Resource Allocation for a Bike Sharing service using Bayesian Networks

Carlo Longhi
carlo.longhi@studio.unibo.it

Alma Mater Studiorum Università di Bologna

May 21, 2022

Abstract

Bike sharing services are a useful and environmentally friendly mobility solution. With their gradual introduction and growing number of users, the natural problem of resource allocation [1] arises: how many bikes should be available in a certain period? The factors involved in this problem are usually too complex to analyse by a human, but data comes to help. In this work we use a Bayesian Network to get more useful insights on the usage data of a bike sharing service and demonstrate that Bayesian Networks are a powerful tool to deal with this kind of problems.

Contents

1 Introduction

Bike sharing services produce a huge amount of data that could be leveraged in order to better understand which are the factors that influence the number of rides. Getting a deeper understanding of these factors is crucial to deal with the problem of resource allocation, a critical problem for any sharing service. Solving this problem and gaining a better understanding of its users needs are crucial tasks to improve the sharing service.

In this work we tried to design a Bayesian Network capable of analyzing the data provided in a Kaggle's database [2], representing bike sharing data for the city of London. The code, making use of the library *pgmpy* [3] is available in a public GitHub repository

2 Data and Preprocessing

The dataset [2] contains 17414 entries, each one of them representing the bike sharing data for the city of London at an interval of 1 hour. The data was collected between the 1st of April 2015 and the 1st of March 2017.

For each entry, the following features are reported:

- **timestamp**: time of the collected entry
- **cnt**: amount of bike sharing rides initiated in the specified interval
- **t1**: registered temperature measured in Celsius degrees
- **t2**: perceived temperature measured in Celsius degrees
- **hum**: registered humidity in percentage
- **wind_speed**: wind speed measured as km/h
- **weather_code**: weather category: 1=Clear, 2=Scattered Clouds, 3=Broken Clouds, 4=Cloudy, 7=Rain, 10=Rain with Thunderstorm, 26=Snowfall
- **is_holiday**: True if the day of the entry is an holiday, False otherwise
- **is_weekend**: True if the day of the entry is in the weekend, False otherwise
- **season**: season of the year: 0=Spring, 1=Summer, 2=Fall, 3=Winter

The first step, before trying to construct the network, is to pre-process the data in order to:

1. Discretize continuous features, since pmgpy does not support continuous variables
2. Reduce the range of values of some features to decrease the quantity of RAM required by *pgmpy* to deal with our network.

In particular, from the *timestamp*, we decide to only keep the hour. This choice has been dictated by the fact that information about the day is already expressed by the *is_weekend* and *is_holiday* features and the data is collected at intervals of one hour. Moreover, since we want to reduce the range of our variables, we do not keep the *hour* as is but assign it to one of the four time intervals of the day:

- from 0.00 to 6.59 is given the value 0
- from 7.00 to 12.59 is given the value 1
- from 13.00 to 18.59 is given the value 2
- from 19.00 to 23.59 is given the value 3

Finally, we remap the *weather_code* to {0,..,6} values and transform *t1, t2, hum, wind_speed, cnt* to quartiles, effectively reducing their range of values to {0,...,3}. Now, for example, a value of 3 for *cnt* represents a peak in the number of bike sharing rides.

3 The Network

Bayesian networks are a type of probabilistic graphical model that uses Bayesian inference for probability computations. Bayesian networks represent random variables with nodes and conditional dependencies by edges in a directed graph [4].

To build a Bayesian Network, the first step is to define the network structure by means of a set of random variables and their dependencies. Then, we have to define, for each node, the conditional probability distributions (CPD) quantifying the influence of the parents on that node.

The library *pgmpy* gives the opportunity of defining both the network structure and CPDs manually or automatically.

3.1 Manual Definition

To define a Bayesian Network we first need to summarise our knowledge of the features in the database and their relations.

We know that the number of rides (*cnt*), can be affected by the perceived temperature (*t2*), the wind speed (*wind_speed*), the weather (*weather_code*), if it is an holiday (*is_holiday*) or if it is in the weekend (is_weekend) and by the hour of the day (*hour*). Moreover, we know that the perceived temperature (*t2*) is affected by the actual temperature (*t1*), humidity (*hum*) and wind speed (*wind_speed*) [5] and the actual temperature (*t1*) is affected by the season (*season*). Finally, the humidity level (*hum*) should be affected by the wind speed (*wind_speed*) and the weather (*weather_code*).

The network obtained by expressing these dependencies is the one in Fig. 1.

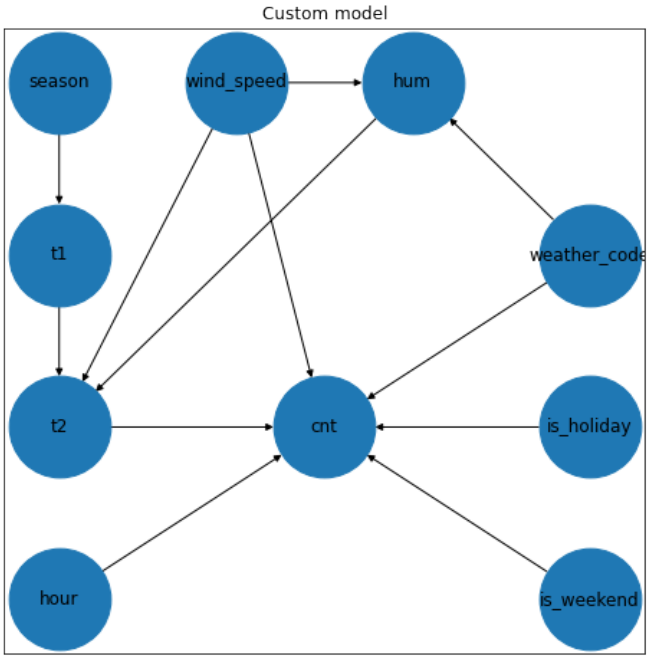

Figure 1:

3.2 Learning the structure

To study how well our model captures the dependencies between the features in the database, it could be useful to learn the structure of the network directly from the data. Then, we can compare the model defined manually with the one learned from the data.

The two main approaches to the problem of learning the structure of a Bayesian Network are: *score-based* and *constraint-based*. In this work, we focus on *score-based algorithms*. The objective of these types of algorithms is the following:

$$\underset{G}{argmax}(score(G, D))$$

where $score(G, D)$ is a given scoring function which measures how much the graph G fits the database D. The scoring function we use is the *Bayesian Dirichlet equivalent uniform function* (BDeu), that, assuming a prior probability over all possible direct acyclic graphs, computes the posterior probability of a certain graph given the data [6]. The last ingredient is the search algorithm

used to find the local maximum. In this work, we make use of the *Hill Climb Search* algorithm, in the implementation provided by *pgmpy*.

The network obtained by this search process is shown in Fig.2.

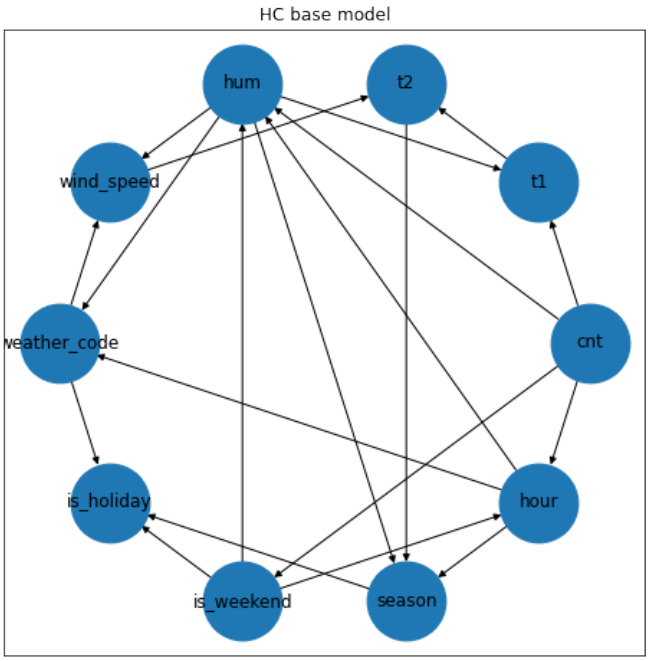

Figure 2:

Unfortunately, this network presents some problems:

- some variables depend on *cnt*, but we know that this can not be the case in reality
- the dependency humidity → perceived temperature, that we know to be true, is missing

These problems can be easily solved by introducing some prior knowledge in the network. Using *pgmpy* we constrain *cnt* to not influence other variables and we enforce the humidity → perceived temperature dependency.

The resulting network is shown in Fig.3.

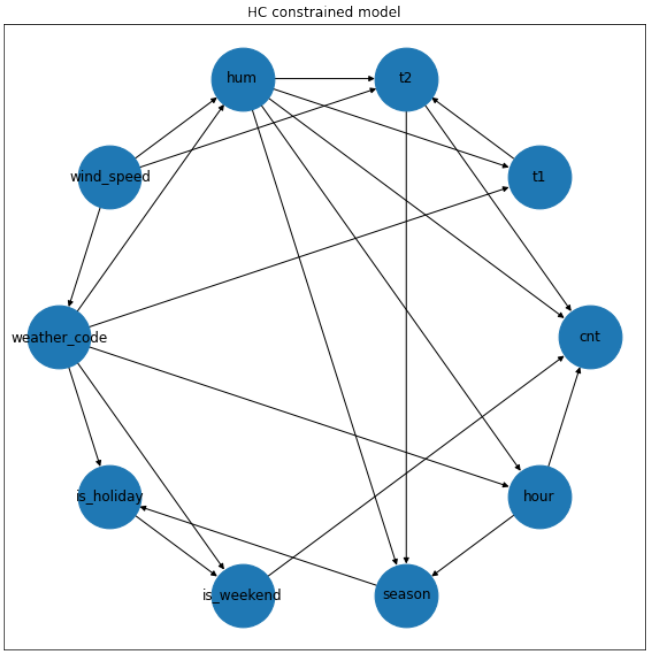

Figure 3:

By comparing different structure scoring functions on the manually defined network and the automatically defined one , we can see a clear advantage in using the latter:

Table 1: Structure scores for the two defined networks

	BDeu	K2	BIC
Custom Model	-173576	-169698	-192636
Learned Model	-161976	-161588	-163640

Our network still presents some problems: the variables *hour, season, is_holiday* and *is_weekend* can not depend on other variables. We further introduce constraints to solve it and show the result in Fig. 4

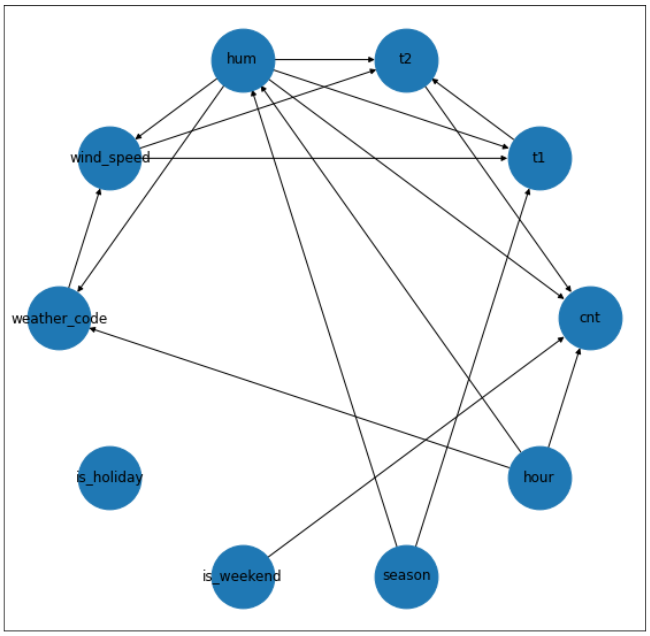

Figure 4:

We compute the scores for this final model and obtain:

Table 2: Structure scores for the final network

	BDeu	K2	BIC
Final Model	-160604	-160321	-162090

This final network obtains slightly better results than the previous one and, for this reason, seems to represent our knowledge about the problem at best.

Studying this model, we can see how it represents some dependencies which we missed to consider, as for example weather $\rightarrow$ wind speed and season $\rightarrow$ humidity. It also dropped some dependencies that, instead, we included in our original model as holiday $\rightarrow$ number of rides. We also note that *is_holiday* can now be dropped, since it does not influence any other variable.

4 Learning the Parameters

Having defined a network structure, it is possible to learn the network's parameters. To do so, we can use the *Maximum Likelihood Estimation* algorithm (MLE). Given a Bayesian Network, its parameters $\theta = \{\theta_1, ..., \theta_n\}$ with (θ_i the conditional probability distribution of the variable X_i) and a global likelihood function $\mathcal{L}(\theta|\mathcal{D})$ defined as:

$$\mathcal{L}(\theta|\mathcal{D}) = \prod_i l_i(\theta_i|\mathcal{D})$$

where $l_i(\theta_i|\mathcal{D})$ is the local conditional likelihood associated with variable X_i and defined as.

$$l_i(\theta_i|\mathcal{D}) = \prod_m P(x_i = m|Parents(X_i) = m, \theta_i)$$

The goal of the maximum likelihood estimation is to solve the optimization problem

$$\hat{\theta} = \underset{\theta}{argmax}\mathcal{L}(\theta|\mathcal{D})$$

Using this approach, we have estimated the parameters of the manually defined model and the final model obtained learning the structure with all the constraints.

5 Probabilistic Reasoning

Having defined our networks structure and their parameters, we want to test their ability to represent our problem. In order to do so, we can try to perform some inferences. We test the manually defined model and the learned one separately.

5.1 Manually Defined Model

In this model the number of sharing rides (cnt, range={0,...,3}) is influenced by the following features:

- wind_speed, range={0,...,3}
- perceived temperature (t2), range={0,...,3}
- hour, range={0,...,3}
- is_weekend, range={True, False}
- is_holiday, range={True, False}
- weather_code, range={0,...,6}

We want to study in which way they influence the probability of a peak of bike sharing rides to occur. To do so, we perform a series of queries:

$$p(cnt = 3|wind\_speed = w)$$

$$p(cnt = 3|t2 = t)$$

$$p(cnt = 3|hour = h)$$

$$p(cnt = 3|is\_weekend = e)$$

$$p(cnt = 3|is\_holiday = o)$$

$$p(cnt = 3|weather\_code = c)$$

with $w, t, h \in \{0, ..., 3\}$, $e, o \in \{True, False\}$ and $c \in \{0, ..., 6\}$. We query our model for values of *cnt* equals to 3 because we are interested in understanding when do peaks of sharing rides occur. Results are shown in Fig.5.

Figure 5: Custom model

Based on these results, we can already make some observations:

- the number of bike sharing rides is higher during the morning and afternoon, between 7.00 and 19.00
- the perceived temperature affects in a positive way the number of rides
- the wind speed has a slight negative effect on the number of rides

- the number of rides slightly drops during weekends and holidays, compared to the other days

To obtain more insights on these observations, we now want to study how the other features influence *cnt* if we fix *is_weekend*. The queries designed are of the type:

$$P(cnt|is\_weekend = False, t2 = t, hour = h, weather\_code = c)$$

Results are reported in Fig.6.

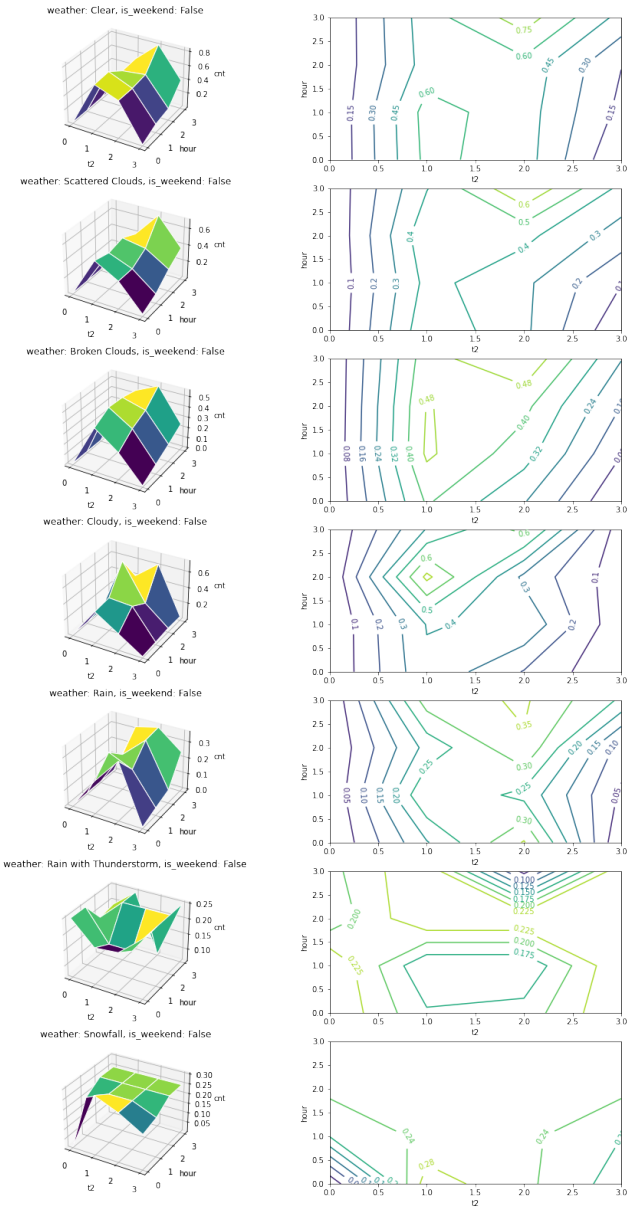
weather: Clear, is_weekend: False
weather: Scattered Clouds, is_weekend: False
weather: Broken Clouds, is_weekend: False
weather: Cloudy, is_weekend: False
weather: Rain, is_weekend: False
weather: Rain with Thunderstorm, is_weekend: False
weather: Snowfall, is_weekend: False
t2
hour
cnt

Figure 6:

From the plots, we can observe that, during week days, bike sharing peaks are more likely to occur when the perceived temperature is high and in the evening. We can also see how, when the it is raining or snowing, the probability of a peak to occur is much lower, compared to more favorable weathers.

5.2 Learned Model

When using the learned model, the number of bike sharing rides (cnt, range={0,...,3}) is affected by the following features:

- t2, range={0,...,3}
- hum, range={0,...,3}
- is_weekend, range={True, False}
- hour, range={0,...,3}

To study how they influence the probability of a peak of bike sharing rides to happen we perform a series of queries of the type:

$$p(cnt = 3|hum = m)$$

$$p(cnt = 3|t2 = t)$$

$$p(cnt = 3|hour = h)$$

$$p(cnt = 3|is\_weekend = e)$$

with $m, t, h \in \{0, ..., 3\}$ and $e \in \{True, False\}$. Results are shown in Fig.7.

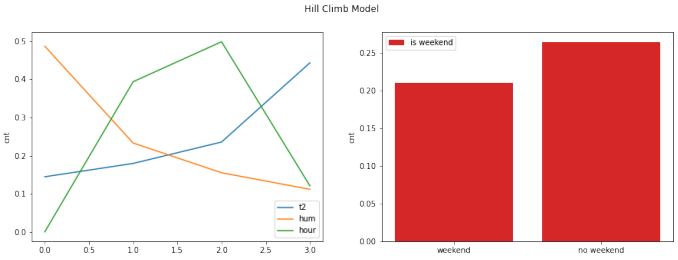

Figure 7:

From these results, it is straightforward to conclude that:

- the highest number of rides is initiated between 7.00 and 19.00
- higher perceived temperatures correspond to an higher number of bike rides
- higher humidity values correspond to a lower number of bike rides

- the number of bike rides drop during the weekend

As for the manually defined model, we now want to study how the other features influence *cnt* if we fix *is_weekend*. The queries designed are of the type:

$$P(cnt|is\_weekend = False, t2 = t, hour = h, hum = m)$$

Results are reported in Fig.8.

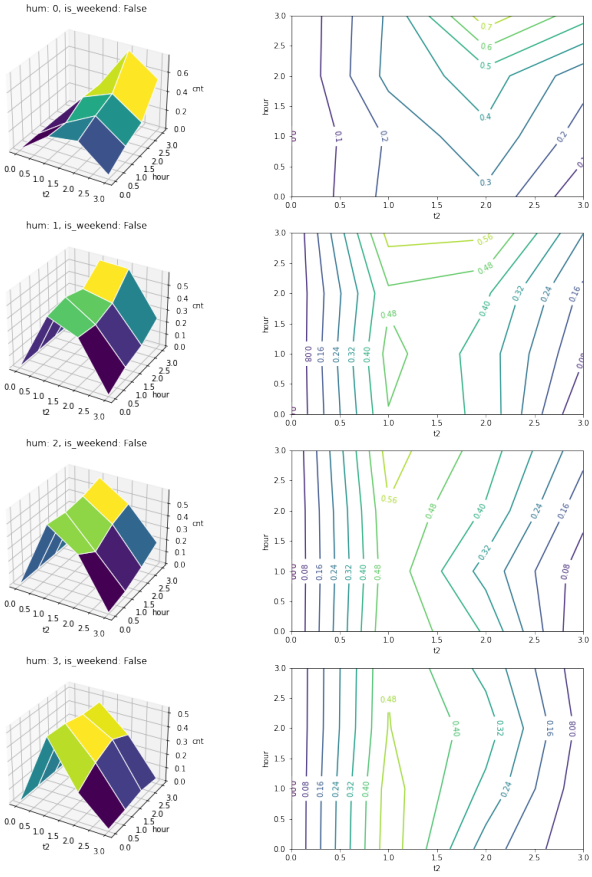

Figure 8:

From the results of these queries we can observe how bike sharing peaks are

most probable to occur between the 7.00 and 19.00, the hours associated with the morning and afternoon. As we expected, higher perceived temperatures, influence positively the number of bike sharing peaks.

6 Conclusions

Bayesian Networks are a powerful tool, able to model complex probabilistic relationships and dependencies. Their application could improve decision making processes in a variety of domains. The library *pgmpy* is a useful instrument to work with and perform inference on Bayesian Networks in an easy and intuitive way. Moreover, the possibility of introducing constraints on the dependencies in a network, makes it very straightforward to introduce background knowledge of the problem when learning from data.

In this work, we have shown how can we apply a Bayesian Network to a problem occurring often in the real world. We were able to both model our problem manually and automatically from a database. Finally, we used our model to perform inference on the database to gain interesting insights on the data.

References

[1] Leonardo Caggiani, Rosalia Camporeale, Mario Marinelli, Michele Ottomanelli: User satisfaction based model for resource allocation in bike-sharing systems, Transport Policy Vol. 80, 2019

[2] London bike sharing dataset, available at https://www.kaggle.com/datasets/hmavrodiev/london-bike-sharing-dataset

[3] pgmpy, Python library for the implementation of Bayesian Networks, available at https://pgmpy.org/

[4] Introduction to Bayesian Networks, Devin Soni, Towards Data Science, 2018

[5] Apparent Temperature, Wikipedia, `https://en.wikipedia.org/wiki/Apparent_temperature`

[6] D. Heckerman, D. Geiger, D. M. Chickering, "Learning bayesian networks: The combination of knowledge and statistical data", Machine Learning, vol. 20, no. 3, pp. 197-243, 1995