bike_sharing_challenge

February 28, 2020

George Tzannetos

This is a comprehensive report of the data analysis and modeling of the San Francisco bike share data. The goal is to provide a model that predicts the net change of bikes in each station for the next hour.

1 Data Exploration

The first part is to import the data and explore them. That means understand the data, clean then, find missing values, and understand which one could be important for our case.

In this case, we have 3 csv files, weather, station, trip. The idea is from the trips data to extract how many trips have started and stopped in each station every one hour.

Weather Data

We can see that the weather dataframe contains a lot of weather information. One important remark is that the weather info are daily and not hourly,as we want our predictive model to be. We have 365 uniques dates in our date range and the weather data contain 1825 entries. That means that we have weather data for each one of the 5 zip locations every day. In the weather data, the Events column contains multiple nan values, but it can be safely assumed that no event happened in that day, thus it was a day with a normal weather. Moreover, I fill some nan values of the weather with the mean value of the column.

Station Data

For the station csv, we know that some of the stations are moved and changed. However there is no information about the installation time of each station, thus we consider each station independant.

Trip Data

Contains all the trips, the start station and time and end station and end time.

We can proceed now with starting to combining the dataframes and generating the training data.

The idea is to create a dataframe for the date range 01/09/2014 until 31/08/2015 with one row for each hour of the day. I should do the same for each station ID.

I create a date dataframe containing each day and hour from 01/09/2014 00 till 01/09/2015 00. The weekdays column is also added, showing 0 –> Monday and 6 –> Sunday. Also Business day and Holidays are added as features.

Next part would be to match the location the stations with the zip location of the weather data. The matching is the following: {94107: 'San Francisco', 94063: 'Redwood City', 94301: 'Palo Alto', 94041: 'Mountain View', 95113: 'San Jose'}

After that, important for our case is to count how many trips started and ended for each station every hour, e.g give me the trips started at station 2 at 01/09/2014 from 1:00 till 1:59. Then it would be possible to find the net change at this specific station for the following hour by subtracting (trips ended - trips started). This will be also the target variable, i.e the difference between the columns Stop_CountTrips and Start_CountTrips.

Another addition to the training data would be to add the weather conditions for each station each day(since we are missing hourly data). This is done by merging the weather data with the merged dataframe that was created.

To get more insight into the features, I calculated the statistics and the feature correlation. For example we can see that business day and weekdays are highly corellated. Also Zip with stationID and the temperatures with the dew points are highly correlated features. Moveover, I find which features are mostly corellated with our target variable.

At this point, a first table with training data is obtained, that could provide the required predictive model. The table is further analysed and also specific columns will be chosen in the modeling phase.

2 Modelling Approach

The problem is a regression one. Thus, an extreme gradient boosting regressor was chosen to learn and model the above data. The XGBoost model is tuned, using the Grid search approach of sklearn and also cross validation is performed to better estimate the model's performance in an unseen dataset.

Regarding our features, first the Time column is dropped, since each specific date occurs only once in our dataset. Moreover, start and stop trip columns should be removed, because they contain all the info about the target variable(the model would easily learn to subtract the 2 columns and achieve accurate results). Since the data are time series, it should be obvious to split the data to past for train and future for prediction. However, there is no future information linked to our estimations, since each prediction is based on;y on the current status of each station.

After I define my feature table, the first part is to split the data to train and validation. As mentioned above, there are two ways to do it. The first one would be to split the data in a sequential order, since the data are a time series. However, in the features, there is not information about the future. Therefore, there is no future leak and I could split the data randomly without a problem. After that, I use GridSearch method to get the best parameters of the model.

I start with a single training and use the best combination that found using GridSearch. The GridSearch provides us with a good combination of parameters. Then, additional experimentation was done to get an insight of the model parameters by tweaking them. Finally, I apply 3-fold cross validation, to get an good indication of how the model will perform on unseen data. The first root mean squared error(RMSE) obtained is: **1.043**.

After training, the importance of the features is also plotted, along with the gain across the splits

each feature is used in. This provides us with a better understanding of the model, and what plays an important role on the decisions.

At that point, I proceeded with some feature selection.

Feature Selection By running some experiments, the following conclusions, about which features improve the performance, are made:

- with Dock_count
- without Zip
- with PrecipitationIN
- without WindSpeed
- without Holidays
- with Business_days
- With Temperature

We can see that it is better to remove highly corellated columns from our features. Also, a high correlated feature with the target variable, like WindDir does not mean that could provide meaningful information in predicting the net rate, and may lead to overfitting. Thus, the models should perform better after removing them.

However, no decrease in the validation RMSE was shown while removing the features based on the above list. The performance of the model didn't improve, and it even got worse. Moreover, it was much easier to **overfit**. The RMSE this time **0.1078**

Improvement of the dataset using external data I decided to extent the original dataset by adding a new feature. In Kaggle another cvs file containing status data was found. Here I will try to extract information and see if by using that, our model can be improved. I follow the feature analysis made above, so keep only the features that seem relevant, and the add this new feature. The data source can be found as status.csv here: https://www.kaggle.com/benhamner/sf-bay-area-bike-share.

The status data is cleaned to include only information relevant for our date range. This step is included in the source code. After I clean the data, we can see that the file contains the availability of bikes and docks at each station every one minute. The plan for this dataframe is to extract and add in my dataset two columns containing the available bikes and free dock stations at the beginning of every hour. Thus I have to groupby the info contained in the statusDf on hour,time and stationID. Then I will keep only the last value and use it as information for the next hour, meaning that this value will be the number of available bikes at the beginning of the next hour.

The features chosen here are: 'Weekday', 'Business_day', 'Station_ID', 'Dock_count', 'Month', 'Hours', 'MeanTemperatureF', 'PrecipitationIn', 'Events' , 'bikes_available', 'Net_Rate'.

After that, I repeated the same steps as before, in the modeling to train our model. I use exactly the same parameters as I used for training the previous model above. These parameters were shown to give the best results, thus it can safely stated, by comparing the 2 models RMSE, that the added feature of available_bikes was important in improving the model's perfomance. For this case, not all the weather data were used, and the available bikes were added. The RMSE achieved this time is: **1.033**.

In the end, another model of Gradient Boosting was used. It is similar to XGBoost, however XGBoost is better known to handle missing data and harder to overfit. Training this model, I obtained an RMSE on 5-fold CV: **1.195**, which is much worse than the XGBoost model.

3 Performance Analysis

In this section we will discuss the different results, that were achieved at the modelling section. A small explanation will be given on each one of the models and different versions of the train dataset. I present all the results in the table below:

DatasetVersion	Model	CrossVal RMSE
WithAllWeatherData	XGBoost	1.043
RelevantFeatures	XGBoost	1.078
WithAvailBikes	XGBoost	1.033
WithAvailBikes	GradientBoosting	1.195

It can be seen, that with a simple feature selection, based on correlation and importance, and the addition of an extra feature we were able to slightly increase the model's performance. It is interesting that all the initial features plus the available bikes did not improve the performance. For that reason, it is not being included in the above table. The extra feature helped more, when most of the weather data were removed. Moreover, the extreme gradient boosting model shows better performance than the gradient boosting model. However, the gradient boosting model parameters were not experimented in great extent, and it is possible that with further tuning, I could be able to increase the performance to similar levels with the XGboost.

4 Conclusions

In this report, I analysed the San Francisco Bike share data, and made a first attempt to generate a model that can predict the hourly change on each station. The data provided were explored and analysed. Correlations and statistics of the features were shown and taken into account in the feature selection process. Finally, the dataset was generated in order to train successfully a model. For modelling, the option of extreme gradient boosting regressor was chosen, since it is shown to produce good results in tabular data. Moreover, it is generally fast and good in avoiding overfitting. Grid search was used to get an insight of the parameters and tune them. Furthermore, the performance was shown and estimated by doing a 3-fold cross validation on different models and different combinations of the features. An external data source was also added, introducing a useful feature for the model to better predict the net change and reduce the RMSE.

Some remarks:

- It was found, that not all the weather information were important to get the model with the best performance. This may be because the weather information are correlated together, but also not all the data are relevant to the net change.
- It is quite easy for the model to overfit. That is why, the small learning rates lead to a better performance.

- XGBoost performed better than Gradient Boosting model, and moreover it was much faster to train.
- High regularization(L1, L2) had to be introduced to avoid the model from overfitting.
- An external data source containing information about the available bikes was a meaningful feature for our model and helped to improve the performance.

5 Potential Improvements

In this section, potential improvements are discussed: 1. To begin with, an initial improvement that I would suggest, is to try to obtain *hourly weather data*. Since now, I worked with weather data information only for a specific date. This information could provide us with a better representation on how the use of bikes changes throughout the day and how this is influenced by the weather. 2. A second idea would be to increase the date range in total, i.e include more years of data. In this case, only the data of one year is used. 3. Another idea would be to change the model approach. An alternative would be to use a time series network. For example, a Recurrent neural network or LSTM, to handle our data as a time series and forecast future changes in the stations. However, the future changes are not always dependant on the past data. 4. Last but not least, I could try to model the stations as a graph neural network. It can be seen that the stations are interconnected, ie. a lot of times bikes that start from one station may end the ride to another. This could be potentially modelled using a GNN.