Prediction System for Bike-sharing program

Himanshu Mendhe
Rochester Institute of Technology
hpm7152@rit.edu

P. Shashank Adavi
Rochester Institute of Technology
pa8546@rit.edu

Aradhya Mehta
Rochester Institute of Technology
adm9401@rit.edu

ABSTRACT

This prediction system will provide an estimate of the number of bikes which might be required during a certain day or hour and will also be based on prior usage of bikes under similar conditions of time and weather. This project also will make use of a few data mining algorithms in order to assist with the count for predicting the number of bikes shared at a certain hour. The project aims to primarily compare historical usage patterns of the bikes based on the weather data in order to predict the bike count in the city of Washington D.C.

1. INTRODUCTION

Using a bicycle is the cleanest and healthiest form of transport available to mankind. A lot of organizations, universities, cities and towns have bike-sharing programs where bikes are pooled and then loaned over a period of time. One of the issues such initiatives face is managing their bicycling equipment so that they may account for peak-times, slack times, according to the user's behavior patterns. These patterns can be influenced by time of the day or year or external conditions like weather. This project will try to develop a prediction system which would take into account prior usage statistics of various users. It will use data mining algorithms to predict the same. The dataset which will be used to model such systems will be obtained from kaggle.com[3].

2. MOTIVATION

Bike sharing systems are basically systems which function through a central command, known as a kiosk location. These kiosk locations control every process in the system such as, becoming a member, renting out the bikes and also returning them. The kiosk location also functions in a way that if a bike has been rented from a location, say X, it can be returned to any other kiosk location in the vicinity, say Y. The data that is generated by these bike-sharing systems is exactly the kind of data which is beneficial to our project because many attributes such as travel duration, time elapsed are easily recorded and thus these systems act as a anticipating network, whose results can be used to study patterns of vehicular movement in the city.

3. DATA DESCRIPTION

The dataset that we will be making use of is the bike sharing demand dataset from kaggle.com. The dataset contains data which is related to the conditions under which bikes are shared from a certain time to time. Our project involves making a correct prediction of the number of bikes that might be shared on an hourly basis based on the data that is available with us. For that, the attributes such as 'casual', 'registered', 'count' are not the ones which will help us in predicting. The rest of the attributes will play an important part in predicting the count of the bikes during a certain hour.
Various taxonomies are involved in the data which are as follows:

- datetime - hourly date + timestamp
- season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
- holiday - whether the day is considered a holiday
- workingday - whether the day is neither a weekend nor holiday
- weather - 1: clear, Few clouds, Partly cloudy 2: mist + cloudy, mist + broken clouds, mist + few clouds, mist 3: light snow, light rain + thunderstorm +scattered clouds, light rain + scattered clouds 4: heavy rain + ice pallets + thunderstorm + mist, snow + fog
- temp - temperature in celsius
- atemp - 'feels like' temperature in celsius
- humidity - relative humidity
- windspeed - wind speed
- casual - number of non-registered user rentals initiated
- registered - number of registered user rentals initiated
- count - number of total rentals

.

4. ARCHITECTURE

The different dataset files are of *.csv format. The below image shows a few rows of the dataset. As mentioned earlier, we shall not be requiring all the attributes mentioned in the dataset below, i.e. to predict the total count of bikes for a particular hour, we probably might not need the last three attributes, i.e. 'casual', 'registered' and 'count'.

datetime	season	holiday	workingd	weather	temp	atemp	humidity	windspee	casual	registerec	count
01/01/11 00:00	1	0	0	1	9.84	14.395	81	0	3	13	16
01/01/11 01:00	1	0	0	1	9.02	13.635	80	0	8	32	40
01/01/11 02:00	1	0	0	1	9.02	13.635	80	0	5	27	32
01/01/11 03:00	1	0	0	1	9.84	14.395	75	0	3	10	13
01/01/11 04:00	1	0	0	1	9.84	14.395	75	0	0	1	1
01/01/11 05:00	1	0	0	2	9.84	12.88	75	6.0032	0	1	1
01/01/11 06:00	1	0	0	1	9.02	13.635	80	0	2	0	2
01/01/11 07:00	1	0	0	1	8.2	12.88	86	0	1	2	3
01/01/11 08:00	1	0	0	1	9.84	14.395	75	0	1	7	8
01/01/11 09:00	1	0	0	1	13.12	17.425	76	0	8	6	14
01/01/11 10:00	1	0	0	1	15.58	19.695	76	16.9979	12	24	36
01/01/11 11:00	1	0	0	1	14.76	16.665	81	19.0012	26	30	56
01/01/11 12:00	1	0	0	1	17.22	21.21	77	19.0012	29	55	84
01/01/11 13:00	1	0	0	2	18.86	22.725	72	19.9995	47	47	94
01/01/11 14:00	1	0	0	2	18.86	22.725	72	19.0012	35	71	106
01/01/11 15:00	1	0	0	2	18.04	21.97	77	19.9995	40	70	110
01/01/11 16:00	1	0	0	2	17.22	21.21	82	19.9995	41	52	93
01/01/11 17:00	1	0	0	2	18.04	21.97	82	19.0012	15	52	67
01/01/11 18:00	1	0	0	3	17.22	21.21	88	16.9979	9	26	35
01/01/11 19:00	1	0	0	3	17.22	21.21	88	16.9979	6	31	37
01/01/11 20:00	1	0	0	2	16.4	20.455	87	16.9979	11	25	36

Figure 1: Sample Data

The above is a sample format of the dataset. As it is difficult to represent all the fields on the report, we have added the train.csv, test.csv files along with this report in the zip file.

5. DESIGN

The Data Flow of our project is fairly simple and straightforward. We are required to indulge in very limited amount of data-cleaning. We will be required to split the *datetime* attribute into two more attributes, *date* and *time*. We might be also required to discretize some of our numeric data. For example, attributes like *temp*, *atemp*, *humidity*, *windspeed* can all be discretized into numeric intervals. Rest of the dataflow will involve Data Mining.

5.1 Data Mining using Predictive model

The main aim of our project is to predict how many bikes will be required at a given time and weather conditions based on the similar conditions. Hence, The Data Mining algorithms we use has to involve prediction. After going through many prediction techniques, we have found that the two main methods of prediction are:

1. Predicting on the basis of knowing the prior probability of an event happening.

2. Predicting by making a mathematical model which will output predictions based upon previous values.

We have decided to implement both these techniques and then do a comparative study of both. For the first part, we have implemented the Naïve Bayes algorithm. We contemplated using the Bayesian Networks, but our dataset has attributes which are fairly independent of each other and Bayesian network requires some relationship between its attributes. The only glaring relationship between attributes which can be observed in our dataset is the one between *workingday* and *holiday*. This can be removed in the data-cleaning stage. For the second part, we have implemented the Multilayer Perceptron algorithm (MLP) for Neural Networks as a predictor.

6. IMPLEMENTATION

In the previous submissions, we had made the data dictionary for the dataset and the four steps that were to be used in building a dataset were successfully implemented. We calculated the descriptive statistics for the numeric attributes and also the range for those attributes. In, the second phase of the project, we did the very little of scope of data cleaning that was to be performed in the dataset. Also, once the dataset was cleaned, we made the data available for the Naïve Bayes algorithm to be performed in WEKA just so we could get an initial idea of the results and the expected performance of the algorithm. [1] [2]

6.1 Multilayer Perceptron

We implemented this algorithm in JAVA by making use of the WEKA libraries. For the Multilayer perceptron (MLP) algorithm, the algorithm makes use of backpropagation to classify its instances. Since the dataset has been discretized before the MLP has been implemented, the nodes in the network are all sigmoid. There are a few parameters in this algorithm which are more important for a better and successful classification of instances. Some of them are:

1. Learning rate (-L): This parameter simply trains and controls the size of the weight. Learning rate by default is set to 0.3 and is values lies between [0-1].

2. Momentum (-M): This term simply adds a certain amount of the previous weight generated to the current weight being computed. Learning rate and momentum are in some ways inversely proportional, so if the learning rate is high, we need to make sure that the momentum is low and vice-versa. Having a high value of both the learning rate and momentum will result in the weight going beyond the minimum.

3. Number of Epochs (-N): This value is used to determine as to when the training of the classification has to stop whenever the number of iterations go past the value of number of epochs. When the error value is at its minimum, the value in -N represents the maximum number of iterations that are possible.

6.2 Naïve Bayes

A classification algorithm that we have made use of is the Naïve Bayes algorithm, which similar to the MLP algorithm, has been implemented in JAVA by making use of the WEKA libraries. The values that are generated when using the Naïve Bayes are chosen based on the raining data that we have analyzed. For the Naïve Bayes though, we did face a few problems when dealing with our dataset. Our dataset contains attributes which are only numeric after cleaning, i.e. all the previous nominal attribute weren't necessarily required for the project. To implement Naïve Bayes, all the attributes have to be nominal, and thus we discretized it

using WEKA. That is the phase actually where we encountered the most difficulties, because all the attributes in the dataset were being converted to nominal apart from the last attribute in the queue. We tried various ways of trying to get the attribute converted to nominal. Finally, a fairly basic approach solved the problem, and that was to change the position of the nominal attribute with that numeric attribute and thus the last attribute now became nominal and we were then able to proceed with the Naïve Bayes algorithm.

7. APPLICATION

For the application that we have made for the project, we have used WindowBuilder in Eclipse. WindowBuilder is basically a JAVA GUI designer, and consists of the SWT and Swing designer and thus creating an application is not a very arduous task. The initial page and the results page snapshot can be seen below.

8. EVALUATING BOTH MODELS

In this section, we will initially explain the results of the Naïve Bayes algorithm followed by the results obtained by the multilayer perceptron(MLP) algorithm. After the individual results for both the implementations have been explained, we will compare them through a few common characteristics that are shared by both the algorithms.

8.1 Naïve Bayes

The Naïve Bayes algorithm , when run, gives the output as a range of values, which means that the predicted count of the bikes for that particular hour lies between those values. We will now analyze the confusion matrix that has been derived by executing the train dataset. As we can see in the below snapshot, the correctly classified instances outweigh the incorrectly classified instances for almost all the classes apart from class j.

```
=== Confusion Matrix ===

   a    b    c    d    e    f    g    h    i    j   <-- classified as
4216   68    0    0    0    0    0    0    0    0 |    a = '(-inf-98.6]'
 153 1949  233    0    1    0    0    0    1    0 |    b = '(98.6-196.2]'
   1  208 1360  113    4    0    0    0    0    0 |    c = '(196.2-293.8]'
   2    4  247  723   81    9    1    0    0    0 |    d = '(293.8-391.4]'
   5    0    1  211  352   65    6    0    0    0 |    e = '(391.4-489]'
   8    0    1   12  110  254   34    0    0    0 |    f = '(489-586.6]'
   1    0    0    3    8   52  142   27    0    0 |    g = '(586.6-684.2]'
   1    0    0    0    1    8   29   62   15    0 |    h = '(684.2-781.8]'
   0    2    0    0    1    0    1   17   64    0 |    i = '(781.8-879.4]'
   0    0    0    0    0    0    0    0   16    3 |    j = '(879.4-inf)'
```

Figure 2: Naïve Bayes Confusion Matrix

The descriptive statistics also say a little about our dataset. The kappa statistic is a chance-corrected measure between the classifications and the true classes. Since our value is greater than 0 and inclining more towards 1, it means that the classifier is doing than chance, i.e. this is proof that the classifier is indeed giving satisfying results. The relative absolute error of 32% indicates the numeric prediction capability rather than classification. Thus, the error is basically a magnitude and these rates are a reflection of the same.

```
Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        9125               83.8233 %
Incorrectly Classified Instances      1761               16.1767 %
Kappa statistic                          0.7862
Mean absolute error                      0.0492
Root mean squared error                  0.1544
Relative absolute error                 32.3533 %
Root relative squared error             56.028  %
Total Number of Instances            10886
```

Figure 3: Naïve Bayes Statistics

8.2 Multilayer Perceptron

Unlike the Naïve Bayes algorithm, the Multilayer Perceptron algorithm generates a certain single value as its predicted count. To generate this count though, we used the normalization method to predict the count. Since we have normalized the dataset by a factor of 100, we make use of the basic cross-multiplication formula to predict the count. In words, the count is the product of the maximum value * value from the model divided by the normalization factor, which in this case is 100. This is how the count has been predicted. Other factors which help us in analyzing the algorithm is the correlation coefficient. The correlation coefficient is basically used to determine as to how confused a classifier is, i.e. the closer the value of the classifier to 1, the more confused the classifier is. In our case, the value of the correlation coefficient is 0.3897, which means that the classifier is not confused and the classifications have been done pretty accurately.

```
Time taken to build model: 59.66 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.3897
Mean absolute error                     19.9459
Root mean squared error                 24.7263
Relative absolute error                136.5431 %
Root relative squared error            133.356  %
Total Number of Instances            10886
```

Figure 4: MLP Statistics

8.3 Comparisons of the two models

Term	NB	MLP
Mean absolute error	0.0492	19.94
Root mean squared error	0.1544	24.72
Relative absolute error	32.35%	136.54%
Root relative squared error	56.02%	133.35%
Total number of instances	10886	10886

As can be seen from the table, for the same dataset, the error rates, i.e. the mean absolute error, root mean squared error etc. have very different and largely varying values. But as we can see, the Naïve Bayes error rates have far lower values as compared to the multilayer perceptron, which basically means that the Naïve Bayes model has worked better for the dataset that we have chosen. The root mean squared error is basically used to calculate the average magnitude of the error. This value is usually greater than the mean absolute error, and if the difference between those two errors is constantly increasing, then the variance in the individual errors is also high. In this case, again we can see that since the difference between the two errors is greeter in MLP, it means that the MLP model has higher errors individually and thus again the Naïve Bayes works better than the MLP algorithm. But as we can see in the final row, that the number of instances classified is the same for both models at 10,886, which basically means that regardless of the errors in the models, both of our models have been successfully executed. Now, according to these descriptive statistics, it is clear that the Naïve Bayes algorithm works better than the multilayer perceptron algorithm. But, according to our classification, the above theory does not hold true. One of the reasons is that the descriptive statistics that we observe, such as the error rates etc. have been used on the training data. We classify our model though, based on the test data. Thus, according to our analysis it looks like the model has overfitted on the training data and thus produces the results in favour of Naïve Bayes, which is actually not true according to our implementation.

9. REFERENCES

[1] Prediction system for bike sharing demand in phase 1.

[2] Prediction system for bike sharing demand in phase 2.

[3] KagBS. Data - bike sharing demand | kaggle @ONLINE, May 2014.