SmartCity-Hackathon

Baldán Lozano, Francisco Javier
Fernández Basso, Carlos Jesús
García Gil, Diego Jesús

Problem


1020
1010
1030
1060
1050
1040
Sierras Subbéticas
Alcalá la Real
Piñar
Lucena
Rute
Guadix
Parque Natural Sierra de Baza
Albolote
Granada
Huétor Tájar
Loja
La Calahorra
Sierra Nevada
Parque Nacional de Sierra Nevada
Antequera
Villanueva del Trabuco
Alhama de Granada
Láujar de Andarax
Colmenar
Parque Natural Sierras de Tejeda, Almijara y...
Berja
Vélez-Málaga
Torre del Mar
Nerja
Almuñécar
Motril
Castell de Ferro
Adra
El Ejido
Roquetas de Mar
Málaga

Data Preprocessing

- 160.192 steps
- Added detection time variable
- Merged data into one single dataset
- 14 attributes

Prediction Application

Purpose of this application

This application focuses **on obtaining data in time series format** in order to be able to predict the average time each vehicle detection by each sensor. If **high detection times** are obtained they can be **associated with the presence of traffic jams**.

Data Format

1. **Obtain the data time sampled / accumulated every second detection**.

A time series is defined as a set of data recorded at equidistant time intervals. So the natural first step of this application is to transform the original dataset for the average detection time sampled every second (from this first transformation to obtain **new datasets sampled at different intervals** is facilitated, such as **1, 10 , 15, 30, 45 and 60 minutes**)

Prediction objectives

2. **Getting predictions one-step-ahead detection time**.

In this case it was decided to make **predictions of detection times in each sensor** in order to associate the high detection times recorded with the presence of **traffic jams**.

In this case we have made **predictions for 10, 15, 30, 45 and 60 minutes**. With the initial data and processing on those obtained in the last two months have been obtained runtimes less than 2 minutes.

Prediction objectives (2)

This step generates an output file for prediction with **3 columns**, which contain the time axis in **UNIX time**, the **actual data** used in training the model prediction and the set of output data contains the actual data to the penultimate prediction point and **the prediction of the last point** (test), respectively.

Additional comments

If you run the experiment on a **Linux system**, you can **take advantage of parallelization characteristics** implemented in the code by increasing the number of cores in these types of functions:

mclapply (x, function (x) {}, mc.cores = getOption ("mc.cores", 1))

Future works

The next step nature of this work is to **associate a timeout threshold** from which a jam would be detected in the sensor. This threshold would be **defined by the data and the characteristics of the road where the sensor is located**.

Route Analysis

- Clean the data:
  - Date to UNIX time
  - Order dataset by time (time series)
  - Remove ids with frequency < 2
  - Add hour, minute, second, day and month variables
  - Check for vehicles that have passed for at least two different sensors, the same day, with a difference of 3 hours max.
  - 3079 vehicles

Route Analysis (2)

- Checks for a given id if it has been detected previously by other sensor, if that's the case, it calculates the average speed
- Results are most used routes and average speed
- Example for route 1010 - 1020:

idDisp	Avg. Speed	At
7D03FA3475165187CAC76AC5FAB84F6ABFEFA2CA	116.1	1444403586
BE8D87EF037736DED4F550E4E23A20A68B05A8C9	102.83	1444404979
183D25070470F6B27A27D3A50BC08BBF52BE3E24	81.38	1444407669