 Selected Instance Classifiers

The following Weka instance classifiers were selected for evaluation. This selection was informed both by findings from the literature and by my own experimentation.

- Naive Bayes [11]: This serves as a baseline for all other classifiers as it is most simple algorithm.
- C4.5 Decision Tree [9]: The de facto classifier, used by most other works and found to be the best performer in speed and accuracy. The Weka implementation is labelled J48.
- k-Nearest Neighbour [10]: Not as good an all rounder as the C4.5 algorithm but a strong contender nonetheless. The Weka implementation is referred to as IBK.

4.3 Combined Feature Classifier Results

The most accurate classifiers result from combining multiple feature and sensor types together, which provides the classifier algorithms with more information to work with. Figure 4.1 shows a comparison of single sensor feature groups and multiple sensor feature groups. Full classifier evaluation tables are included in appendix A. The chart illustrates that the choice of features causes far more variation than the choice of classifier.

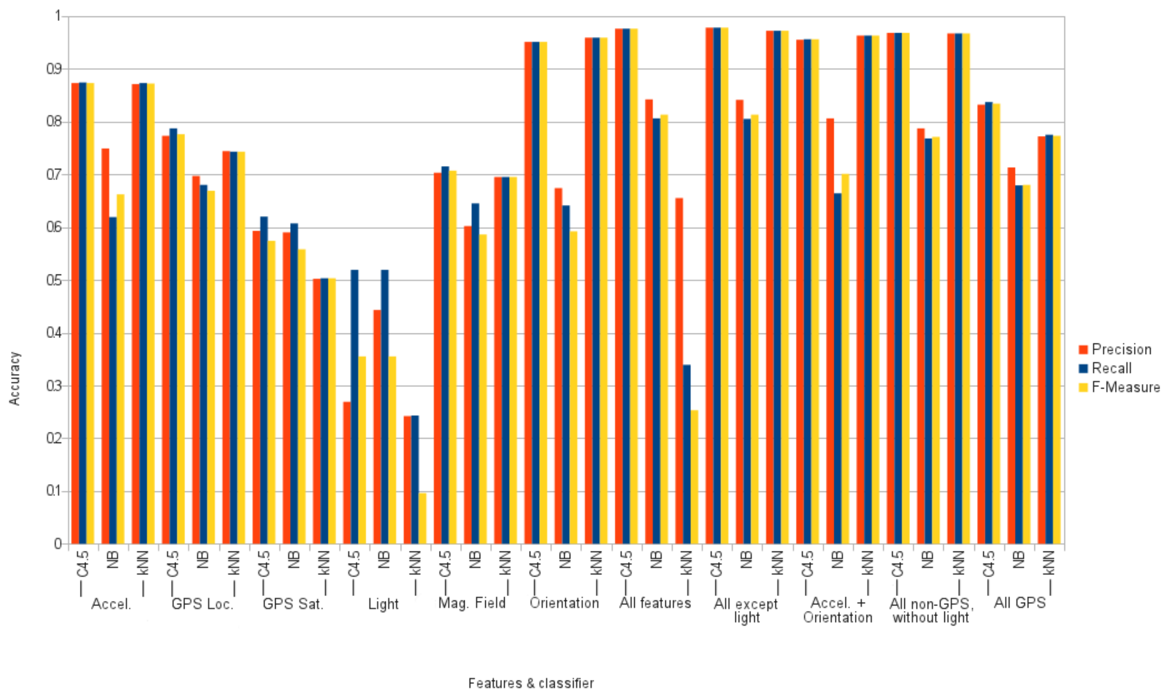


Figure 4.1: Chart comparing accuracy of sensor features and classifier combinations.

4.3.1 Accelerometer

Accelerometer data proves to be a good source, despite the variation in sample rates and the novel FFT approach taken. C4.5 and kNN classifiers both produce good results with recalls of 0.874 and 0.873 respectively; this is on par with the work of Manzoni et al. [4] (recall 0.822).

4.3.2 GPS location

The C4.5 and kNN recall rates of 0.787 and 0.743 are respectable. These are similar to the results of Zheng et al. [2], [3] (recall 0.818).

4.3.3 GPS satellites

The recall rates of 0.620 and 0.503 for C4.5 and kNN classifiers show that GPS satellite data does contain some relevant information but not enough to consider it as a standalone data source.

4.3.4 Light level

As expected, due to the very low event creation rates of Android, there is insufficient light level data for the recall results of 0.519 (C4.5) and 0.243 (kNN) to be conclusive. It is worth nothing however that the C4.5 recall of 0.519 is likely to have been influenced by the data-set class prior distributions (despite resampling) and so be an overestimate.

4.3.5 Magnetic field strength

The magnetic field strength features give promising results with recalls of 0.715 (C4.5) and 0.695 (kNN). However this is probably not enough to use as a standalone data source.

4.3.6 Orientation

The orientation data produce very accurate results of 0.951 (C4.5) and 0.959 (kNN). It is interesting to note that the kNN classifier breaks the trend and outperforms C4.5. The inferences are more accurate than those made with accelerometer data and rival those of the multiple-sensor classifiers. None of the reviewed literature have tried using orientation data. One thing to consider, however, is that of all the data sources tested, orientation is the source that is most likely to vary between different users. Due to the small number of volunteers (8), the results here may be overestimates of what could be expected if the system were used by a larger number of people.

4.3.7 All features

The all features classifiers are worth looking at because in theory they should be the most accurate, having the largest amount of information to work with. The C4.5 classifier yields recall of 0.978 while the kNN classifier gives recall of 0.339. This anomaly may be explained by the fact that the kNN learning algorithm may be susceptible to confusion by extreme outlying values; which are more likely to be present in a larger feature set.

4.3.8 All except light

Given that the light level data do not appear to enable inference of any accuracy I thought it would be interesting to remove light level data from the feature set and compare to the all features results. The C4.5 recall actually increases to 0.978 though this is most likely due to random error resulting from the cross validation and re-sampling (which are done on a per-classifier basis). The kNN recall increases from 0.339 to 0.972, which leads me to conclude that the kNN algorithm cannot handle spurious features very well.

4.3.9 Accelerometer and orientation

Android's orientation data is actually derived from accelerometer values. Based on this the extra power drawn by using orientation data should be minimal. Therefore in any power sensitive application that used one of these sources it would make sense to use the other as well, given the minimal extra cost. The combined recall possible is 0.956 with C4.5 and 0.963 with kNN; marginally better than orientation data alone.

4.3.10 Non-GPS

GPS receivers draw more power than the other data sources combined: Kjaergaard et al. [12] found that the Nokia N95 GPS receiver uses approximately six times as much power as its accelerometer. Power sensitive applications might wish to avoid using GPS data for this reason, but use all other features to obtain the highest recall possible without excess power drain. Not using light level data, the expected recall would be 0.968 with the C4.5 algorithm and 0.967 with the kNN algorithm.

4.3.11 GPS

The previously mentioned Android sensor event power bug[1] prevents applications on some Android devices from using sensor data when the device's screen is off. In such cases an inference system would essentially be working with only GPS data (location and satellite) and could expect recall of 0.837 (C4.5) or 0.775 (kNN).

[1] `http://code.google.com/p/android/issues/detail?id=3708`

4.4 Single Feature Classifier Results

Full classifier evaluation tables are included in appendix A. The comparison here is with just the C4.5 Decision Tree classifier.

4.4.1 Accelerometer and orientation features

Figure 4.2 shows the performance of individual accelerometer and orientation features. From this we can see that the orientation X mean is the least useful feature. Given that the X mean is defined as the Azimuth of the device this makes sense; in theory it should not contain any useful information.

Y (pitch) and Z (roll) values contain useful information relating to how the user is carrying or using the device, which will vary depending on whether they are walking, cycling or seated in a vehicle, hence they enable high recall.

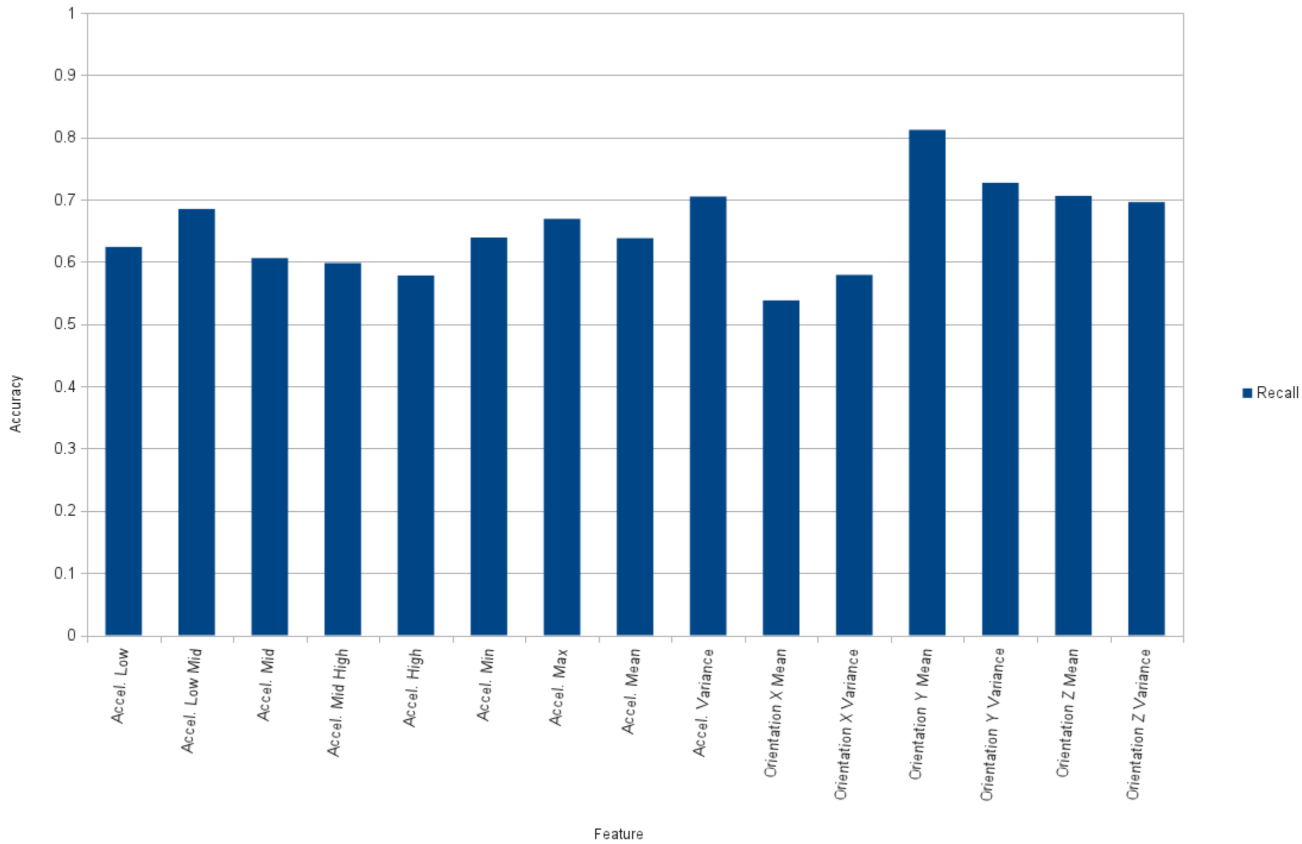


Figure 4.2: Recall with C4.5 classifiers and accelerometer & orientation features.

4.4.2 GPS location and satellite features

From figure 4.3 it is apparent that many of the GPS location features implemented are not actually informative; only the speed features and total distance (which is essentially a speed feature given the fixed segment durations) are. Not surprisingly the SNR mean is marginally more useful than the satellite count mean; they both essentially measure the same property (Radio Frequency (RF) signal attenuation) and the SNR mean just provides a more accurate measurement.

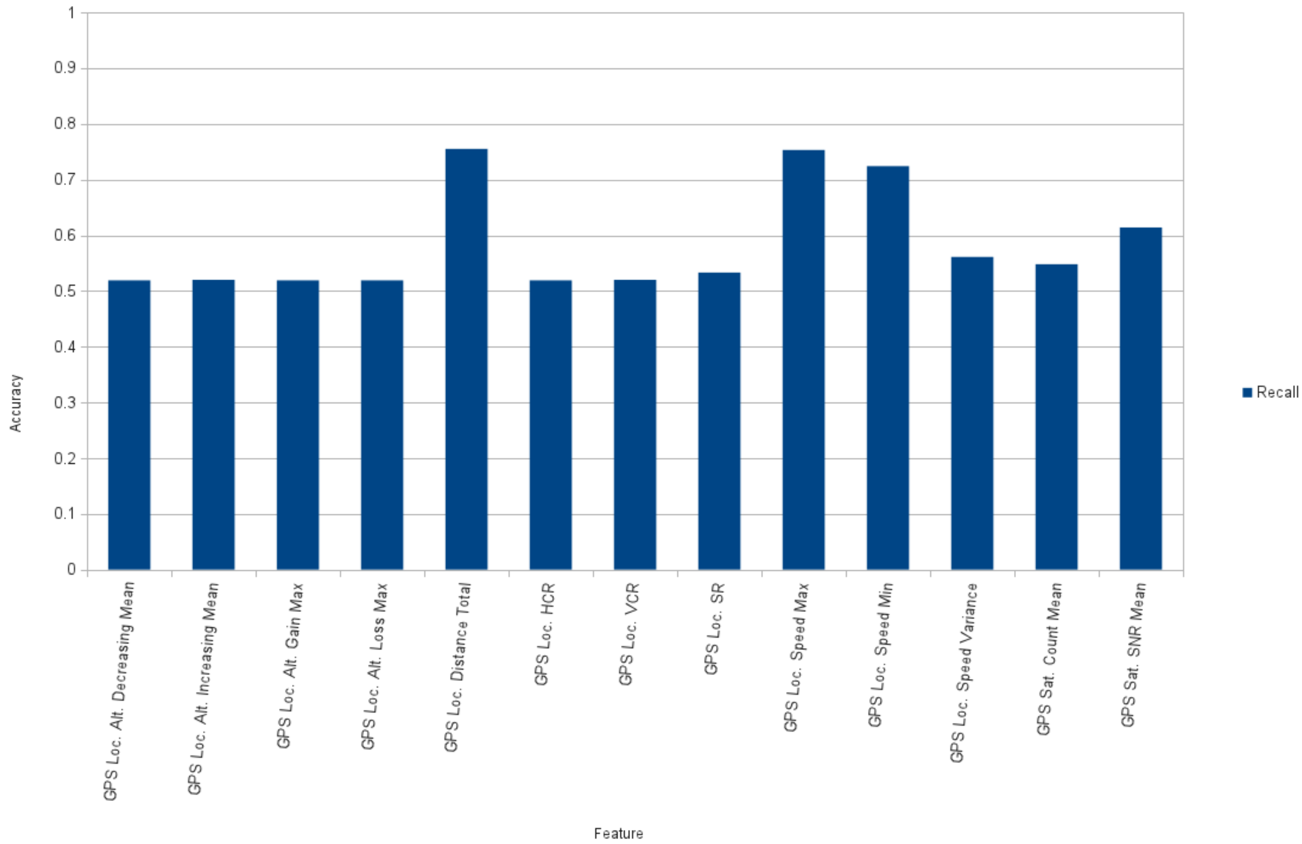


Figure 4.3: Recall with C4.5 classifiers and GPS location & satellite features.

4.4.3 Magnetic field features

Figure 4.4 shows us that the single most useful magnetic field strength feature is the mean but that all features provide information.

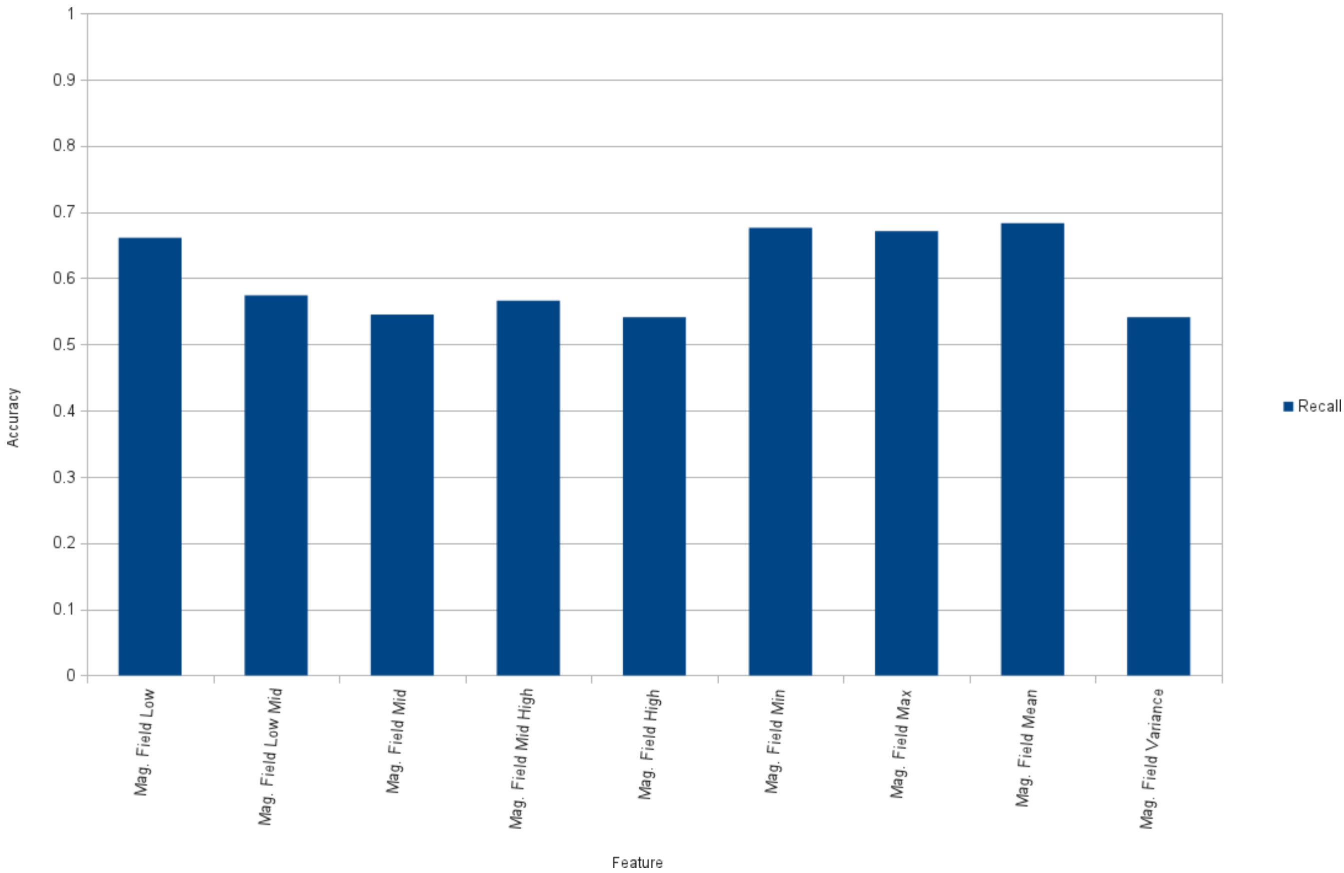


Figure 4.4: Recall with C4.5 classifiers and magnetic field features.

4.5 Remarks on Feature Selection for Specific Applications

For the energy metering and traffic routing applications described in the introduction chapter, the speed of the device would be required by the application and hence the GPS would be consuming power anyway, in which case my recommendation would be to use all of the sensor features I have described except for the light level features and the non-speed GPS location features (as they do not provide any information gain). The expected recall rates using a C4.5 classifier would be 0.978.

For more general context awareness where speed and/or distance data is not required, power savings may be made by avoiding the use of GPS features. My recommendation would be to use accelerometer, magnetic field strength and orientation data; with a C4.5 classifier the expected recall would be 0.968.

4.6 Evaluation Test Harness

Figure 4.5 shows the structure of the classification and evaluation areas of the Inference Framework.

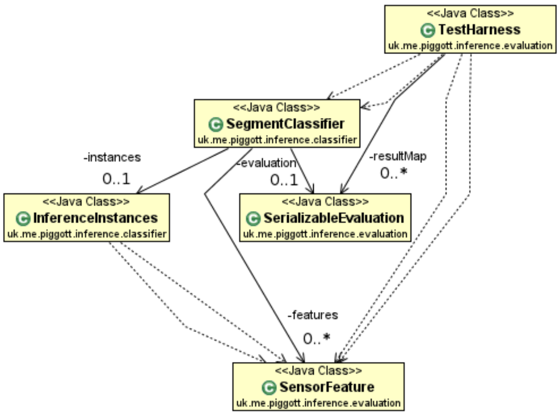


Figure 4.5: Inference Framework classifier and evaluation class diagram.

4.6.1 Classifier evaluation

The evaluation test harness creates a list of feature sets, the elements of which are of type SensorFeature[2]. This list of feature sets is then iterated through and for each feature set three segment classifiers are constructed, one using Weka's J48 classifier, one using Weka's NaiveBayes classifier and one using Weka's IBk classifier. These segment classifiers encapsulate a Weka classifier, a feature set, and a segment length (fixed at 10,000ms). A pool of threads equal in size to the number of available processors is then spawned. These threads remove segment classifiers from the list to be tested and terminate when none remain, doing for each:

1. Build the contained Weka classifier using the routes that have been selected for evaluation.

2. Perform 10-fold cross validation of the classifier.

3. Render a misclassification map[3] for the classifier, saving it to disk. This shows all correctly classified segments in green and all incorrectly classified segments in red.

4. Add the resulting Weka Evaluation object into a map from segment classifier names to evaluation objects for later sorting and viewing.

[2]This is just a wrapper that points to two enumeration values, the first being the sensor type and the second being the sensor features.

[3]Each test harness thread constructs a single LocationMap and reuses it for each segment classifier it tests.

4.6.2 Feature distributions and visualisation

The Inference Framework also provides a distribution command, which produces two images per sensor feature:

Feature histogram

A histogram showing the distributions of the feature value across transport classes is produced using Weka's AttributeVisualizationPanel class. This class does not appear to have been designed for external use, as I have not been able to get it to show class labels. Despite this, the produced distributions are quite insightful and helped with spotting problems with feature implementation code (similar distributions across all classes being suggestive of a problem). As an example, figure 4.6 shows the distribution variation for the `GPS_SATELLITES/MEAN_SNR` feature. Appendix C shows these distributions for all sensor and feature type combinations.

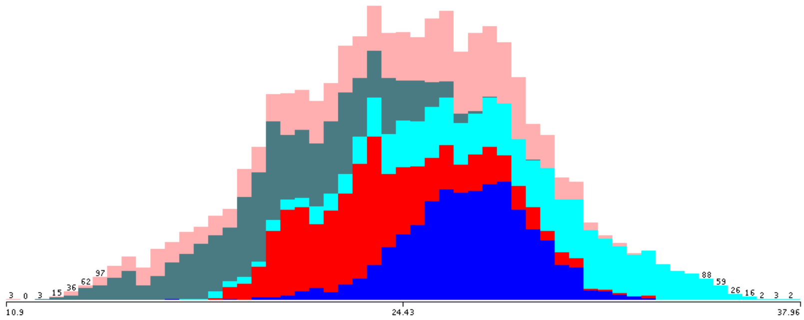


Figure 4.6: GPS satellite SNR mean class distributions. Minimum: 10.9, Maximum: 37.957, Mean: 24.445, StdDev: 4.249.

Feature value map

A map showing false-colour visualisation of the feature value on a per segment basis. Figure 4.7 shows the `GPS_SATELLITES/COUNT_MEAN` feature visualisation and figures 4.8, 4.9 and 4.10 show the `ORIENTATION/{X,Y,Z}_MEAN` feature visualisations. Appendix B shows these visualisations for all sensor and feature type combinations.

The value of these visualisations is in how readily they convey the variance of feature values; the goal being to maximise it. A visualisation predominantly filled with a single colour would be suggestive of a problem with the feature in question. With figure 4.7 the red portions indicate a maximum number of visible GPS satellites, while cooler colours indicate fewer visible satellites. Not surprisingly, the two long routes which clearly saw fewer GPS satellites are rail journeys.

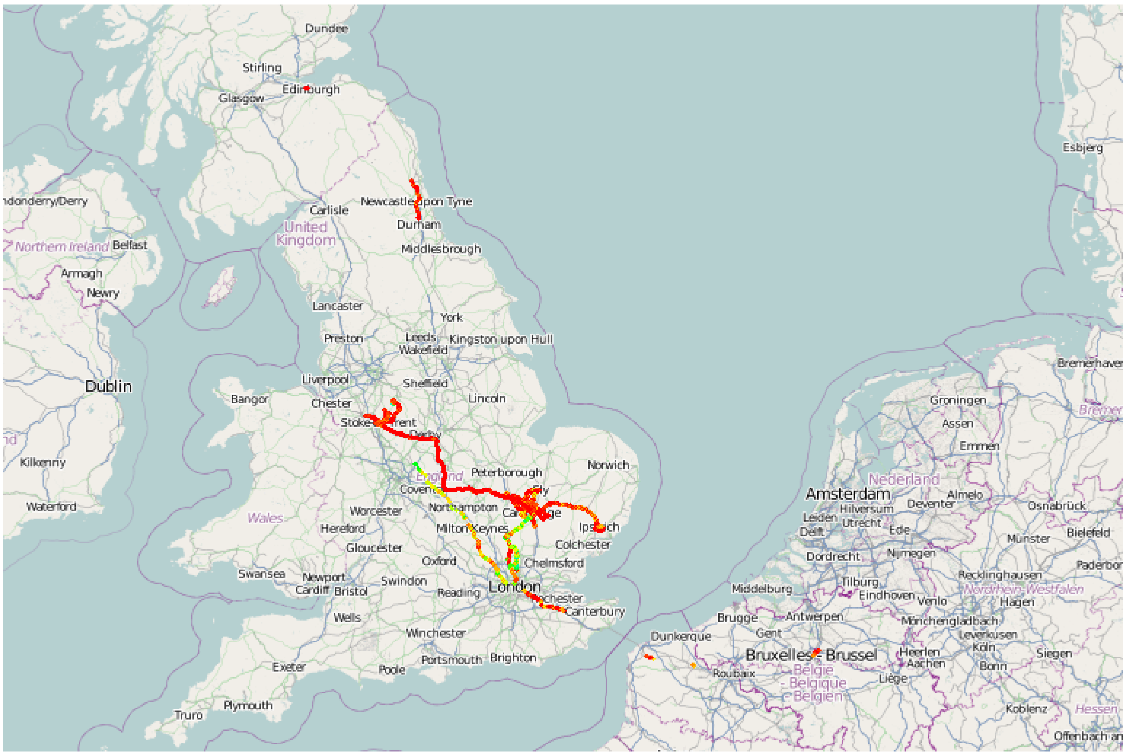


Figure 4.7: GPS satellite count mean visualisation.

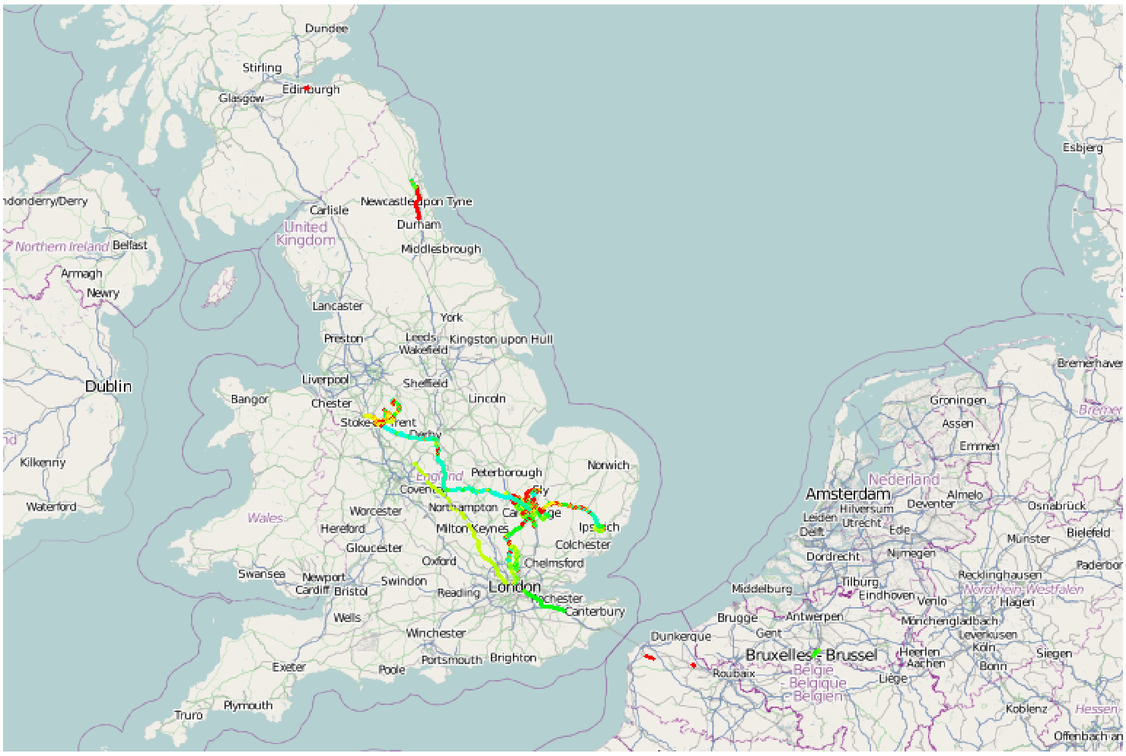


Figure 4.8: Orientation mean Azimuth visualisation.

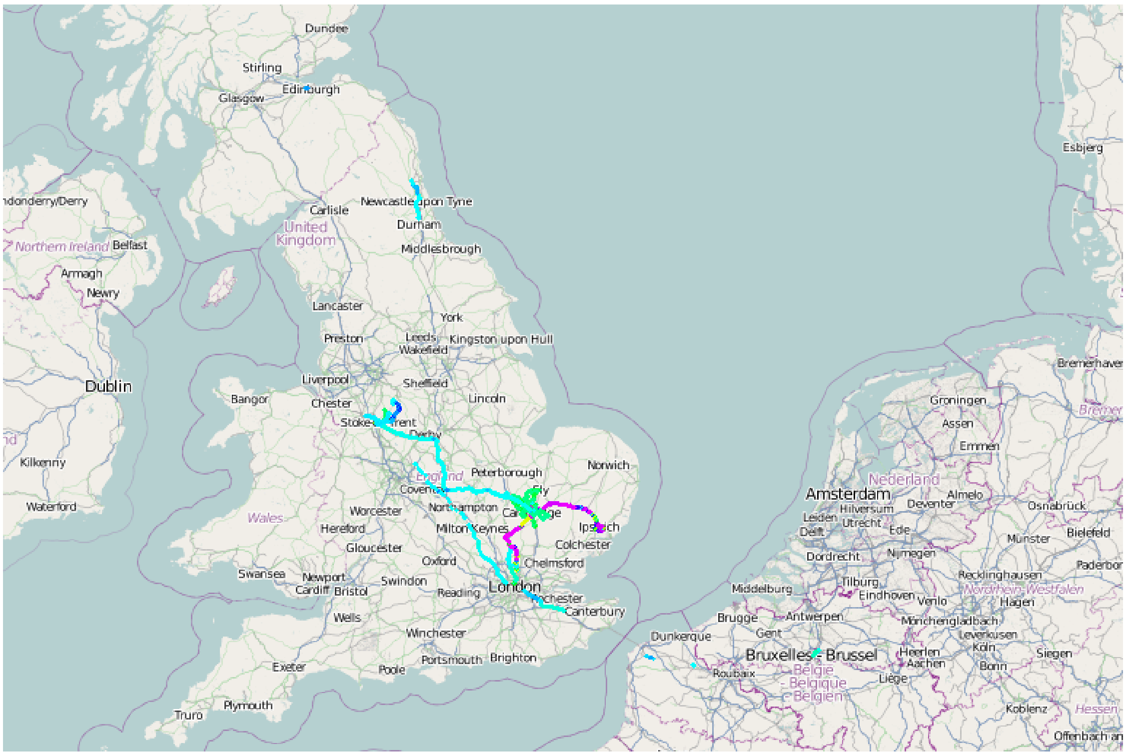


Figure 4.9: Orientation mean pitch visualisation.

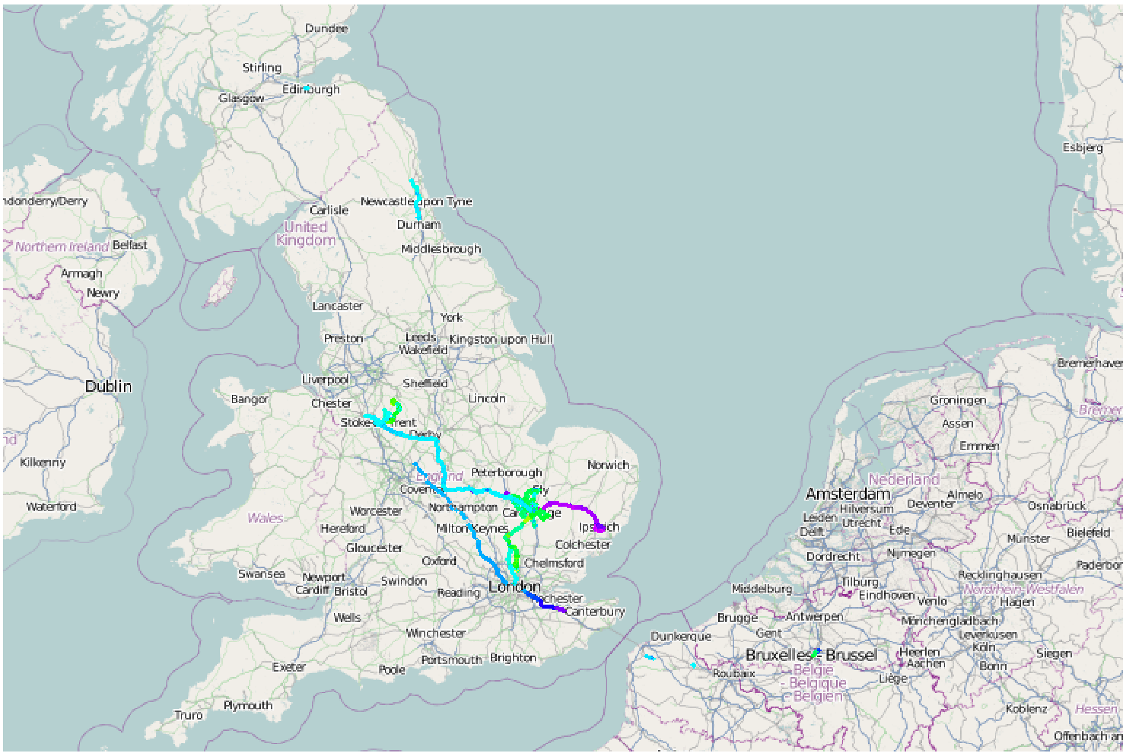


Figure 4.10: Orientation mean roll visualisation.

Consistent with the finding that the orientation X mean (mean Azimuth) is the weakest of the orientation features, there does not appear to be any pattern to figure 4.8. This is expected; even if there is a pattern to how users orient the phone relative to the vehicle they are in, the Azimuth mean will tend to zero for cyclic routes.

Orientation mean pitch and roll make more sense when analysed together. Suppose that there are four predominant Azimuth-agnostic positions that smartphones may be in:

- Flat on a surface with the screen pointing up.
- Upright, with the microphone end pointing down and the speaker end pointing up, as may be expected of a pocketed phone and a standing user.
- Partway-upright, as may be expected of a phone on the back of a cyclist or held in the hand for use.
- Sideways-upright, with one side pointing down and one side pointing up, as may be expected of a pocketed phone and a seated user.

Combining figures 4.9 and 4.10, there are four predominant combinations of pitch and roll. The assignments (blue pitch, blue roll = flat), (purple pitch, green roll = sideways-upright), (purple pitch, purple roll = upright) and (green pitch, green roll = partway-upright) are the most likely, given knowledge of the individual route transportation modes.

4.6.3 Misclassification plots

Having trained and evaluation each feature-set/classifier pair, the test harness generates a misclassification plot by running the trained classifier on each segment in the dataset. Figures 4.11 and 4.12 show the usefulness of these plots; the large majority of misclassifications in 4.12 are for car and bike journeys and are randomly distributed within those contexts. This reassures me that there are no degenerate cases to deal with.

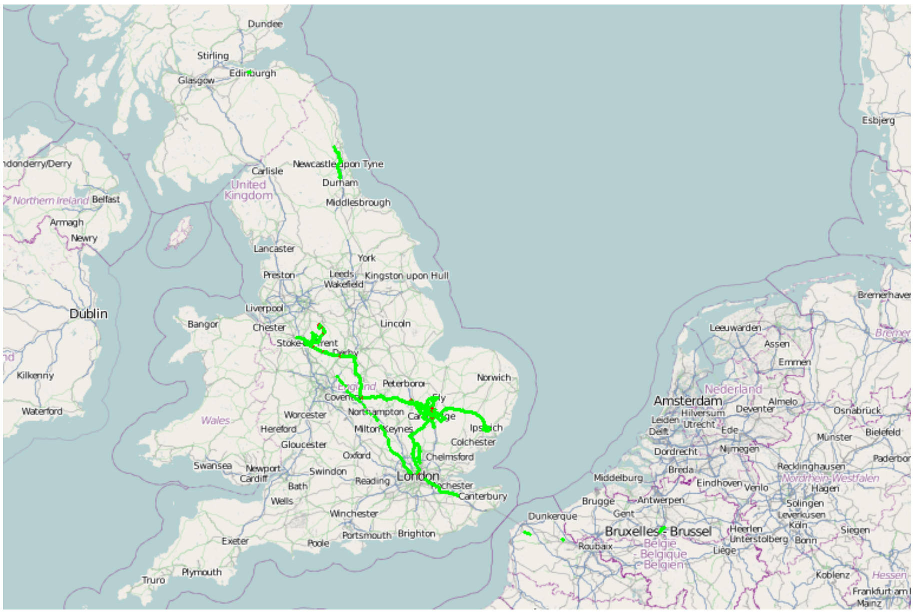


Figure 4.11: Misclassification plot of classifications with all features except light, with the C4.5 decision tree classifier. Green: correctly classified segment, Red: incorrectly classified segment.

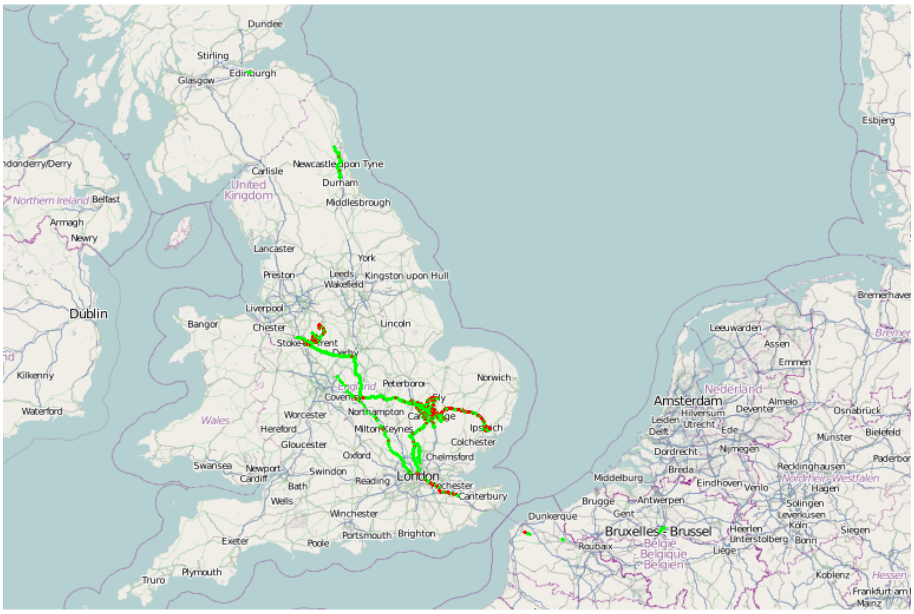


Figure 4.12: Misclassification plot of classifications with the GPS satellite mean count and mean SNR features and the C4.5 decision tree classifier. Green: correctly classified segment, Red: incorrectly classified segment.

4.7 Test Data

4.7.1 Data-set summary

Repeated for emphasis, table 4.1 gives a quantitative view on the data-set. To ensure all route traces had an associated ground truth I designed Route Tracer so that route tracing was activated by pressing the one of five transport class buttons matching the current vehicle type.

	time	distance
devices		
1	87h 25m 01s	2,773.953 km
2	10h 17m 34s	397.238 km
3	01h 47m 18s	146.667 km
volunteers		
volunteer 1	01h 47m 18s	146.667 km
volunteer 2	02h 19m 55s	9.495 km
volunteer 3	88h 07m 04s	2,809.96 km
volunteer 4	01h 35m 32s	89.378 km
volunteer 5	02h 39m 05s	210.646 km
volunteer 6	03h 00m 59s	51.712 km
transport classes		
bike	45h 22m 51s	981.988 km
bus	03h 28m 37s	79.403 km
car	27h 16m 55s	1,611.907 km
train	05h 52m 15s	578.631 km
walk	17h 29m 15s	65.929 km
totals	99h 29m 53s	3,317.858 km

Table 4.1: Summary of data-set.

4.7.2 Data-set resampling

There is some variation in how well different transportation modes are represented, for example, approximately half of the data-set by duration is cycling data, though by distance car data dominates. I made use of Weka's Resample class to produce random subsamples of the data-set, within which the class distributions are approximately uniform. This is important in order to ensure that the evaluation is not biased; without resampling even a null classifier could provide a true positive rate of around 0.5 just by using its knowledge of the prior distributions and classifying all instances as cycling.

Chapter 5

Conclusion

5.1 Result

All of the required success criterion set out in appendix D have been met. The extension activity of implementing a third classifier is covered by the creation of classifiers using non-GPS/accelerometer data.

An application for collecting sensor data with transportation mode labels was created and used to assemble a substantial data-set for training and testing classifiers with. A framework for generation and refinement of sensor features was created. A test harness was created and used to generate and evaluate a substantial number of segment classifiers, which infer the mode of transport of 10 second segments of routes where the mode of transport may be one of {bike, bus, car, train, walk}. Although not required by the success criterion it became apparent that mapping of the data-set would provide a number of useful insights and so an OSM based map system was created and used to visualise feature values and segment misclassifications.

As well as producing GPS and accelerometer feature based classifiers of comparable accuracy to those in the literature I found that orientation features produce very good results (recall 0.959 with a kNN classifier), while magnetic field data give respectable results with recall of 0.715 with a C4.5 Decision Tree. I also found that GPS inference could be significantly enhanced by using GPS satellite data (recall increases from 0.787 to 0.837 with a C4.5 Decision Tree). The insight is that GPS satellite data will indicate where location data are inaccurate, which firstly allows the classifier to place less confidence in the location data and secondly is information in itself.

5.2 Future Directions

5.2.1 Choice of parameters

When starting the project my intention was that it would primarily be an investigation of a relatively small number of fairly different algorithms, varying in terms of what instance classifiers they used, what segmentation methods they used and also what, if any, smoothing methods they used (e.g. DHMM), but being fairly similar in terms of what sensor data and thus what features they used.

The direction of the project has deviated slightly from the original plan, in that more classifiers have been constructed and evaluated and more sensor data types and more features have been used and generated for those classifiers. This is in part due to the fact that my own preliminary findings[1] indicated that - consistent with the existing work in the area - the choice of machine learning algorithm is actually not of much significance, typically making a difference of only a few percent at most in terms of True Positive (TP) rate (the Naive Bayes classifier being one exception). It is also in part due to limitations of the data-set which in turn were caused by limitations of what could be asked of volunteers; since sample routes contain traces of only one transportation mode it was not feasible to investigate smoothing methods, and some types of segmentation (e.g. walk vs. non walk) were also ruled out.

These two factors combined to push the project into primarily investigating which sensor types and derived features are most informative for segment classification. With more time I would investigate further the definitions of the sensor features used. As figures C.1 and C.2 show, some features do not appear to be providing much information to classifiers and may either be fundamentally flawed or require refinement.

5.2.2 Personal energy metering

The accuracy of the best resulting classifier (recall of 97.8%) is very promising and it would really be good to develop a standalone Android application for personal transportation energy metering. Better yet would be to automate the personal transportation energy metering component of the Energy Meter[2] application I created as part of my Undergraduate Research Opportunities Program (UROP).

5.3 Final Words

Despite a number of problems including the timing and power[3] issues in Android's sensor system and the growth of the data-set beyond a size that would fit in physical memory, I believe the project has been a great success. It has developed a number of classifiers greater in accuracy than many of those developed by related works[4].

In addition to testing approaches from existing literature, I have been able to confirm my original hypotheses that magnetic field strength and GPS satellite information are useful data sources. Finally, I found that orientation sensor information is a strong data source.

[1] I actually tested a much larger range of classifiers than documented, and found most to perform similarly. As a result of this finding I decided to focus my efforts on developing and testing new sensor features.

[2] `http://www.cl.cam.ac.uk/research/dtg/summer/`

[3] `http://code.google.com/p/android/issues/detail?id=3708`

[4] However, direct comparisons are not particularly meaningful due to the variation in the classes supported; classifiers that distinguish between a larger number of classes, particularly those which don't merge similar classes e.g. buses and cars may struggle to compete with those which don't.

Bibliography

[1] Reddy et al. Determining Transportation Mode On Mobile Phones. ISWC '08: IEEE International Symposium on Wearable Computers (2008) 7, 8, 14

[2] Zheng et al. Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web. WWW '08: Proceeding of the 17th international conference on World Wide Web (2008) 7, 8, 14, 36, 90, 91, 93

[3] Zheng et al. Understanding mobility based on GPS data. UbiComp '08: Proceedings of the 10th International Conference on Ubiquitous Computing (2008) 7, 8, 14, 36, 90, 91

[4] Manzoni et al. Transportation mode identication and real-time CO2 emission estimation using smartphones. http://senseable.mit.edu/co2go/ 7, 8, 16, 36

[5] Froehlich et al. UbiGreen: investigating a mobile tool for tracking and supporting green transportation habits. CHI '09: Proceedings of the 27th international conference on Human factors in computing systems (2009) 3

[6] Thiagarajan et al. Cooperative Transit Tracking using Smart-phones. SenSys '10: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (2010) 4

[7] Lu et al. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. SenSys '10: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (2010) 4

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. 7, 16

[9] J.R. Quinlan. C4. 5: programs for machine learning. Morgan Kaufmann, 1993. ISBN 1558602380. 16, 35

[10] Aha, D., and D. Kibler (1991) "Instance-based learning algorithms", Machine Learning, vol.6, pp. 37-66. 35

[11] George H. John and Pat Langley (1995). Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345. Morgan Kaufmann, San Mateo. 35

[12] M. B. Kjergaard et al. Entracked: Energy-Efficient Robust Position Tracking for Mobile Devices. MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services (2009) 37

Appendix A

Classifier Results

A.1 Single Sensor Classifiers

A.1.1 Accelerometer Classifiers

MAGNITUDE_HIGH_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.867	0.587	0.614	0.867	0.719	0.676	BIKE
	0	0	0	0	0	0.65	BUS
	0.235	0.118	0.39	0.235	0.293	0.596	CAR
	0.001	0	0.154	0.001	0.003	0.753	TRAIN
	0.477	0.058	0.585	0.477	0.526	0.797	WALK
Weighted Avg.	0.578	0.342	0.508	0.578	0.522	0.677	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.964	0.832	0.555	0.964	0.705	0.691	BIKE
	0	0	0	0	0	0.678	BUS
	0.004	0.001	0.474	0.004	0.008	0.548	CAR
	0	0	0	0	0	0.788	TRAIN
	0.416	0.043	0.627	0.416	0.5	0.802	WALK
Weighted Avg.	0.562	0.439	0.496	0.562	0.441	0.677	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.594	0.435	0.595	0.594	0.595	0.579	BIKE
	0.062	0.037	0.062	0.062	0.062	0.512	BUS
	0.281	0.233	0.279	0.281	0.28	0.526	CAR
	0.101	0.05	0.103	0.101	0.102	0.524	TRAIN
	0.406	0.103	0.404	0.406	0.405	0.651	WALK
Weighted Avg.	0.444	0.302	0.444	0.444	0.444	0.571	

MAGNITUDE_LOW_MID_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.817	0.256	0.775	0.817	0.796	0.838	BIKE
	0.003	0.001	0.13	0.003	0.006	0.835	BUS
	0.761	0.163	0.601	0.761	0.671	0.864	CAR
	0.281	0.035	0.308	0.281	0.294	0.871	TRAIN
	0.414	0.04	0.64	0.414	0.503	0.758	WALK
Weighted Avg.	0.685	0.18	0.664	0.685	0.666	0.834	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.804	0.216	0.801	0.804	0.803	0.835	BIKE

	0	0	0	0	0	0.826	BUS
	0.836	0.298	0.475	0.836	0.605	0.872	CAR
	0	0	0	0	0	0.864	TRAIN
	0.284	0.01	0.83	0.284	0.423	0.703	WALK
Weighted Avg.	0.663	0.186	0.653	0.663	0.626	0.826	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.723	0.297	0.724	0.723	0.724	0.712	BIKE
	0.107	0.035	0.105	0.107	0.106	0.534	BUS
	0.525	0.152	0.527	0.525	0.526	0.684	CAR
	0.185	0.045	0.188	0.185	0.187	0.569	TRAIN
	0.406	0.104	0.402	0.406	0.404	0.651	WALK
Weighted Avg.	0.577	0.21	0.577	0.577	0.577	0.682	

MAGNITUDE_LOW_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.857	0.482	0.657	0.857	0.744	0.743	BIKE
	0	0	0	0	0	0.813	BUS
	0.326	0.12	0.465	0.326	0.383	0.666	CAR
	0.002	0	0.214	0.002	0.004	0.827	TRAIN
	0.68	0.062	0.654	0.68	0.667	0.855	WALK
Weighted Avg.	0.624	0.288	0.562	0.624	0.578	0.748	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.962	0.83	0.556	0.962	0.704	0.744	BIKE
	0	0	0	0	0	0.832	BUS
	0	0	0	0	0	0.711	CAR
	0	0	0	0	0	0.837	TRAIN
	0.507	0.032	0.732	0.507	0.599	0.859	WALK
Weighted Avg.	0.574	0.435	0.396	0.574	0.454	0.761	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.628	0.398	0.63	0.628	0.629	0.615	BIKE
	0.1	0.035	0.101	0.1	0.1	0.534	BUS
	0.33	0.219	0.327	0.33	0.329	0.555	CAR
	0.134	0.048	0.135	0.134	0.135	0.539	TRAIN
	0.524	0.081	0.525	0.524	0.524	0.722	WALK
Weighted Avg.	0.494	0.276	0.494	0.494	0.494	0.609	

MAGNITUDE_MAXIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.837	0.326	0.735	0.837	0.783	0.808	BIKE
	0	0	0	0	0	0.815	BUS
	0.632	0.179	0.533	0.632	0.578	0.784	CAR
	0.315	0.012	0.588	0.315	0.41	0.873	TRAIN
	0.431	0.033	0.695	0.431	0.532	0.808	WALK
Weighted Avg.	0.669	0.218	0.645	0.669	0.647	0.806	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.812	0.31	0.739	0.812	0.774	0.792	BIKE
	0	0	0	0	0	0.82	BUS
	0.636	0.26	0.44	0.636	0.52	0.792	CAR
	0	0	0	0	0	0.875	TRAIN

	0.393	0.023	0.749	0.393	0.516	0.752	WALK
Weighted Avg.	0.634	0.228	0.601	0.634	0.604	0.792	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.686	0.344	0.683	0.686	0.685	0.675	BIKE
	0.107	0.034	0.11	0.107	0.108	0.532	BUS
	0.41	0.185	0.416	0.41	0.413	0.617	CAR
	0.303	0.04	0.297	0.303	0.3	0.624	TRAIN
	0.463	0.094	0.46	0.463	0.461	0.689	WALK
Weighted Avg.	0.544	0.241	0.543	0.544	0.544	0.655	

MAGNITUDE_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.941	0.626	0.619	0.941	0.746	0.669	BIKE
	0	0	0	0	0	0.696	BUS
	0.235	0.05	0.601	0.235	0.338	0.678	CAR
	0.31	0.011	0.61	0.31	0.411	0.788	TRAIN
	0.518	0.014	0.867	0.518	0.649	0.803	WALK
Weighted Avg.	0.638	0.34	0.627	0.638	0.587	0.698	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.929	0.692	0.592	0.929	0.723	0.709	BIKE
	0	0	0	0	0	0.832	BUS
	0.099	0.082	0.281	0.099	0.146	0.602	CAR
	0.119	0.015	0.311	0.119	0.172	0.688	TRAIN
	0.489	0.008	0.912	0.489	0.637	0.692	WALK
Weighted Avg.	0.585	0.381	0.526	0.585	0.514	0.684	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.63	0.406	0.626	0.63	0.628	0.612	BIKE
	0.1	0.035	0.101	0.1	0.1	0.535	BUS
	0.348	0.208	0.35	0.348	0.349	0.571	CAR
	0.281	0.039	0.287	0.281	0.284	0.622	TRAIN
	0.537	0.079	0.538	0.537	0.537	0.726	WALK
Weighted Avg.	0.509	0.276	0.508	0.509	0.509	0.616	

MAGNITUDE_MID_HIGH_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.87	0.546	0.632	0.87	0.732	0.733	BIKE
	0	0	0	0	0	0.662	BUS
	0.277	0.126	0.415	0.277	0.332	0.645	CAR
	0.002	0	0.25	0.002	0.004	0.78	TRAIN
	0.539	0.051	0.645	0.539	0.587	0.833	WALK
Weighted Avg.	0.598	0.322	0.537	0.598	0.547	0.726	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.957	0.759	0.576	0.957	0.719	0.742	BIKE
	0	0	0	0	0	0.611	BUS
	0.069	0.022	0.501	0.069	0.122	0.577	CAR
	0	0	0	0	0	0.785	TRAIN
	0.48	0.04	0.674	0.48	0.561	0.817	WALK
Weighted Avg.	0.584	0.405	0.52	0.584	0.485	0.71	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.621	0.414	0.618	0.621	0.619	0.603	BIKE
	0.04	0.037	0.04	0.04	0.04	0.501	BUS
	0.304	0.223	0.305	0.304	0.305	0.539	CAR
	0.145	0.048	0.146	0.145	0.145	0.551	TRAIN
	0.449	0.093	0.453	0.449	0.451	0.68	WALK
Weighted Avg.	0.472	0.287	0.471	0.472	0.471	0.592	

MAGNITUDE_MID_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.852	0.522	0.638	0.852	0.73	0.709	BIKE
	0	0	0	0	0	0.734	BUS
	0.311	0.123	0.449	0.311	0.368	0.641	CAR
	0.006	0.001	0.178	0.006	0.011	0.83	TRAIN
	0.593	0.057	0.643	0.593	0.617	0.825	WALK
Weighted Avg.	0.606	0.31	0.544	0.606	0.559	0.717	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.96	0.856	0.547	0.96	0.697	0.706	BIKE
	0	0	0	0	0	0.771	BUS
	0	0	0	0	0	0.601	CAR
	0	0	0	0	0	0.848	TRAIN
	0.46	0.026	0.752	0.46	0.571	0.807	WALK
Weighted Avg.	0.566	0.448	0.395	0.566	0.446	0.705	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.602	0.433	0.6	0.602	0.601	0.584	BIKE
	0.092	0.036	0.09	0.092	0.091	0.526	BUS
	0.292	0.224	0.295	0.292	0.294	0.535	CAR
	0.145	0.046	0.15	0.145	0.147	0.547	TRAIN
	0.485	0.091	0.478	0.485	0.481	0.698	WALK
Weighted Avg.	0.466	0.297	0.465	0.466	0.465	0.585	

MAGNITUDE_MINIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.798	0.358	0.706	0.798	0.749	0.784	BIKE
	0	0	0	0	0	0.796	BUS
	0.638	0.186	0.524	0.638	0.575	0.788	CAR
	0.001	0	0.143	0.001	0.001	0.795	TRAIN
	0.473	0.055	0.595	0.473	0.527	0.799	WALK
Weighted Avg.	0.639	0.24	0.589	0.639	0.606	0.788	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.819	0.406	0.685	0.819	0.746	0.753	BIKE
	0	0	0	0	0	0.794	BUS
	0.605	0.261	0.428	0.605	0.501	0.74	CAR
	0	0	0	0	0	0.771	TRAIN
	0.171	0.012	0.715	0.171	0.277	0.752	WALK
Weighted Avg.	0.598	0.276	0.565	0.598	0.55	0.752	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.66	0.373	0.656	0.66	0.658	0.649	BIKE
	0.108	0.036	0.104	0.108	0.106	0.542	BUS

	0.439	0.177	0.443	0.439	0.441	0.637	CAR
	0.163	0.048	0.158	0.163	0.16	0.565	TRAIN
	0.387	0.102	0.395	0.387	0.391	0.646	WALK
Weighted Avg.	0.519	0.256	0.519	0.519	0.519	0.637	

MAGNITUDE_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.823	0.24	0.787	0.823	0.805	0.853	BIKE
	0.002	0	0.167	0.002	0.004	0.818	BUS
	0.708	0.165	0.58	0.708	0.638	0.83	CAR
	0.276	0.031	0.332	0.276	0.301	0.863	TRAIN
	0.616	0.029	0.783	0.616	0.689	0.873	WALK
Weighted Avg.	0.705	0.171	0.689	0.705	0.69	0.85	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.629	0.083	0.891	0.629	0.737	0.859	BIKE
	0	0	0	0	0	0.786	BUS
	0.92	0.405	0.422	0.92	0.579	0.813	CAR
	0	0	0	0	0	0.851	TRAIN
	0.582	0.021	0.829	0.582	0.684	0.779	WALK
Weighted Avg.	0.636	0.145	0.687	0.636	0.624	0.833	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.734	0.292	0.731	0.734	0.732	0.72	BIKE
	0.113	0.035	0.111	0.113	0.112	0.542	BUS
	0.475	0.164	0.482	0.475	0.478	0.653	CAR
	0.196	0.046	0.194	0.196	0.195	0.574	TRAIN
	0.573	0.074	0.571	0.573	0.572	0.748	WALK
Weighted Avg.	0.595	0.206	0.595	0.595	0.595	0.693	

All accelerometer features

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.938	0.093	0.916	0.938	0.927	0.938	BIKE
	0.62	0.011	0.681	0.62	0.649	0.816	BUS
	0.868	0.042	0.868	0.868	0.868	0.92	CAR
	0.727	0.015	0.726	0.727	0.726	0.863	TRAIN
	0.777	0.028	0.828	0.777	0.802	0.892	WALK
Weighted Avg.	0.874	0.064	0.873	0.874	0.873	0.918	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.708	0.078	0.907	0.708	0.795	0.854	BIKE
	0.521	0.082	0.198	0.521	0.287	0.888	BUS
	0.445	0.077	0.65	0.445	0.528	0.846	CAR
	0.697	0.178	0.18	0.697	0.286	0.879	TRAIN
	0.592	0.043	0.701	0.592	0.642	0.754	WALK
Weighted Avg.	0.619	0.078	0.749	0.619	0.662	0.84	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.931	0.087	0.921	0.931	0.926	0.922	BIKE
	0.579	0.011	0.679	0.579	0.625	0.782	BUS
	0.874	0.047	0.856	0.874	0.865	0.912	CAR
	0.736	0.014	0.746	0.736	0.741	0.863	TRAIN
	0.789	0.03	0.817	0.789	0.803	0.881	WALK

Weighted Avg.	0.873	0.062	0.871	0.873	0.872	0.905	

A.1.2 GPS Location Classifiers

ALTITUDE_DECREASING_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.997	0.992	0.52	0.997	0.684	0.509	BIKE
	0	0	0	0	0	0.492	BUS
	0.006	0.004	0.36	0.006	0.012	0.502	CAR
	0.008	0.001	0.279	0.008	0.016	0.524	TRAIN
	0	0	0	0	0	0.507	WALK
Weighted Avg.	0.519	0.516	0.373	0.519	0.359	0.507	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.996	0.988	0.521	0.996	0.684	0.521	BIKE
	0	0	0	0	0	0.545	BUS
	0	0	0	0	0	0.521	CAR
	0.024	0.003	0.283	0.024	0.044	0.524	TRAIN
	0.007	0.003	0.277	0.007	0.013	0.488	WALK
Weighted Avg.	0.519	0.513	0.326	0.519	0.359	0.517	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.994	0.985	0.521	0.994	0.684	0.519	BIKE
	0	0	0	0	0	0.541	BUS
	0.008	0.005	0.315	0.008	0.015	0.519	CAR
	0.008	0.001	0.297	0.008	0.015	0.507	TRAIN
	0.005	0.003	0.25	0.005	0.01	0.524	WALK
Weighted Avg.	0.519	0.513	0.4	0.519	0.361	0.52	

ALTITUDE_GAIN_MAXIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.996	0.986	0.522	0.996	0.685	0.512	BIKE
	0	0	0	0	0	0.503	BUS
	0	0	0	0	0	0.499	CAR
	0.013	0.002	0.237	0.013	0.024	0.527	TRAIN
	0.01	0.005	0.252	0.01	0.019	0.508	WALK
Weighted Avg.	0.519	0.513	0.32	0.519	0.359	0.509	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.988	0.963	0.525	0.988	0.686	0.512	BIKE
	0	0	0	0	0	0.491	BUS
	0	0	0	0	0	0.499	CAR
	0.023	0.004	0.246	0.023	0.042	0.53	TRAIN
	0.031	0.017	0.232	0.031	0.054	0.506	WALK
Weighted Avg.	0.518	0.502	0.32	0.518	0.366	0.508	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.996	0.987	0.521	0.996	0.684	0.513	BIKE
	0	0	0	0	0	0.539	BUS
	0.001	0.001	0.129	0.001	0.001	0.515	CAR
	0.004	0.001	0.208	0.004	0.007	0.52	TRAIN
	0.011	0.005	0.28	0.011	0.022	0.524	WALK

Weighted Avg.	0.519	0.513	0.354	0.519	0.359	0.516	

ALTITUDE_INCREASING_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.994	0.978	0.523	0.994	0.685	0.508	BIKE
	0	0	0	0	0	0.5	BUS
	0.002	0.001	0.333	0.002	0.004	0.502	CAR
	0.041	0.005	0.328	0.041	0.073	0.524	TRAIN
	0.013	0.004	0.333	0.013	0.025	0.508	WALK
Weighted Avg.	0.52	0.509	0.419	0.52	0.364	0.507	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.997	0.987	0.521	0.997	0.685	0.509	BIKE
	0	0	0	0	0	0.532	BUS
	0	0	0	0	0	0.523	CAR
	0.026	0.004	0.253	0.026	0.047	0.527	TRAIN
	0.005	0.002	0.317	0.005	0.01	0.493	WALK
Weighted Avg.	0.519	0.513	0.331	0.519	0.359	0.511	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.992	0.975	0.524	0.992	0.685	0.51	BIKE
	0	0	0	0	0	0.53	BUS
	0.006	0.006	0.258	0.006	0.012	0.514	CAR
	0.042	0.005	0.324	0.042	0.075	0.513	TRAIN
	0.009	0.003	0.375	0.009	0.018	0.525	WALK
Weighted Avg.	0.52	0.508	0.407	0.52	0.365	0.514	

ALTITUDE_LOSS_MAXIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.999	0.999	0.519	0.999	0.683	0.502	BIKE
	0	0	0	0	0	0.497	BUS
	0	0	0	0	0	0.499	CAR
	0	0	0	0	0	0.507	TRAIN
	0.001	0.001	0.133	0.001	0.002	0.502	WALK
Weighted Avg.	0.519	0.519	0.289	0.519	0.355	0.502	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.997	0.99	0.521	0.997	0.684	0.505	BIKE
	0	0	0	0	0	0.5	BUS
	0	0	0	0	0	0.5	CAR
	0.03	0.005	0.261	0.03	0.054	0.513	TRAIN
	0.001	0	0.25	0.001	0.002	0.502	WALK
Weighted Avg.	0.519	0.514	0.321	0.519	0.358	0.504	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.997	0.992	0.52	0.997	0.684	0.518	BIKE
	0	0	0	0	0	0.546	BUS
	0.001	0.001	0.171	0.001	0.002	0.515	CAR
	0.007	0.001	0.323	0.007	0.014	0.512	TRAIN
	0.004	0.003	0.221	0.004	0.008	0.522	WALK
Weighted Avg.	0.519	0.516	0.361	0.519	0.357	0.519	

DISTANCE_TOTAL

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.898	0.258	0.79	0.898	0.84	0.879	BIKE
	0.007	0	0.389	0.007	0.014	0.641	BUS
	0.644	0.06	0.774	0.644	0.703	0.835	CAR
	0.235	0.001	0.923	0.235	0.375	0.815	TRAIN
	0.81	0.087	0.615	0.81	0.699	0.939	WALK
Weighted Avg.	0.755	0.161	0.753	0.755	0.731	0.865	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.805	0.217	0.8	0.805	0.802	0.839	BIKE
	0	0	0	0	0	0.567	BUS
	0.549	0.048	0.786	0.549	0.646	0.765	CAR
	0.121	0	0.966	0.121	0.215	0.801	TRAIN
	0.946	0.19	0.461	0.946	0.62	0.855	WALK
Weighted Avg.	0.697	0.152	0.726	0.697	0.676	0.811	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.774	0.202	0.805	0.774	0.789	0.795	BIKE
	0.097	0.035	0.098	0.097	0.098	0.52	BUS
	0.577	0.124	0.6	0.577	0.589	0.734	CAR
	0.327	0.037	0.33	0.327	0.329	0.654	TRAIN
	0.732	0.082	0.607	0.732	0.664	0.832	WALK
Weighted Avg.	0.671	0.15	0.674	0.671	0.672	0.768	

HEADING_CHANGE_RATE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.518	BIKE
	0	0	0	0	0	0.538	BUS
	0	0	0	0	0	0.523	CAR
	0	0	0	0	0	0.523	TRAIN
	0	0	0	0	0	0.555	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.526	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.997	0.987	0.522	0.997	0.685	0.549	BIKE
	0	0	0	0	0	0.615	BUS
	0.004	0.003	0.286	0.004	0.007	0.536	CAR
	0.005	0	0.368	0.005	0.01	0.534	TRAIN
	0.01	0.003	0.392	0.01	0.019	0.614	WALK
Weighted Avg.	0.52	0.513	0.417	0.52	0.361	0.557	

SPEED_MAXIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.968	0.396	0.725	0.968	0.829	0.869	BIKE

	0	0	0	0	0	0.548	BUS
	0.644	0.049	0.808	0.644	0.717	0.819	CAR
	0.232	0	0.979	0.232	0.375	0.818	TRAIN
	0.556	0.022	0.814	0.556	0.66	0.946	WALK
Weighted Avg.	0.753	0.221	0.745	0.753	0.722	0.853	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.988	0.824	0.564	0.988	0.718	0.556	BIKE
	0	0	0	0	0	0.539	BUS
	0.235	0.03	0.716	0.235	0.354	0.774	CAR
	0.001	0.011	0.007	0.001	0.002	0.706	TRAIN
	0	0	0	0	0	0.871	WALK
Weighted Avg.	0.57	0.436	0.468	0.57	0.459	0.663	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.966	0.394	0.725	0.966	0.829	0.896	BIKE
	0	0	0	0	0	0.724	BUS
	0.646	0.051	0.803	0.646	0.716	0.855	CAR
	0.234	0	0.979	0.234	0.378	0.837	TRAIN
	0.554	0.022	0.814	0.554	0.659	0.952	WALK
Weighted Avg.	0.753	0.22	0.744	0.753	0.721	0.885	

SPEED_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.949	0.411	0.714	0.949	0.815	0.852	BIKE
	0	0	0	0	0	0.61	BUS
	0.604	0.053	0.786	0.604	0.683	0.797	CAR
	0.214	0.003	0.804	0.214	0.338	0.788	TRAIN
	0.535	0.036	0.721	0.535	0.614	0.936	WALK
Weighted Avg.	0.729	0.231	0.711	0.729	0.697	0.839	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.988	0.673	0.613	0.988	0.757	0.845	BIKE
	0	0	0	0	0	0.553	BUS
	0.503	0.045	0.783	0.503	0.613	0.767	CAR
	0.097	0.002	0.734	0.097	0.172	0.752	TRAIN
	0	0	0	0	0	0.856	WALK
Weighted Avg.	0.641	0.36	0.548	0.641	0.551	0.812	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.943	0.415	0.71	0.943	0.81	0.869	BIKE
	0.005	0.003	0.063	0.005	0.009	0.617	BUS
	0.572	0.058	0.759	0.572	0.653	0.801	CAR
	0.201	0.009	0.551	0.201	0.294	0.701	TRAIN
	0.523	0.034	0.729	0.523	0.609	0.937	WALK
Weighted Avg.	0.716	0.235	0.692	0.716	0.685	0.844	

SPEED_MINIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.955	0.445	0.698	0.955	0.807	0.867	BIKE
	0	0	0	0	0	0.567	BUS
	0.588	0.042	0.82	0.588	0.685	0.814	CAR
	0.204	0	0.973	0.204	0.337	0.807	TRAIN

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.507	0.035	0.713	0.507	0.592	0.92	WALK
Weighted Avg.	0.724	0.246	0.718	0.724	0.69	0.847	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.782	0.188	0.818	0.782	0.8	0.838	BIKE
	0	0	0	0	0	0.564	BUS
	0.541	0.037	0.824	0.541	0.653	0.741	CAR
	0.121	0	0.994	0.121	0.216	0.646	TRAIN
	0.942	0.233	0.411	0.942	0.572	0.858	WALK
Weighted Avg.	0.683	0.141	0.738	0.683	0.67	0.797	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.955	0.446	0.698	0.955	0.806	0.884	BIKE
	0	0	0	0	0	0.692	BUS
	0.588	0.042	0.819	0.588	0.685	0.835	CAR
	0.203	0	0.976	0.203	0.337	0.826	TRAIN
	0.506	0.035	0.713	0.506	0.592	0.924	WALK
Weighted Avg.	0.724	0.247	0.718	0.724	0.69	0.868	

SPEED_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.968	0.839	0.554	0.968	0.705	0.652	BIKE
	0	0	0	0	0	0.7	BUS
	0.211	0.042	0.62	0.211	0.315	0.73	CAR
	0.098	0.003	0.644	0.098	0.17	0.684	TRAIN
	0.014	0.001	0.632	0.014	0.027	0.803	WALK
Weighted Avg.	0.561	0.446	0.566	0.561	0.456	0.697	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.991	0.994	0.518	0.991	0.681	0.502	BIKE
	0	0	0	0	0	0.462	BUS
	0	0	0	0	0	0.487	CAR
	0	0	0	0	0	0.476	TRAIN
	0.007	0.007	0.141	0.007	0.014	0.514	WALK
Weighted Avg.	0.516	0.517	0.29	0.516	0.355	0.497	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.867	0.444	0.678	0.867	0.761	0.813	BIKE
	0.088	0.028	0.108	0.088	0.097	0.573	BUS
	0.527	0.101	0.627	0.527	0.573	0.754	CAR
	0.18	0.025	0.292	0.18	0.223	0.682	TRAIN
	0.392	0.013	0.838	0.392	0.534	0.911	WALK
Weighted Avg.	0.649	0.259	0.647	0.649	0.628	0.797	

STOP_RATE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.96	0.856	0.547	0.96	0.697	0.59	BIKE
	0	0	0	0	0	0.541	BUS
	0	0	0	0	0	0.554	CAR
	0	0	0	0	0	0.531	TRAIN
	0.238	0.065	0.387	0.238	0.294	0.724	WALK
Weighted Avg.	0.533	0.454	0.341	0.533	0.405	0.596	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.836	0.743	0.548	0.836	0.662	0.577	BIKE
	0	0	0	0	0	0.524	BUS
	0	0	0	0	0	0.555	CAR
	0	0	0	0	0	0.522	TRAIN
	0.486	0.161	0.342	0.486	0.401	0.717	WALK
Weighted Avg.	0.505	0.409	0.335	0.505	0.403	0.587	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.96	0.857	0.547	0.96	0.697	0.589	BIKE
	0	0	0	0	0	0.55	BUS
	0	0.001	0.125	0	0.001	0.551	CAR
	0	0	0	0	0	0.525	TRAIN
	0.233	0.064	0.385	0.233	0.29	0.723	WALK
Weighted Avg.	0.532	0.454	0.371	0.532	0.405	0.595	

VELOCITY_CHANGE_RATE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.998	0.992	0.52	0.998	0.684	0.51	BIKE
	0	0	0	0	0	0.507	BUS
	0.003	0.002	0.38	0.003	0.006	0.503	CAR
	0.023	0.002	0.402	0.023	0.044	0.544	TRAIN
	0	0	0	0	0	0.508	WALK
Weighted Avg.	0.52	0.515	0.384	0.52	0.359	0.51	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.977	0.954	0.525	0.977	0.683	0.51	BIKE
	0	0	0	0	0	0.503	BUS
	0.022	0.018	0.288	0.022	0.041	0.5	CAR
	0.068	0.013	0.234	0.068	0.106	0.551	TRAIN
	0	0	0	0	0	0.509	WALK
Weighted Avg.	0.516	0.5	0.355	0.516	0.37	0.509	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.998	0.993	0.52	0.998	0.684	0.51	BIKE
	0	0	0	0	0	0.514	BUS
	0.004	0.001	0.481	0.004	0.008	0.502	CAR
	0.023	0.001	0.478	0.023	0.043	0.545	TRAIN
	0	0	0	0	0	0.507	WALK
Weighted Avg.	0.52	0.516	0.413	0.52	0.359	0.509	

All GPS location features

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.892	0.188	0.837	0.892	0.864	0.908	BIKE
	0.068	0.015	0.148	0.068	0.093	0.645	BUS
	0.722	0.075	0.756	0.722	0.738	0.863	CAR
	0.451	0.007	0.784	0.451	0.573	0.839	TRAIN
	0.83	0.052	0.733	0.83	0.778	0.947	WALK
Weighted Avg.	0.787	0.124	0.773	0.787	0.776	0.889	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.781	0.206	0.804	0.781	0.792	0.844	BIKE
	0.024	0.007	0.118	0.024	0.04	0.638	BUS

	0.561	0.054	0.771	0.561	0.65	0.764	CAR
	0.155	0.011	0.434	0.155	0.228	0.795	TRAIN
	0.876	0.191	0.441	0.876	0.586	0.86	WALK
Weighted Avg.	0.68	0.149	0.697	0.68	0.669	0.816	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.829	0.16	0.848	0.829	0.839	0.844	BIKE
	0.14	0.033	0.141	0.14	0.14	0.544	BUS
	0.674	0.1	0.685	0.674	0.68	0.791	CAR
	0.473	0.022	0.541	0.473	0.505	0.72	TRAIN
	0.805	0.06	0.697	0.805	0.747	0.882	WALK
Weighted Avg.	0.743	0.119	0.744	0.743	0.743	0.819	

A.1.3 GPS Satellite Classifiers

SATELLITE_COUNT_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.973	0.838	0.556	0.973	0.708	0.611	BIKE
	0	0	0	0	0	0.659	BUS
	0.049	0.011	0.594	0.049	0.091	0.573	CAR
	0.118	0.013	0.331	0.118	0.174	0.75	TRAIN
	0.168	0.033	0.468	0.168	0.247	0.661	WALK
Weighted Avg.	0.548	0.443	0.52	0.548	0.435	0.618	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.97	0.824	0.56	0.97	0.71	0.638	BIKE
	0	0	0	0	0	0.692	BUS
	0.005	0	1	0.005	0.011	0.571	CAR
	0	0	0	0	0	0.795	TRAIN
	0.253	0.072	0.375	0.253	0.302	0.665	WALK
Weighted Avg.	0.542	0.438	0.589	0.542	0.415	0.636	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.969	0.827	0.558	0.969	0.708	0.642	BIKE
	0.003	0.001	0.107	0.003	0.006	0.684	BUS
	0.054	0.014	0.551	0.054	0.099	0.593	CAR
	0.152	0.018	0.327	0.152	0.208	0.771	TRAIN
	0.153	0.032	0.452	0.153	0.228	0.678	WALK
Weighted Avg.	0.547	0.438	0.512	0.547	0.437	0.644	

SNR_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.943	0.609	0.626	0.943	0.752	0.701	BIKE
	0	0	0	0	0	0.715	BUS
	0.327	0.025	0.81	0.327	0.466	0.664	CAR
	0.18	0.018	0.357	0.18	0.239	0.859	TRAIN
	0.239	0.067	0.38	0.239	0.294	0.659	WALK
Weighted Avg.	0.614	0.333	0.597	0.614	0.56	0.695	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.937	0.584	0.634	0.937	0.756	0.747	BIKE
	0	0	0	0	0	0.784	BUS

	0.329	0.025	0.81	0.329	0.468	0.701	CAR
	0.123	0.011	0.395	0.123	0.188	0.913	TRAIN
	0.268	0.092	0.335	0.268	0.297	0.648	WALK
Weighted Avg.	0.612	0.323	0.596	0.612	0.56	0.732	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.623	0.402	0.626	0.623	0.625	0.611	BIKE
	0.087	0.035	0.087	0.087	0.087	0.529	BUS
	0.408	0.188	0.411	0.408	0.41	0.609	CAR
	0.238	0.042	0.24	0.238	0.239	0.596	TRAIN
	0.218	0.14	0.212	0.218	0.215	0.541	WALK
Weighted Avg.	0.471	0.279	0.472	0.471	0.471	0.596	

All GPS satellite features

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.926	0.561	0.64	0.926	0.757	0.742	BIKE
	0.073	0.01	0.217	0.073	0.109	0.722	BUS
	0.381	0.06	0.673	0.381	0.487	0.72	CAR
	0.308	0.023	0.432	0.308	0.36	0.85	TRAIN
	0.187	0.039	0.451	0.187	0.265	0.674	WALK
Weighted Avg.	0.62	0.313	0.593	0.62	0.574	0.731	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.928	0.56	0.641	0.928	0.759	0.759	BIKE
	0	0	0	0	0	0.79	BUS
	0.318	0.024	0.808	0.318	0.457	0.71	CAR
	0.379	0.033	0.389	0.379	0.384	0.915	TRAIN
	0.187	0.087	0.271	0.187	0.221	0.687	WALK
Weighted Avg.	0.607	0.311	0.59	0.607	0.558	0.746	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.652	0.382	0.648	0.652	0.65	0.634	BIKE
	0.106	0.034	0.108	0.106	0.107	0.539	BUS
	0.439	0.179	0.441	0.439	0.44	0.628	CAR
	0.303	0.04	0.299	0.303	0.301	0.628	TRAIN
	0.258	0.126	0.262	0.258	0.26	0.565	WALK
Weighted Avg.	0.503	0.264	0.502	0.503	0.503	0.618	

A.1.4 Light Level Classifiers

MAGNITUDE_HIGH_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.499	BIKE
	0	0	0	0	0	0.498	BUS

	0	0	0.333	0	0	0.5	CAR
	0	0	0	0	0	0.502	TRAIN
	0	0	0	0	0	0.498	WALK
Weighted Avg.	0.519	0.519	0.351	0.519	0.355	0.499	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.499	BIKE
	0	0	0	0	0	0.501	BUS
	0.998	0.999	0.243	0.998	0.391	0.5	CAR
	0	0	0	0	0	0.501	TRAIN
	0.002	0.001	0.296	0.002	0.004	0.501	WALK
Weighted Avg.	0.243	0.244	0.103	0.243	0.096	0.5	

MAGNITUDE_LOW_MID_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.499	TRAIN
	0	0	0	0	0	0.499	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.999	0.997	0.519	0.999	0.684	0.501	BIKE
	0	0	0	0	0	0.5	BUS
	0.001	0.001	0.235	0.001	0.002	0.5	CAR
	0	0	0	0	0	0.501	TRAIN
	0	0	0	0	0	0.501	WALK
Weighted Avg.	0.519	0.518	0.327	0.519	0.355	0.501	

MAGNITUDE_LOW_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.499	WALK

Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.999	0.997	0.519	0.999	0.684	0.501	BIKE
	0	0	0	0	0	0.5	BUS
	0.002	0.001	0.289	0.002	0.003	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.501	WALK
Weighted Avg.	0.519	0.518	0.34	0.519	0.356	0.501	

MAGNITUDE_MAXIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0.999	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.498	BUS
	0.001	0	0.75	0.001	0.002	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.499	WALK
Weighted Avg.	0.519	0.519	0.452	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0.001	0.214	0	0	0.5	BIKE
	0	0	0	0	0	0.501	BUS
	0.999	0.999	0.243	0.999	0.391	0.5	CAR
	0	0	0	0	0	0.502	TRAIN
	0.002	0	0.5	0.002	0.003	0.501	WALK
Weighted Avg.	0.243	0.244	0.244	0.243	0.096	0.5	

MAGNITUDE_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0.999	0.519	1	0.683	0.499	BIKE
	0	0	0	0	0	0.499	BUS
	0.001	0	0.75	0.001	0.002	0.5	CAR
	0	0	0	0	0	0.502	TRAIN
	0	0	0	0	0	0.498	WALK
Weighted Avg.	0.519	0.519	0.452	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class

	0	0.001	0.2	0	0	0.442	BIKE
	0	0	0	0	0	0.54	BUS
	0.999	0.999	0.243	0.999	0.391	0.516	CAR
	0	0	0	0	0	0.54	TRAIN
	0.001	0	0.455	0.001	0.003	0.477	WALK
Weighted Avg.	0.243	0.244	0.23	0.243	0.096	0.474	

MAGNITUDE_MID_HIGH_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.499	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.999	0.997	0.519	0.999	0.684	0.501	BIKE
	0	0	0	0	0	0.5	BUS
	0.002	0.001	0.349	0.002	0.005	0.5	CAR
	0	0	0	0	0	0.501	TRAIN
	0	0	0	0	0	0.501	WALK
Weighted Avg.	0.519	0.518	0.355	0.519	0.356	0.501	

MAGNITUDE_MID_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.999	0.997	0.519	0.999	0.684	0.501	BIKE
	0	0	0	0	0	0.5	BUS
	0.002	0.001	0.349	0.002	0.005	0.5	CAR

	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.518	0.355	0.519	0.356	0.5	

MAGNITUDE_MINIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.499	BIKE
	0	0	0	0	0	0.499	BUS
	0.001	0	0.714	0.001	0.002	0.5	CAR
	0	0	0	0	0	0.501	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.443	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0.016	0.005	0	0	0.49	BIKE
	0	0	0	0	0	0.497	BUS
	0.99	0.992	0.243	0.99	0.39	0.499	CAR
	0	0	0	0	0	0.549	TRAIN
	0.002	0	0.467	0.002	0.004	0.504	WALK
Weighted Avg.	0.241	0.25	0.13	0.241	0.096	0.497	

MAGNITUDE_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.499	BIKE
	0	0	0	0	0	0.498	BUS
	0	0	0.5	0	0	0.499	CAR
	0	0	0	0	0	0.501	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.391	0.519	0.355	0.499	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0.5	0	0	0.499	BIKE
	0	0	0	0	0	0.501	BUS
	0.999	0.999	0.243	0.999	0.391	0.5	CAR
	0	0	0	0	0	0.501	TRAIN
	0.002	0.001	0.292	0.002	0.004	0.501	WALK
Weighted Avg.	0.243	0.243	0.362	0.243	0.096	0.5	

All light level features

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0	0	0	0	0	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.5	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.5	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0.999	0.519	1	0.683	0.5	BIKE
	0	0	0	0	0	0.499	BUS
	0.001	0	0.714	0.001	0.002	0.5	CAR
	0	0	0	0	0	0.5	TRAIN
	0	0	0	0	0	0.499	WALK
Weighted Avg.	0.519	0.519	0.443	0.519	0.355	0.5	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0.25	0	0	0.527	BIKE
	0	0	0	0	0	0.457	BUS
	0.999	0.999	0.243	0.999	0.391	0.513	CAR
	0	0	0	0	0	0.457	TRAIN
	0.001	0	0.357	0.001	0.003	0.522	WALK
Weighted Avg.	0.243	0.244	0.242	0.243	0.096	0.517	

A.1.5 Magnetic Field Strength Classifiers

MAGNITUDE_HIGH_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.902	0.727	0.572	0.902	0.7	0.632	BIKE
	0	0	0	0	0	0.529	BUS
	0.225	0.12	0.378	0.225	0.282	0.59	CAR
	0.001	0	0.5	0.001	0.001	0.611	TRAIN
	0.123	0.022	0.493	0.123	0.197	0.591	WALK
Weighted Avg.	0.541	0.41	0.488	0.541	0.461	0.611	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.992	0.96	0.527	0.992	0.688	0.629	BIKE
	0	0	0	0	0	0.489	BUS
	0	0	0.5	0	0.001	0.588	CAR
	0.001	0.001	0.053	0.001	0.001	0.517	TRAIN
	0.088	0.012	0.567	0.088	0.152	0.567	WALK
Weighted Avg.	0.528	0.5	0.482	0.528	0.38	0.599	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.563	0.471	0.563	0.563	0.563	0.546	BIKE
	0.034	0.036	0.035	0.034	0.035	0.499	BUS
	0.263	0.239	0.261	0.263	0.262	0.511	CAR
	0.055	0.051	0.057	0.055	0.056	0.502	TRAIN
	0.197	0.141	0.193	0.197	0.195	0.533	WALK
Weighted Avg.	0.389	0.327	0.389	0.389	0.389	0.532	

MAGNITUDE_LOW_MID_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.893	0.569	0.629	0.893	0.738	0.669	BIKE
	0	0	0	0	0	0.564	BUS
	0.386	0.18	0.408	0.386	0.397	0.612	CAR
	0	0	0	0	0	0.665	TRAIN
	0.11	0.018	0.506	0.11	0.18	0.6	WALK
Weighted Avg.	0.574	0.342	0.5	0.574	0.506	0.641	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.988	0.899	0.543	0.988	0.7	0.685	BIKE
	0	0	0	0	0	0.524	BUS
	0.052	0.029	0.363	0.052	0.091	0.622	CAR
	0	0	0	0	0	0.674	TRAIN
	0.07	0.012	0.511	0.07	0.123	0.588	WALK
Weighted Avg.	0.536	0.475	0.445	0.536	0.404	0.649	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.596	0.434	0.597	0.596	0.597	0.58	BIKE
	0.059	0.038	0.057	0.059	0.058	0.51	BUS
	0.292	0.228	0.292	0.292	0.292	0.533	CAR
	0.094	0.053	0.091	0.094	0.092	0.522	TRAIN
	0.193	0.136	0.196	0.193	0.194	0.527	WALK
Weighted Avg.	0.416	0.305	0.417	0.416	0.416	0.555	

MAGNITUDE_LOW_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.905	0.425	0.697	0.905	0.787	0.759	BIKE
	0.419	0.003	0.828	0.419	0.556	0.829	BUS
	0.643	0.151	0.578	0.643	0.609	0.762	CAR
	0.167	0.007	0.577	0.167	0.259	0.79	TRAIN
	0.067	0.012	0.479	0.067	0.117	0.591	WALK
Weighted Avg.	0.661	0.26	0.634	0.661	0.609	0.739	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.953	0.655	0.611	0.953	0.745	0.806	BIKE
	0.013	0.002	0.183	0.013	0.024	0.842	BUS
	0.291	0.071	0.568	0.291	0.385	0.796	CAR
	0	0	0	0	0	0.82	TRAIN
	0.064	0.062	0.151	0.064	0.09	0.484	WALK
Weighted Avg.	0.576	0.367	0.484	0.576	0.494	0.758	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.678	0.346	0.679	0.678	0.678	0.665	BIKE
	0.426	0.022	0.428	0.426	0.427	0.706	BUS
	0.464	0.17	0.467	0.464	0.465	0.645	CAR
	0.239	0.043	0.238	0.239	0.238	0.597	TRAIN
	0.206	0.14	0.203	0.206	0.205	0.533	WALK
Weighted Avg.	0.524	0.245	0.524	0.524	0.524	0.639	

MAGNITUDE_MAXIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.913	0.412	0.705	0.913	0.796	0.763	BIKE
	0.484	0.006	0.764	0.484	0.592	0.827	BUS
	0.653	0.151	0.582	0.653	0.616	0.769	CAR
	0.175	0.003	0.754	0.175	0.283	0.803	TRAIN
	0.071	0.01	0.557	0.071	0.126	0.579	WALK
Weighted Avg.	0.671	0.252	0.658	0.671	0.619	0.742	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.955	0.692	0.598	0.955	0.736	0.811	BIKE
	0	0	0	0	0	0.852	BUS
	0.249	0.061	0.57	0.249	0.347	0.803	CAR
	0	0	0	0	0	0.829	TRAIN
	0.073	0.063	0.166	0.073	0.102	0.488	WALK
Weighted Avg.	0.567	0.383	0.474	0.567	0.481	0.764	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.693	0.336	0.69	0.693	0.692	0.68	BIKE
	0.486	0.02	0.484	0.486	0.485	0.737	BUS
	0.469	0.171	0.469	0.469	0.469	0.651	CAR
	0.257	0.039	0.267	0.257	0.262	0.607	TRAIN
	0.215	0.135	0.216	0.215	0.215	0.541	WALK
Weighted Avg.	0.537	0.238	0.536	0.537	0.537	0.651	

MAGNITUDE_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.92	0.386	0.72	0.92	0.808	0.779	BIKE
	0.493	0.005	0.805	0.493	0.611	0.838	BUS
	0.688	0.154	0.59	0.688	0.635	0.785	CAR
	0.195	0.003	0.778	0.195	0.312	0.823	TRAIN
	0.066	0.008	0.578	0.066	0.118	0.564	WALK
Weighted Avg.	0.683	0.239	0.674	0.683	0.631	0.753	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.954	0.619	0.624	0.954	0.755	0.828	BIKE
	0.123	0.002	0.691	0.123	0.208	0.862	BUS
	0.359	0.077	0.601	0.359	0.449	0.817	CAR
	0	0	0	0	0	0.838	TRAIN
	0.039	0.058	0.105	0.039	0.057	0.473	WALK
Weighted Avg.	0.593	0.349	0.512	0.593	0.517	0.775	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.71	0.318	0.706	0.71	0.708	0.697	BIKE
	0.485	0.02	0.487	0.485	0.486	0.736	BUS
	0.495	0.163	0.495	0.495	0.495	0.666	CAR
	0.296	0.04	0.294	0.296	0.295	0.628	TRAIN
	0.225	0.131	0.229	0.225	0.227	0.548	WALK
Weighted Avg.	0.556	0.227	0.555	0.556	0.555	0.665	

MAGNITUDE_MID_HIGH_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.949	0.743	0.579	0.949	0.719	0.637	BIKE
	0	0	0	0	0	0.534	BUS

	0.27	0.087	0.498	0.27	0.35	0.625	CAR
	0.001	0	0.667	0.001	0.003	0.581	TRAIN
	0.055	0.012	0.441	0.055	0.097	0.562	WALK
Weighted Avg.	0.566	0.409	0.522	0.566	0.473	0.616	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.988	0.895	0.544	0.988	0.702	0.662	BIKE
	0	0	0	0	0	0.534	BUS
	0.097	0.019	0.623	0.097	0.167	0.637	CAR
	0	0	0	0	0	0.53	TRAIN
	0.056	0.013	0.433	0.056	0.099	0.544	WALK
Weighted Avg.	0.545	0.471	0.497	0.545	0.419	0.627	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.583	0.45	0.583	0.583	0.583	0.565	BIKE
	0.043	0.036	0.044	0.043	0.043	0.503	BUS
	0.305	0.225	0.304	0.305	0.304	0.541	CAR
	0.058	0.052	0.058	0.058	0.058	0.5	TRAIN
	0.175	0.143	0.174	0.175	0.175	0.517	WALK
Weighted Avg.	0.407	0.314	0.407	0.407	0.407	0.546	

MAGNITUDE_MID_RANGE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.867	0.658	0.587	0.867	0.7	0.623	BIKE
	0	0	0	0	0	0.525	BUS
	0.287	0.142	0.394	0.287	0.332	0.582	CAR
	0	0	0	0	0	0.585	TRAIN
	0.17	0.036	0.446	0.17	0.246	0.6	WALK
Weighted Avg.	0.545	0.382	0.466	0.545	0.48	0.604	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.99	0.948	0.53	0.99	0.69	0.639	BIKE
	0	0	0	0	0	0.483	BUS
	0.004	0.003	0.32	0.004	0.007	0.585	CAR
	0	0	0	0	0	0.526	TRAIN
	0.101	0.015	0.542	0.101	0.171	0.566	WALK
Weighted Avg.	0.53	0.495	0.433	0.53	0.385	0.603	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.581	0.445	0.585	0.581	0.583	0.566	BIKE
	0.033	0.038	0.033	0.033	0.033	0.497	BUS
	0.287	0.234	0.283	0.287	0.285	0.525	CAR
	0.067	0.051	0.068	0.067	0.067	0.509	TRAIN
	0.187	0.141	0.186	0.187	0.186	0.524	WALK
Weighted Avg.	0.404	0.313	0.404	0.404	0.404	0.544	

MAGNITUDE_MINIMUM

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.903	0.378	0.721	0.903	0.802	0.783	BIKE
	0.451	0.002	0.893	0.451	0.599	0.845	BUS
	0.701	0.175	0.563	0.701	0.625	0.782	CAR
	0.223	0.005	0.697	0.223	0.338	0.819	TRAIN
	0.05	0.004	0.698	0.05	0.093	0.568	WALK

Weighted Avg.	0.676	0.24	0.684	0.676	0.622	0.756	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.948	0.583	0.637	0.948	0.762	0.823	BIKE
	0.236	0.002	0.814	0.236	0.366	0.862	BUS
	0.387	0.097	0.562	0.387	0.459	0.804	CAR
	0	0	0	0	0	0.825	TRAIN
	0.032	0.052	0.095	0.032	0.048	0.479	WALK
Weighted Avg.	0.6	0.334	0.512	0.6	0.528	0.769	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.702	0.323	0.701	0.702	0.702	0.691	BIKE
	0.471	0.021	0.465	0.471	0.468	0.724	BUS
	0.463	0.173	0.463	0.463	0.463	0.649	CAR
	0.248	0.042	0.249	0.248	0.249	0.604	TRAIN
	0.213	0.134	0.215	0.213	0.214	0.537	WALK
Weighted Avg.	0.54	0.232	0.539	0.54	0.539	0.655	

MAGNITUDE_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.957	0.827	0.555	0.957	0.703	0.622	BIKE
	0	0	0	0	0	0.551	BUS
	0	0	0.25	0	0.001	0.5	CAR
	0	0	0	0	0	0.663	TRAIN
	0.302	0.071	0.422	0.302	0.352	0.667	WALK
Weighted Avg.	0.541	0.44	0.411	0.541	0.417	0.598	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.012	0.066	0.161	0.012	0.022	0.453	BIKE
	0	0	0	0	0	0.496	BUS
	0.964	0.957	0.245	0.964	0.391	0.503	CAR
	0	0	0	0	0	0.512	TRAIN
	0.012	0.002	0.505	0.012	0.024	0.582	WALK
Weighted Avg.	0.243	0.268	0.217	0.243	0.11	0.489	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.565	0.469	0.565	0.565	0.565	0.549	BIKE
	0.054	0.037	0.054	0.054	0.054	0.511	BUS
	0.263	0.236	0.263	0.263	0.263	0.515	CAR
	0.083	0.051	0.084	0.083	0.083	0.515	TRAIN
	0.234	0.134	0.232	0.234	0.233	0.551	WALK
Weighted Avg.	0.398	0.325	0.398	0.398	0.398	0.538	

All magnetic field strength features

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.862	0.239	0.795	0.862	0.827	0.842	BIKE
	0.533	0.013	0.623	0.533	0.575	0.849	BUS
	0.685	0.1	0.688	0.685	0.686	0.807	CAR
	0.453	0.025	0.505	0.453	0.478	0.775	TRAIN
	0.387	0.068	0.493	0.387	0.434	0.723	WALK
Weighted Avg.	0.715	0.16	0.703	0.715	0.707	0.812	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.92	0.416	0.705	0.92	0.798	0.85	BIKE
	0.523	0.034	0.375	0.523	0.437	0.834	BUS
	0.578	0.123	0.601	0.578	0.589	0.822	CAR
	0.039	0.025	0.08	0.039	0.052	0.804	TRAIN
	0.035	0.006	0.487	0.035	0.065	0.535	WALK
Weighted Avg.	0.645	0.25	0.602	0.645	0.586	0.794	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.816	0.226	0.796	0.816	0.806	0.795	BIKE
	0.576	0.018	0.557	0.576	0.567	0.784	BUS
	0.658	0.085	0.714	0.658	0.685	0.786	CAR
	0.452	0.032	0.441	0.452	0.446	0.711	TRAIN
	0.445	0.099	0.436	0.445	0.44	0.673	WALK
Weighted Avg.	0.695	0.155	0.695	0.695	0.695	0.77	

A.1.6 Orientation Classifiers

X_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.951	0.881	0.538	0.951	0.687	0.577	BIKE
	0.281	0.018	0.381	0.281	0.323	0.763	BUS
	0.129	0.022	0.65	0.129	0.216	0.616	CAR
	0.007	0.001	0.278	0.007	0.014	0.689	TRAIN
	0.012	0.004	0.338	0.012	0.024	0.619	WALK
Weighted Avg.	0.538	0.464	0.516	0.538	0.426	0.606	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.519	1	0.683	0.538	BIKE
	0	0	0	0	0	0.795	BUS
	0	0	0	0	0	0.544	CAR
	0	0	0	0	0	0.578	TRAIN
	0	0	0	0	0	0.578	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.557	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.533	0.5	0.535	0.533	0.534	0.52	BIKE
	0.169	0.031	0.175	0.169	0.172	0.572	BUS
	0.311	0.222	0.31	0.311	0.311	0.546	CAR
	0.109	0.051	0.108	0.109	0.108	0.532	TRAIN
	0.185	0.143	0.182	0.185	0.184	0.522	WALK
Weighted Avg.	0.392	0.338	0.392	0.392	0.392	0.529	

X_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.803	0.475	0.646	0.803	0.716	0.721	BIKE
	0.011	0.001	0.379	0.011	0.021	0.765	BUS
	0.49	0.186	0.46	0.49	0.475	0.698	CAR
	0.001	0	0.083	0.001	0.001	0.703	TRAIN
	0.287	0.06	0.453	0.287	0.351	0.706	WALK
Weighted Avg.	0.579	0.301	0.532	0.579	0.54	0.714	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class

	1	1	0.519	1	0.683	0.496	BIKE
	0	0	0	0	0	0.671	BUS
	0	0	0	0	0	0.517	CAR
	0	0	0	0	0	0.523	TRAIN
	0	0	0	0	0	0.702	WALK
Weighted Avg.	0.519	0.519	0.269	0.519	0.355	0.539	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.597	0.431	0.599	0.597	0.598	0.584	BIKE
	0.107	0.036	0.103	0.107	0.105	0.54	BUS
	0.325	0.215	0.327	0.325	0.326	0.553	CAR
	0.09	0.052	0.089	0.09	0.09	0.523	TRAIN
	0.269	0.128	0.266	0.269	0.268	0.572	WALK
Weighted Avg.	0.437	0.299	0.438	0.437	0.438	0.57	

Y_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.887	0.104	0.902	0.887	0.894	0.949	BIKE
	0.42	0.01	0.625	0.42	0.502	0.92	BUS
	0.891	0.081	0.781	0.891	0.832	0.952	CAR
	0.498	0.019	0.6	0.498	0.544	0.918	TRAIN
	0.628	0.059	0.648	0.628	0.638	0.905	WALK
Weighted Avg.	0.812	0.084	0.809	0.812	0.808	0.94	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.953	0.606	0.629	0.953	0.758	0.776	BIKE
	0	0	0	0	0	0.627	BUS
	0.513	0.055	0.748	0.513	0.608	0.911	CAR
	0.357	0.024	0.458	0.357	0.401	0.82	TRAIN
	0.005	0.006	0.137	0.005	0.01	0.756	WALK
Weighted Avg.	0.639	0.33	0.553	0.639	0.564	0.803	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.855	0.156	0.855	0.855	0.855	0.851	BIKE
	0.401	0.024	0.391	0.401	0.396	0.691	BUS
	0.763	0.078	0.759	0.763	0.761	0.843	CAR
	0.43	0.031	0.435	0.43	0.433	0.706	TRAIN
	0.531	0.079	0.536	0.531	0.534	0.73	WALK
Weighted Avg.	0.745	0.114	0.745	0.745	0.745	0.818	

Y_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.85	0.218	0.808	0.85	0.829	0.874	BIKE
	0	0	0	0	0	0.765	BUS
	0.736	0.135	0.637	0.736	0.683	0.854	CAR
	0.384	0.029	0.422	0.384	0.402	0.895	TRAIN
	0.585	0.045	0.692	0.585	0.634	0.877	WALK
Weighted Avg.	0.727	0.154	0.699	0.727	0.711	0.866	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.985	0.869	0.55	0.985	0.706	0.659	BIKE
	0	0	0	0	0	0.49	BUS
	0.018	0.026	0.183	0.018	0.032	0.501	CAR

	0.05	0.009	0.232	0.05	0.082	0.583	TRAIN
	0.185	0.01	0.761	0.185	0.297	0.82	WALK
Weighted Avg.	0.545	0.459	0.454	0.545	0.422	0.634	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.758	0.257	0.761	0.758	0.759	0.748	BIKE
	0.071	0.034	0.074	0.071	0.072	0.522	BUS
	0.541	0.146	0.544	0.541	0.543	0.698	CAR
	0.278	0.041	0.277	0.278	0.278	0.629	TRAIN
	0.521	0.088	0.505	0.521	0.513	0.718	WALK
Weighted Avg.	0.619	0.185	0.619	0.619	0.619	0.717	

Z_MEAN

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.803	0.231	0.789	0.803	0.796	0.837	BIKE
	0	0	0	0	0	0.679	BUS
	0.81	0.133	0.662	0.81	0.729	0.883	CAR
	0.047	0.003	0.486	0.047	0.086	0.721	TRAIN
	0.613	0.093	0.533	0.613	0.57	0.846	WALK
Weighted Avg.	0.706	0.166	0.675	0.706	0.679	0.838	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.891	0.796	0.547	0.891	0.678	0.75	BIKE
	0	0	0	0	0	0.61	BUS
	0.254	0.123	0.4	0.254	0.311	0.692	CAR
	0	0	0	0	0	0.616	TRAIN
	0	0	0	0	0	0.778	WALK
Weighted Avg.	0.525	0.443	0.382	0.525	0.428	0.728	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.71	0.319	0.706	0.71	0.708	0.696	BIKE
	0.108	0.034	0.11	0.108	0.109	0.537	BUS
	0.59	0.133	0.588	0.59	0.589	0.73	CAR
	0.125	0.05	0.122	0.125	0.123	0.544	TRAIN
	0.407	0.098	0.417	0.407	0.412	0.657	WALK
Weighted Avg.	0.582	0.216	0.581	0.582	0.582	0.685	

Z_VARIANCE

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.854	0.288	0.762	0.854	0.805	0.837	BIKE
	0	0	0	0	0	0.794	BUS
	0.63	0.155	0.566	0.63	0.597	0.788	CAR
	0.203	0.019	0.372	0.203	0.262	0.873	TRAIN
	0.605	0.035	0.751	0.605	0.67	0.846	WALK
Weighted Avg.	0.696	0.193	0.663	0.696	0.676	0.826	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.961	0.788	0.568	0.961	0.714	0.821	BIKE
	0	0	0	0	0	0.775	BUS
	0.046	0.02	0.426	0.046	0.083	0.76	CAR
	0	0	0	0	0	0.839	TRAIN
	0.534	0.02	0.818	0.534	0.646	0.804	WALK
Weighted Avg.	0.588	0.417	0.519	0.588	0.486	0.803	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.721	0.302	0.72	0.721	0.721	0.708	BIKE
	0.106	0.035	0.105	0.106	0.105	0.538	BUS
	0.437	0.181	0.438	0.437	0.438	0.626	CAR
	0.215	0.044	0.214	0.215	0.214	0.587	TRAIN
	0.54	0.079	0.542	0.54	0.541	0.735	WALK
Weighted Avg.	0.575	0.216	0.575	0.575	0.575	0.679	

All orientation features

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.97	0.034	0.968	0.97	0.969	0.98	BIKE
	0.835	0.005	0.874	0.835	0.854	0.931	BUS
	0.967	0.011	0.967	0.967	0.967	0.981	CAR
	0.928	0.004	0.935	0.928	0.931	0.974	TRAIN
	0.894	0.02	0.887	0.894	0.89	0.956	WALK
Weighted Avg.	0.951	0.024	0.951	0.951	0.951	0.975	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.946	0.594	0.632	0.946	0.758	0.885	BIKE
	0.271	0.033	0.244	0.271	0.257	0.841	BUS
	0.184	0.019	0.755	0.184	0.296	0.901	CAR
	0.302	0.01	0.639	0.302	0.41	0.878	TRAIN
	0.537	0.022	0.81	0.537	0.646	0.887	WALK
Weighted Avg.	0.641	0.318	0.674	0.641	0.592	0.888	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.974	0.027	0.975	0.974	0.975	0.974	BIKE
	0.898	0.004	0.9	0.898	0.899	0.953	BUS
	0.967	0.01	0.97	0.967	0.968	0.979	CAR
	0.936	0.004	0.922	0.936	0.929	0.965	TRAIN
	0.914	0.015	0.912	0.914	0.913	0.951	WALK
Weighted Avg.	0.959	0.019	0.959	0.959	0.959	0.971	

A.2 Multiple Sensor Classifiers

A.2.1 All Sensor Classifiers

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.99	0.016	0.986	0.99	0.988	0.989	BIKE
	0.937	0.001	0.964	0.937	0.95	0.968	BUS
	0.977	0.008	0.975	0.977	0.976	0.984	CAR
	0.945	0.003	0.951	0.945	0.948	0.979	TRAIN
	0.951	0.007	0.957	0.951	0.954	0.971	WALK
Weighted Avg.	0.976	0.011	0.976	0.976	0.976	0.984	

NaiveBayes

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.912	0.126	0.886	0.912	0.899	0.946	BIKE
	0.884	0.073	0.32	0.884	0.47	0.953	BUS
	0.754	0.043	0.849	0.754	0.799	0.938	CAR
	0.633	0.019	0.646	0.633	0.639	0.951	TRAIN
	0.561	0.014	0.875	0.561	0.684	0.968	WALK

Weighted Avg.	0.806	0.082	0.842	0.806	0.813	0.948	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.061	0.011	0.857	0.061	0.114	0.529	BIKE
	0.024	0	0.96	0.024	0.047	0.513	BUS
	0.777	0.524	0.323	0.777	0.456	0.628	CAR
	0.236	0	0.994	0.236	0.382	0.621	TRAIN
	0.712	0.303	0.288	0.712	0.41	0.706	WALK
Weighted Avg.	0.339	0.178	0.655	0.339	0.253	0.583	

A.2.2 All Sensors except Light Level Classifiers

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.989	0.014	0.987	0.989	0.988	0.989	BIKE
	0.953	0.002	0.959	0.953	0.956	0.973	BUS
	0.977	0.007	0.977	0.977	0.977	0.983	CAR
	0.946	0.003	0.955	0.946	0.95	0.978	TRAIN
	0.955	0.007	0.958	0.955	0.957	0.975	WALK
Weighted Avg.	0.978	0.01	0.978	0.978	0.978	0.984	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.911	0.126	0.886	0.911	0.898	0.946	BIKE
	0.88	0.074	0.318	0.88	0.467	0.952	BUS
	0.752	0.044	0.848	0.752	0.797	0.939	CAR
	0.639	0.02	0.643	0.639	0.641	0.951	TRAIN
	0.561	0.013	0.879	0.561	0.685	0.968	WALK
Weighted Avg.	0.805	0.082	0.841	0.805	0.813	0.948	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.986	0.022	0.979	0.986	0.983	0.982	BIKE
	0.924	0.003	0.921	0.924	0.922	0.958	BUS
	0.977	0.007	0.979	0.977	0.978	0.985	CAR
	0.958	0.002	0.965	0.958	0.961	0.978	TRAIN
	0.933	0.008	0.952	0.933	0.942	0.962	WALK
Weighted Avg.	0.972	0.015	0.972	0.972	0.972	0.979	

A.2.3 Accelerometer and Orientation Classifiers

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.974	0.032	0.97	0.974	0.972	0.974	BIKE
	0.889	0.004	0.907	0.889	0.898	0.939	BUS
	0.967	0.01	0.969	0.967	0.968	0.979	CAR
	0.93	0.004	0.929	0.93	0.93	0.968	TRAIN
	0.899	0.017	0.903	0.899	0.901	0.947	WALK
Weighted Avg.	0.956	0.022	0.955	0.956	0.956	0.97	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.781	0.1	0.894	0.781	0.834	0.887	BIKE
	0.86	0.145	0.188	0.86	0.308	0.916	BUS
	0.431	0.019	0.882	0.431	0.579	0.904	CAR
	0.592	0.108	0.236	0.592	0.337	0.891	TRAIN
	0.613	0.038	0.736	0.613	0.669	0.787	WALK

Weighted Avg.	0.664	0.073	0.806	0.664	0.701	0.878	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.98	0.031	0.972	0.98	0.976	0.975	BIKE
	0.92	0.003	0.934	0.92	0.927	0.96	BUS
	0.973	0.008	0.976	0.973	0.975	0.983	CAR
	0.941	0.004	0.929	0.941	0.935	0.969	TRAIN
	0.905	0.012	0.931	0.905	0.918	0.947	WALK
Weighted Avg.	0.963	0.02	0.963	0.963	0.963	0.972	

A.2.4 All non-GPS except Light Level Classifiers

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.981	0.022	0.98	0.981	0.98	0.982	BIKE
	0.969	0.001	0.963	0.969	0.966	0.984	BUS
	0.976	0.008	0.976	0.976	0.976	0.984	CAR
	0.927	0.003	0.947	0.927	0.937	0.971	TRAIN
	0.923	0.014	0.92	0.923	0.921	0.956	WALK
Weighted Avg.	0.968	0.015	0.968	0.968	0.968	0.978	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.87	0.135	0.874	0.87	0.872	0.92	BIKE
	0.692	0.047	0.363	0.692	0.476	0.94	BUS
	0.761	0.095	0.72	0.761	0.74	0.911	CAR
	0.513	0.032	0.471	0.513	0.491	0.889	TRAIN
	0.535	0.022	0.81	0.535	0.645	0.862	WALK
Weighted Avg.	0.768	0.1	0.787	0.768	0.771	0.909	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.982	0.028	0.974	0.982	0.978	0.977	BIKE
	0.941	0.001	0.966	0.941	0.954	0.969	BUS
	0.975	0.008	0.977	0.975	0.976	0.984	CAR
	0.948	0.004	0.932	0.948	0.94	0.969	TRAIN
	0.911	0.011	0.936	0.911	0.923	0.949	WALK
Weighted Avg.	0.967	0.018	0.967	0.967	0.967	0.974	

A.2.5 All GPS Classifiers

J48	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.915	0.142	0.874	0.915	0.894	0.918	BIKE
	0.318	0.018	0.409	0.318	0.358	0.682	BUS
	0.772	0.057	0.814	0.772	0.793	0.88	CAR
	0.711	0.01	0.798	0.711	0.752	0.894	TRAIN
	0.848	0.029	0.834	0.848	0.841	0.948	WALK
Weighted Avg.	0.837	0.093	0.832	0.837	0.834	0.903	

NaiveBayes	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.788	0.222	0.793	0.788	0.79	0.878	BIKE
	0.127	0.014	0.263	0.127	0.171	0.789	BUS
	0.577	0.042	0.816	0.577	0.676	0.806	CAR
	0.305	0.01	0.635	0.305	0.412	0.935	TRAIN
	0.742	0.187	0.405	0.742	0.524	0.889	WALK

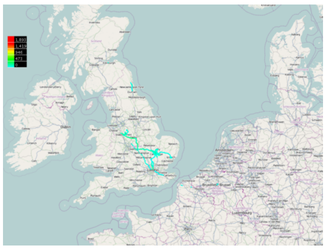
Weighted Avg.	0.679	0.154	0.713	0.679	0.68	0.862	

IBk	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.866	0.179	0.839	0.866	0.852	0.842	BIKE
	0.259	0.026	0.279	0.259	0.269	0.616	BUS
	0.732	0.071	0.768	0.732	0.749	0.828	CAR
	0.672	0.014	0.727	0.672	0.698	0.823	TRAIN
	0.694	0.054	0.688	0.694	0.691	0.822	WALK
Weighted Avg.	0.775	0.12	0.772	0.775	0.773	0.826	

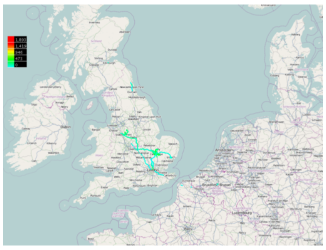
Appendix B

Sensor Feature Visualisations

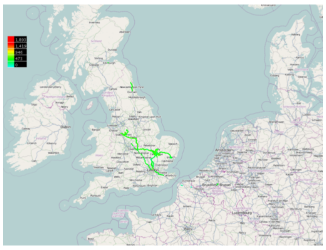
Figures B.1, B.2, B.3 and B.4 show the visualisations of sensor feature values.



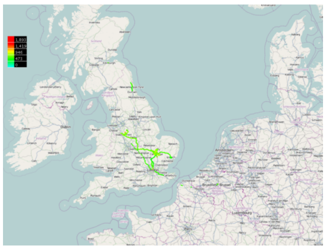
(a) ACCELEROMETER-MAGNITUDE-HIGH-RANGE



(b) ACCELEROMETER-MAGNITUDE-LOW-MID-RANGE

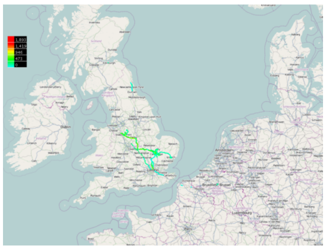


(c) ACCELEROMETER-MAGNITUDE-LOW-RANGE

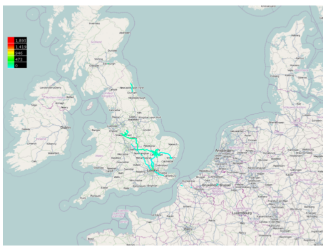


(d) ACCELEROMETER-MAGNITUDE-MAXIMUM

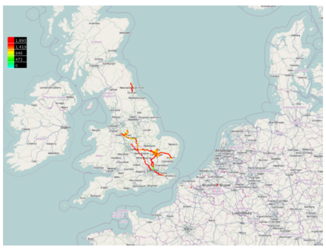
(e) ACCELEROMETER-MAGNITUDE-MEAN



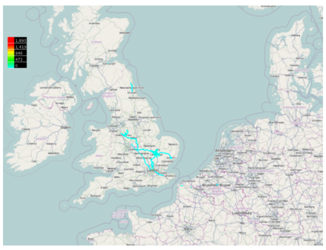
(f) ACCELEROMETER-MAGNITUDE-MID-HIGH-RANGE



(g) ACCELEROMETER-MAGNITUDE-MID-RANGE

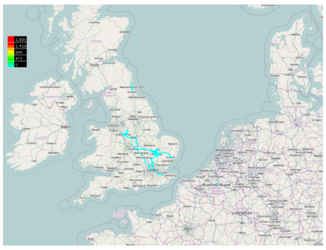


(h) ACCELEROMETER-MAGNITUDE-MINIMUM

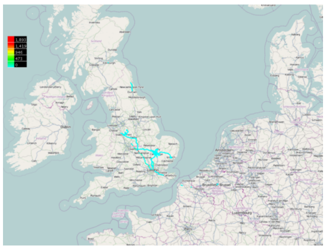


(i) ACCELEROMETER-MAGNITUDE-VARIANCE

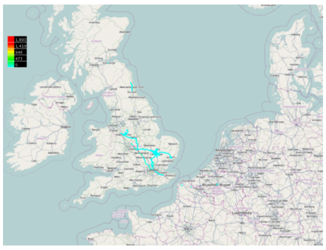
Figure B.1: Feature visualisations



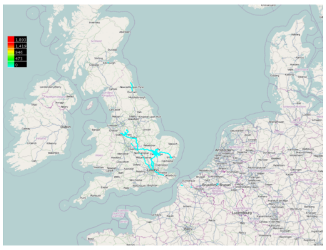
(a) GPS-LOCATION-ALTITUDE-DECREASING-MEAN



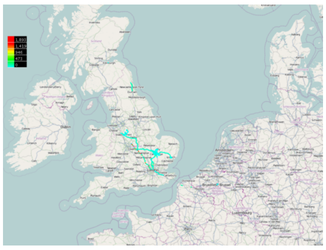
(b) GPS-LOCATION-ALTITUDE-GAIN-MAXIMUM



(c) GPS-LOCATION-ALTITUDE-INCREASING-MEAN

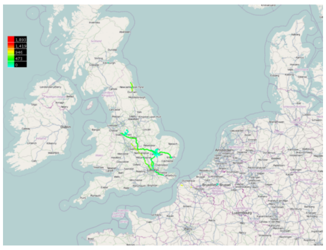


(d) GPS-LOCATION-ALTITUDE-LOSS-MAXIMUM

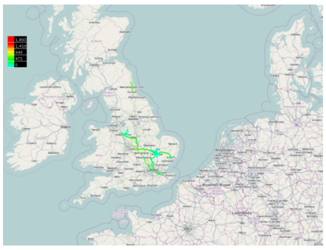


(e) GPS-LOCATION-DISTANCE-TOTAL

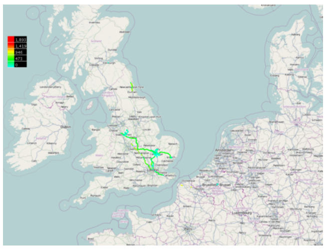
(f) GPS-LOCATION-HEADING-CHANGE-RATE



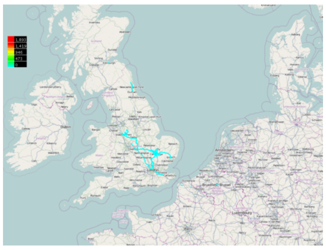
(g) GPS-LOCATION-SPEED-MAXIMUM



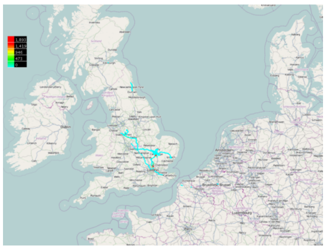
(h) GPS-LOCATION-SPEED-MEAN



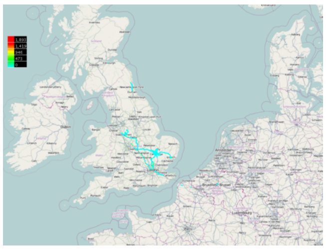
(i) GPS-LOCATION-SPEED-MINIMUM



(j) GPS-LOCATION-SPEED-VARIANCE

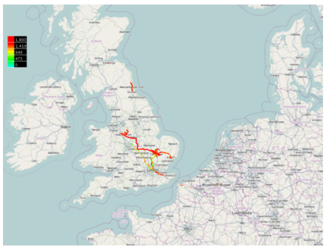


(k) GPS-LOCATION-STOP-RATE



(l) GPS-LOCATION-VELOCITY-CHANGE-RATE

Figure B.2: Feature visualisations



(a) GPS-SATELLITES-SATELLITE-COUNT-MEAN

(b) GPS-SATELLITES-SNR-MEAN

(c) LIGHT-LEVEL-MAGNITUDE-HIGH-RANGE

(d) LIGHT-LEVEL-MAGNITUDE-LOW-MID-RANGE

(e) LIGHT-LEVEL-MAGNITUDE-LOW-RANGE

(f) LIGHT-LEVEL-MAGNITUDE-MAXIMUM

(g) LIGHT-LEVEL-MAGNITUDE-MEAN

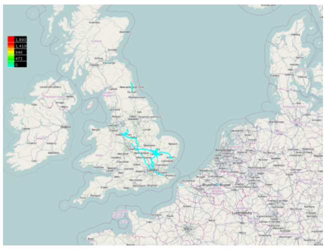
(h) LIGHT-LEVEL-MAGNITUDE-MID-HIGH-RANGE

(i) LIGHT-LEVEL-MAGNITUDE-MID-RANGE

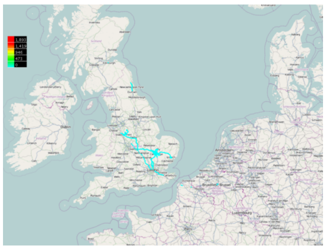
(j) LIGHT-LEVEL-MAGNITUDE-MINIMUM

(k) LIGHT-LEVEL-MAGNITUDE-VARIANCE

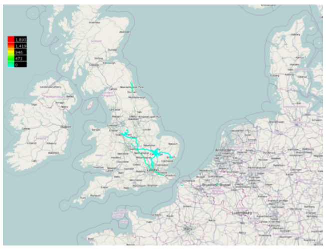
Figure B.3: Feature visualisations



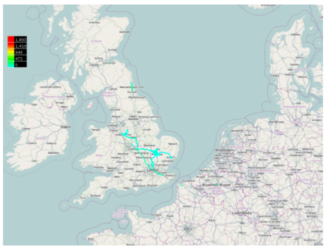
(a) MAGNETIC-FIELD-MAGNITUDE-HIGH-RANGE



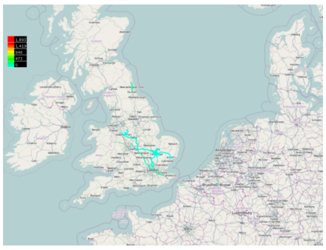
(b) MAGNETIC-FIELD-MAGNITUDE-LOW-MID-RANGE



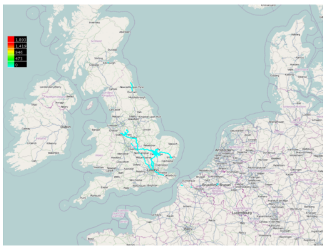
(c) MAGNETIC-FIELD-MAGNITUDE-LOW-RANGE



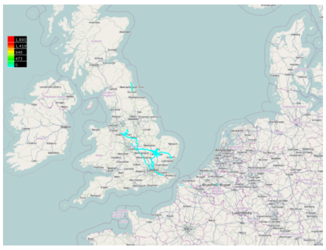
(d) MAGNETIC-FIELD-MAGNITUDE-MAXIMUM



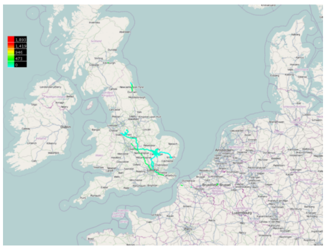
(e) MAGNETIC-FIELD-MAGNITUDE-MEAN



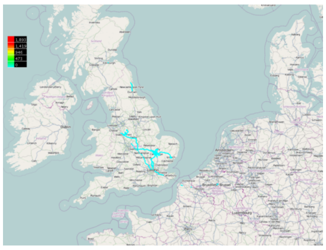
(f) MAGNETIC-FIELD-MAGNITUDE-MID-HIGH-RANGE



(g) MAGNETIC-FIELD-MAGNITUDE-MID-RANGE

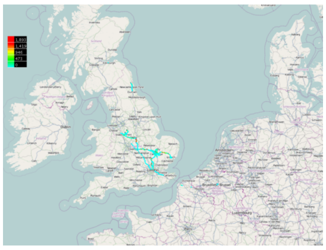


(h) MAGNETIC-FIELD-MAGNITUDE-MINIMUM

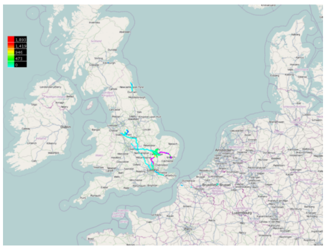


(i) MAGNETIC-FIELD-MAGNITUDE-VARIANCE

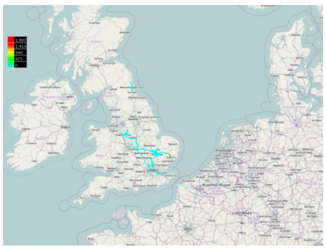
(j) ORIENTATION-X-MEAN



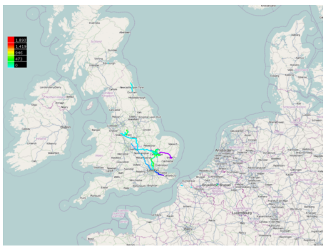
(k) ORIENTATION-X-VARIANCE



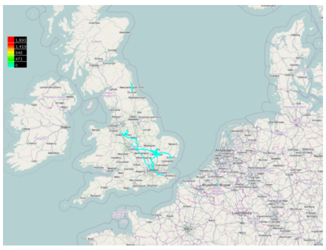
(l) ORIENTATION-Y-MEAN



(m) ORIENTATION-Y-VARIANCE



(n) ORIENTATION-Z-MEAN



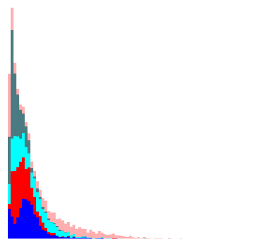
(o) ORIENTATION-Z-VARIANCE

Figure B.4: Feature visualisations

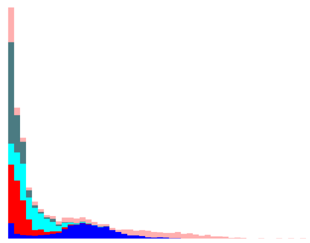
Appendix C

Sensor Feature Distributions

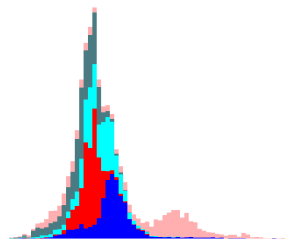
Figures C.1, C.2, C.3 and C.4 show the distributions of sensor feature values as histograms. Each colour represents a transport class. The important property in these histograms is for the distributions to vary across the classes.



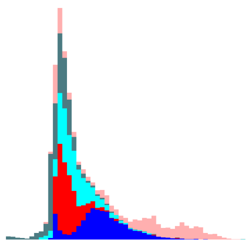
(a) ACCELEROMETER-MAGNITUDE-HIGH-RANGE



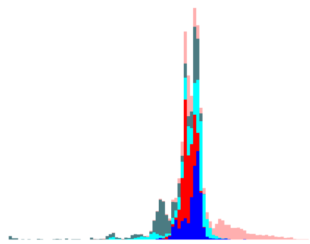
(b) ACCELEROMETER-MAGNITUDE-LOW-MID-RANGE



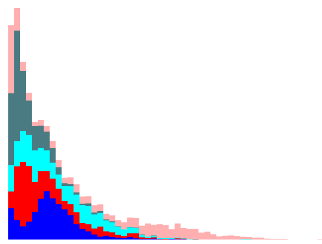
(c) ACCELEROMETER-MAGNITUDE-LOW-RANGE



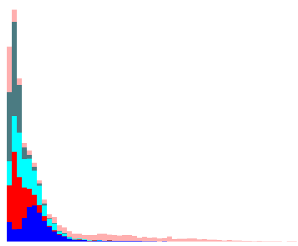
(d) ACCELEROMETER-MAGNITUDE-MAXIMUM



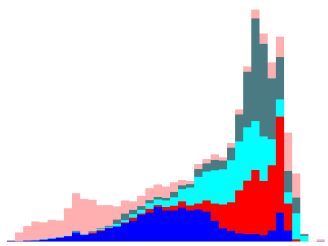
(e) ACCELEROMETER-MAGNITUDE-MEAN



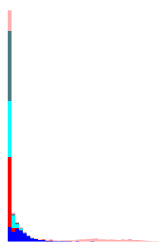
(f) ACCELEROMETER-MAGNITUDE-MID-HIGH-RANGE



(g) ACCELEROMETER-MAGNITUDE-MID-RANGE



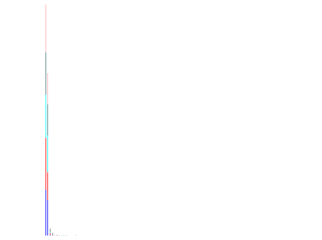
(h) ACCELEROMETER-MAGNITUDE-MINIMUM



(i) ACCELEROMETER-MAGNITUDE-VARIANCE

Figure C.1: Feature distributions

(a) GPS-LOCATION-ALTITUDE-DECREASING-MEAN

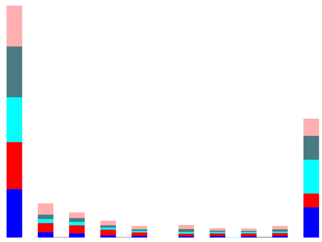


(b) GPS-LOCATION-ALTITUDE-GAIN-MAXIMUM

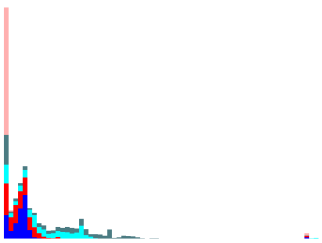
(c) GPS-LOCATION-ALTITUDE-INCREASING-MEAN

(d) GPS-LOCATION-ALTITUDE-LOSS-MAXIMUM

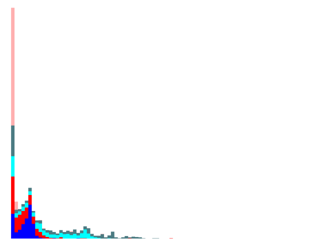
(e) GPS-LOCATION-DISTANCE-TOTAL



(f) GPS-LOCATION-HEADING-CHANGE-RATE



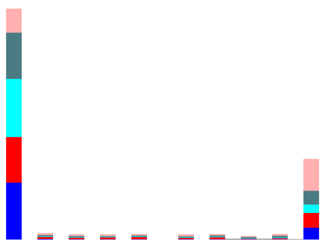
(g) GPS-LOCATION-SPEED-MAXIMUM



(h) GPS-LOCATION-SPEED-MEAN

(i) GPS-LOCATION-SPEED-MINIMUM

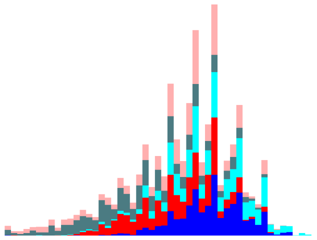
(j) GPS-LOCATION-SPEED-VARIANCE



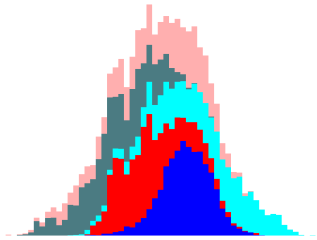
(k) GPS-LOCATION-STOP-RATE

(l) GPS-LOCATION-VELOCITY-CHANGE-RATE

Figure C.2: Feature distributions



(a) GPS-SATELLITES-SATELLITE-COUNT-MEAN



(b) GPS-SATELLITES-SNR-MEAN

(c) LIGHT-LEVEL-MAGNITUDE-HIGH-RANGE

(d) LIGHT-LEVEL-MAGNITUDE-LOW-MID-RANGE

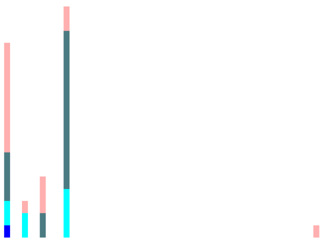
(e) LIGHT-LEVEL-MAGNITUDE-LOW-RANGE

(f) LIGHT-LEVEL-MAGNITUDE-MAXIMUM

(g) LIGHT-LEVEL-MAGNITUDE-MEAN

(h) LIGHT-LEVEL-MAGNITUDE-MID-HIGH-RANGE

(i) LIGHT-LEVEL-MAGNITUDE-MID-RANGE

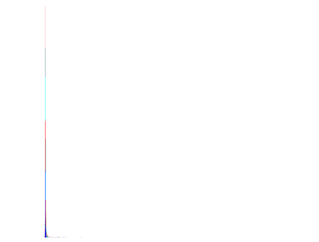


(j) LIGHT-LEVEL-MAGNITUDE-MINIMUM

(k) LIGHT-LEVEL-MAGNITUDE-VARIANCE

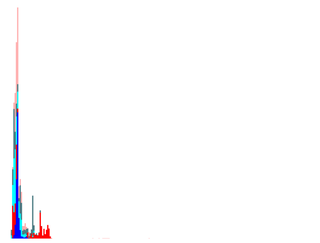
Figure C.3: Feature distributions

(a) MAGNETIC-FIELD-MAGNITUDE-HIGH-RANGE

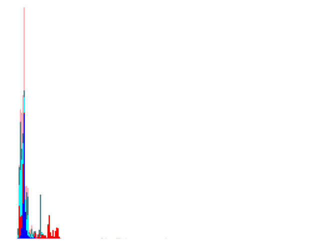


(b) MAGNETIC-FIELD-MAGNITUDE-LOW-MID-RANGE

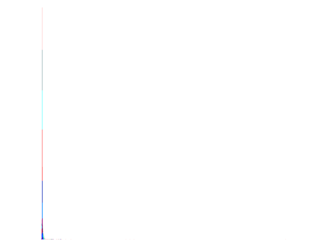
(c) MAGNETIC-FIELD-MAGNITUDE-LOW-RANGE



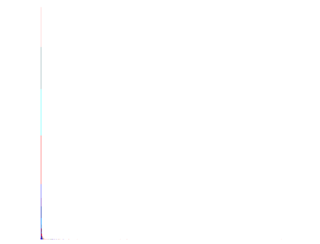
(d) MAGNETIC-FIELD-MAGNITUDE-MAXIMUM



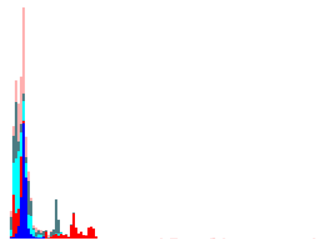
(e) MAGNETIC-FIELD-MAGNITUDE-MEAN



(f) MAGNETIC-FIELD-MAGNITUDE-MID-HIGH-RANGE

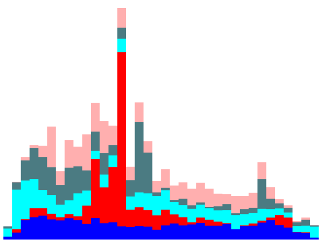


(g) MAGNETIC-FIELD-MAGNITUDE-MID-RANGE

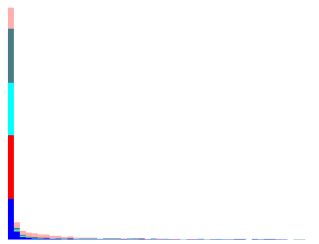


(h) MAGNETIC-FIELD-MAGNITUDE-MINIMUM

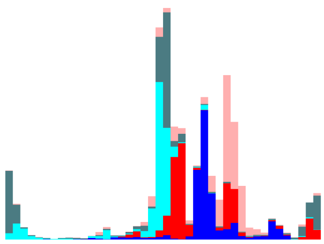
(i) MAGNETIC-FIELD-MAGNITUDE-VARIANCE



(j) ORIENTATION-X-MEAN

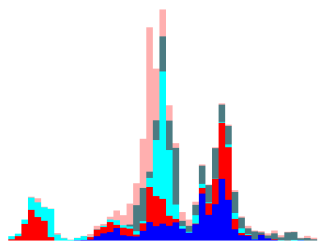


(k) ORIENTATION-X-VARIANCE



(l) ORIENTATION-Y-MEAN

(m) ORIENTATION-Y-VARIANCE



(n) ORIENTATION-Z-MEAN

(o) ORIENTATION-Z-VARIANCE

Figure C.4: Feature distributions

Appendix D

Project Proposal

David Piggott
Fitzwilliam College
dhp26

Computer Science Tripos Part II Project Proposal

Inferring Transportation Mode using Smartphone Sensor Data

October 22, 2010

Project Originator: David Piggott

Project Supervisors: Simon Hay and Mattias Linnap

Director of Studies: Dr. Robert Harle

Overseers: Dr. Jean Bacon and Dr. Andrew Rice

Introduction and Description of the Work

This project will investigate algorithms for infering transportation mode by implementing and comparing two or more algorithms from the literature, using data collected with an Android smartphone. The primary metric of interest in doing this will be the accuracy of the inferences made.

Being able to infer the mode of transport that a person is taking has several applications in the context of ubiquitous computing. One such use is in estimating the personal energy footprint of the user based on the mode of transport, number of people sharing and distance travelled.

As extension activities:

1. A new or a hybrid algorithm may be created and evaluated.

2. An application may be developed that uses whichever algorithm was found to be most appropriate to give users an estimate of their transport energy footprint. This may be done by building on the codebase from my UROP project "A personal energy meter", resulting in a system capable of tracking energy use due to buildings and transportation. Inferring the number of people using shared transport would likely be beyond the scope of this extension.

Resources Required

An Android smartphone with GPS and 3-axis accelerometer. Use of the Android platform dictates that the primary language used will be Java.

Starting Point

I have worked on a UROP project, gaining skills in Android development and software engineering. The project may provide a codebase for the transport energy metering extension.

I am owner of an HTC Hero phone. In the event of this being lost, I would require a replacement phone anyway and so would buy another. As I hope to borrow a G1 phone from the DTG in order to assist data collection, I could also use this as a fallback.

Substance and Structure of the Project

Collection of Sample Data

Test data will collected by writing an Android application that records GPS location at a frequency of 1Hz and accelerometer data at a minimum frequency of 10Hz, along with suitably accurate timestamps. Ground truths will be recorded either by having a button in the application to trigger a log entry, or by identifying segments at the end of each day by recalling the day's events. Data will be collected for at least the following: bicycle, bus, car, train and walk and will be in two stages:

1. Algorithm implementation

   It will be important to get a small amount of test data early on so that I can begin implementing the algorithms. For this I will record traces in and around Cambridge, not necessarily including all transportation modes. This will also allow refinement of the tracer application for:

2. Algorithm evaluation

   For a meaningful analysis a representative quantity of data will be required. It will be necessary to collect data on a variety of routes and in a range of traffic conditions. To assist with this volunteers will provide data by running the tracer application. I may be able to borrow a G1 phone from the DTG to lend to volunteers not in possesion of an Android phone.

   Accurate ground truth records will be best obtained by having volunteers run the application for relatively short periods of time. It may be necessary to allow for 'warm up' time by discarding the first hour/day of data.

Reddy et al. [1] tested with a total of 20 hours of data collected by 6 people. Zheng et al. [2, 3] collected data from 45 people over 6 months covering 20,000km in 15 cities for [2] and for [3] collected data from 65 people over 10 months covering 30,000km in 18 cities.

There is a bug [4] in Android 2.1 that prevents applications receiving accelerometer data when the screen sleeps. At the time of writing (October 17, 2010), the Android Device Dashboard versions page [5] reports 40.4% of active devices are running 2.1 and 33.4% running 2.2. The latest official version for the HTC Hero is 2.1; in my case I've worked around this by running an unofficial 2.2.1 release. Three provisional volunteers also have HTC Hero phones; they may a) be reluctant to change software configuration and b) require assistance installing it. There is a work around for 2.1 [6] though I've not yet tested it.

Table D.1 shows an estimate of the storage requirements for tracing. With 100MiB being the best case the chosen encoding must be space efficient. The non-(GPS location or accelerometer) data may be recorded in anticipation of the first extension activity.

input	value size/bits	values/sample	samples/second	data + timestamp bits/second
accelerometer	32	3	40	3840 + 2560
orientation	32	3	4	384 + 256
magnetic field	32	3	4	384 + 256
light level	32	1	4	128 + 256
GPS location	64	6 (longitude, latitude, altitude, bearing, speed, accuracy)	1	384 + 64
GPS satellites	32	5 (PRN, SNR, azimuth, elevation, used in fix)	variable - max. 255, assuming average of 10	1600 + 0
total				10496

Table D.1: Sensor data bitrate estimate - approx 100MiB/day (assuming no overhead)

Possible log formats are XML, CSV, custom text and custom binary. Issues to be considered are space and processing requirements; there is a compromise to be made between low storage requirements and avoiding excess battery drain due to log generation and compression (as may be required with XML).

General logging strategy may be to have a separate log file for each sensor type, and have the name of the file contain all necessary metadata, i.e. device identifier, sensor type and start time. Ground truths will be logged in separate files and the storage requirement is negligible. A new file will be started everytime: a) the tracer service starts b) sensor data becomes available having previously ceased c) the current log file size reaches some threshold.

Creation of Test Framework

When implemented the inference algorithms will be tested and analysed on a desktop machine, not on the Android platform; inference will not be realtime. Indeed, some algorithms (e.g. [2, 3]) are not suited to realtime analysis due to the segmentation preprocessing step. Testing the algorithms on a desktop machine should increase productivity by avoiding on-phone debugging.

Comparison of Inference Algorithms

Preliminary research into inference algorithms yields three possibilities. In all three papers multiple classification and enhancement algorithms were tested. All found that decision trees and discrete hidden Markov models (DHMMs) resulted in the highest accuracy, and so in the following summary only the accuracy achieved by each paper using these algorithms is considered. Table D.2 summarises their differences.

[1] uses GPS and 3-axis accelerometer data to infer transportation mode (regardless of phone orientation) using features such as force vector magnitude frequency coefficients below 10Hz. The features are input to a decision tree and the resulting inferences are enhanced using a DHMM (considering the likelihood of transitioning from one mode of transport to another, segment by segment). The features are created from constant length segments of 1 second which overlap by 0.5 seconds.

[2] uses only GPS data, but splits the traces first by marking each data point as either walking or non-walking and then growing these points into segments. Segments are thus much longer than those of [1] but still the best inferences were made by use of decision trees and DHMMs. However, the features used are significantly different.

[3] builds on [2] by introducing a spatially indexed database of change points, and using this data to further enhance inferences based on the likelihood of transitioning from one mode of transport to another at the change point in question.

algorithm	[1]	[2]	[3]
input	3-axis accelerometer, GPS	GPS	GPS, changepoint data
segment features	variance (accelerometer), energy (accelerometer), sum of FFT coeffecients 0.5-10Hz (accelerometer), speed (GPS)	length, mean velocity, expectation of velocity, covariance of velocity, top three velocities, top three accelerations	length, ith maximum velocity, ith maximum acceleration, average velocity, expection of velocity, variance of velocity, heading change rate, stop rate, velocity change rate
structure	fixed length segments → decision tree → discrete hidden markov model	walk/non-walk segments → decision tree → discrete hidden markov model	walk/non-walk segments → decision tree → discrete hidden markov model → graph-based post-processing using changepoint data
accuracy	98.8% of segments correctly identified	68.5% of distance correctly identified	76.2% of distance correctly identified
notes	test data was not entirely realistic (volunteers were told to minimise idling)		requires large training set

Table D.2: Comparison of inference algorithms

To give a baseline to the comparison between algorithms I may implement a simple inference algorithm. It might for example split traces into fixed length (e.g. five second) segments and classify each based on its average speed.

Automated transport Energy Metering

In the Energy Metering extension activity, analysis may or may not be realtime depending on the nature of the selected inference algorithm.

Support

Backup and Revision Control

Work will be done on my own machine. All important files are backed up daily via a cron job using rsync over SSH to a remote machine. Manual backups are made to a flash drive (approximately weekly) that I carry at all times. In the event of hardware failure the failed component(s) would be replaced. In the worst case the PWF machines in the Intel lab could be used (though I would have to boot my own OS in order to do Android development).

In terms of revision control, Subversion would be sufficient for a single person project such as this. However, Git provides everything required and more so I will take this opportunity to use a new tool and choose Git.

Supervision

Simon Hay and Mattias Linnap will co-supervise the project. Simon is planning to submit his PhD and so may not be around for the full duration of the project. In this case, Mattias will take over.

Success Criterion

- To have written a data logging application for Android that records GPS location at a minimum frequency of 1Hz and accelerations in three dimensions at a minimum frequency of 10Hz, along with the time.

- To have collected representative GPS and accelerometer data with mode labels for the following transportation modes: bicycle, bus, car, train and walk. This may be just in and around Cambridge or include other locations.

- To have implemented two or more transportation mode inference algorithms, ran them on the data collected and compared quantitatively the accuracy of the inferences made.

- (Extension) to have implemented and evaluated the accuracy of a third inference algorithm.

- (Extension) to have integrated whichever of the compared algorithms is determined to be most suitable with the codebase of the UROP project and implemented functionality to give users an estimate of their personal transport energy footprint over time.

Timetable and Milestones

Preliminaries **Oct 7th - 20th (Mich. w. 1-2, project w. 1-2)**
Determining project scope, writing proposal.
Milestones: Proposal completed and submitted.

Test data application **Oct 21st - Nov 3rd (Mich. w. 3-4, project w. 3-4)**
This will record GPS location and accelerometer data.
Milestones: Tracer application complete.

Collecting test data **Nov 4th - 17th (Mich. w. 5-6, project w. 5-6)**
As much data as can be will be collected during term time, but it may be necessary to collect more during the Michaelmas vacation, particularly for car and train journeys. Volunteer data collection will run parallel to other tasks.
Milestones: Sample data collected.

Implementing common code **Nov 18th - Dec 1st (Mich. w. 7-8, project w. 7-8)**
This will include framework type code (e.g. creating a skeleton pipeline, in which the algorithm specific inference modules are placed), as well as code for extraction and preprocessing of features.
Milestones: Theory fully understood and program structure detailed.

Implementing inference algorithms **Dec 2nd - 15th (Mich. Vac. w. 1-2, project w. 9-10)**
Likely to be [1] and [2].
Milestones: Two inference algorithms implemented

Testing inference algorithms **Dec 16th -29th (Mich. Vac. w. 3-4, project w. 10-11)**
Testing itself should be quick in that the data will have already been collected; reacting to the test results (i.e. fixing bugs) may take longer.
Milestones: Comparison of algorithms completed

Extension application **Dec 30th - Jan 12th (Mich Vac w. 5-6, project w. 12-13)**
This will vary in scope depending on time remaining.
Milestones: Inference system integrated with UROP codebase

Extension algorithm **Jan 13th - 26th (Mich. Vac. w. 7 - Lent w. 1, project w. 14-15)**
This will vary in scope depending on time remaining.
Milestones: An additional algorithm implemented and compared

Evaluation **Jan 27th - Feb 9th (Lent w. 2-3, project w. 16-17)**
This will be an evaluation of two things:

1. The algorithms will be compared, and the product of any extension activities will be evaluated.
2. The success of the project in terms of time management and achieving goals will be evaluated.

Milestones: Progress report written and submitted (Fri 4 Feb), progress presentation written and Introduction, Preparation, Implementation and Evaluation chapters completed

Writing the Dissertation **Feb 10th - March 23rd (Lent w. 4-5, project w. 18-19)**
It is my intention to write this in parallel to working on the actual project so a relatively small amount of time will be required to finalise it.
Milestones: Conclusion written, Dissertation complete.

Overflow **Feb 10th - March 23rd (Lent w. 6 - Easter w. 3, project w. 20-32)**
A buffer zone between the final timetabled task and the deadline, in anticipation of exams.
Milestone: Submission of Dissertation (Fri 20 May)

Bibliography

[1] Reddy et al. Determining Transportation Mode On Mobile Phones. ISWC '08: IEEE International Symposium on Wearable Computers (2008) 90, 91, 93

[2] Zheng et al. Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web. WWW '08: Proceeding of the 17th international conference on World Wide Web (2008) 7, 8, 14, 36, 90, 91, 93

[3] Zheng et al. Understanding mobility based on GPS data. UbiComp '08: Proceedings of the 10th International Conference on Ubiquitous Computing (2008) 7, 8, 14, 36, 90, 91

[4] http://code.google.com/p/android/issues/detail?id=3708 90

[5] http://developer.android.com/resources/dashboard/platform-versions.html 90

[6] http://stackoverflow.com/questions/2143102/accelerometer-stops-delivering-samples-when-the-screen-is-off-on-droid-nexus-one 90