Data Mining Group Coursework: Bike Sharing Demand

Team LLZLL

Yunhong Liu
University of Southampton
United Kingdom
yl2y17@soton.ac.uk

Zida Li
University of Southampton
United Kingdom
zl4y17@soton.ac.uk

Bowen Luo
University of Southampton
United Kingdom
bl2m17@soton.ac.uk

Yingyi Zhang
University of Southampton
United Kingdom
yz11u17@soton.ac.uk

Yuan Lou
University of Southampton
United Kingdom
yl2m17@soton.ac.uk

ABSTRACT

Bike sharing is a burgeoning trip mode. It is widely accepted and has become more and more popular since it is good for the environment and offers an alternative way to have a short trip in the urban area. This paper presents the process of using the historical weather information and timestamp to predict the demand for bike-sharing services in Washington D.C.. The given features like *season*, *temperature* and *humidity* are analysed and preprocessed by doing feature engineering and some machine learning algorithms like Multiple Linear Regression, Random Forest and Gradient Boosting Regression Tree are used to train models to make predictions. The results show that for this predictive problem the most important feature is *hour* which is extracted from the timestamp and the model stacking produces the best prediction result.

1 INTRODUCTION

Bike sharing has become a popular way of transportation for the citizens in the main cities around the world since it is convenient, environment-friendly and ideal for going on an excursion. Nowadays there are more than five hundred programs of bike sharing system, some of them are operated by the government and the others are run by private enterprises. The data produced by the users of those bike sharing systems is attracting the attention of many researchers since this data can be used to explore the citizens' daily travel models, and the enterprise policymakers are also interested in this data to decide where to put more bike stations and when to put more bikes to meet the demands.

We get the inspiration from a Kaggle competition: *Bike Sharing Demand*[1]. In this competition, we are provided with a dataset which contains records of the usage of bike sharing service offered by Capital Bikeshare in Washington D.C., the US, between the year 2011 to 2012. Each record is a combination of a time slot, the weather condition and the total number of bikes that had been rented during that slot. The dataset is separated into the training and the test set. The training set contains the records of the first twenty days of every month and the test set keeps the left ten days' records. The aim of this competition is to train models to predict the total number of bikes that will be rented at a specific time and under certain weather conditions, so this is a regression problem. The accuracy of predictions will be evaluated on the Root Mean Squared Logarithmic Error (RMSLE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

where n is the number of records to predict, p_i is the predicted value and a_i is the actual value.

[1]https://www.kaggle.com/c/bike-sharing-demand

2 EXPLORATORY DATA ANALYSIS AND FEATURE ENGINEERING

Exploratory Data Analysis (EDA) is helpful for understanding the data better and revealing some data preprocessing operations that can be taken to improve the accuracy of prediction, so the first step is to explore the dataset. At the meantime, some feature engineering works like generating new features and dropping useless features are done in order to optimise the performance of models that will be trained in the process that follows. There are 10886 records in the training set and the test set has 6493 records. For each record, there are nine columns of features named *datetime*, *season*, *holiday*, *workingday*, *weather*, *temp*, *atemp*, *humidity* and *windspeed* respectively, and records in the training set have three other columns named *casual*, *registered* and *count*, these three represent the number of rentals made by non-registered users, registered users and the sum of them. There is not any missing value in this dataset.

2.1 Categorical Features

Among all the features, the *datetime*, *season*, *holiday*, *workingday*, *weather* are categorical features. The *datetime* has the form of "2011-01-01 08:00:00" which is not very useful for data analysis and model training, so we extract five new features *year*, *month*, *day*, *hour* and *weekday* from *datetime* and then exclude this feature from the dataset. The *season* is using [1, 2, 3, 4] to represent the four seasons from spring to winter and *weather* is using [1, 2, 3, 4] to represent " Clear or cloudy", " Misty", " Light rain or snow" and " Extreme weather". The *holiday* is labelled by Fanaee-T and Gama[2] and days with events like Black Friday, Christmas Day and Bike D.C. are labelled as 1 and the others are labelled as 0. The *workingday* uses 1 to indicate that a day is a working day and uses 0 to represent weekends. We visualise the relationships between count and these categorical features by using bar charts and line

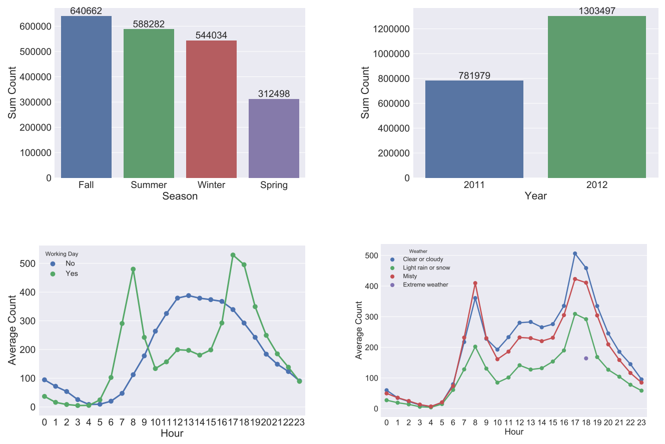


Figure 1: Examples of Categorical Feature Visualisation

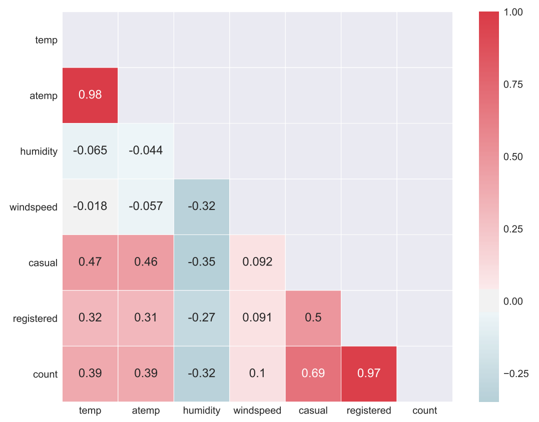


Figure 2: Heat Map of Correlation Coefficients

charts (some examples are shown in Figure 1) and here is what we found:

- The demand for bike-sharing service is doubled in 2012 compared with 2011.
- Spring has the lowest average demand for bike-sharing and summer and fall have relatively high demands.
- There are two peaks of demand in a day which are around 7-9 am and 5-6 pm, and it is well-known that these two time slots normally are rush hours.
- In working days, the peaks of bike demand appear at rush hours and the demand will drop down a lot between 9 am and 4 pm. However, on the weekends and holidays, the peaks of bike demand exactly appear between 9 am and 4 pm. Maybe these patterns imply that in the working days the major user group of bike-sharing service is office workers and students, however, on weekends and holidays, the major user group is citizens who have demands for short trips during the daytime.
- The bike sharing has the highest demand in days with nice weather conditions, which means if the weather conditions are not that ideal the bike sharing will become a secondary choice compared to other means of transportation. Especially when the weather conditions are extremely bad, the demand for bike sharing only appears at evening peak.

We also find that some features are delivering similar information, like *weekday*, *workingday* and *holiday*, *month* and *season* have some overlaps too. So some of them may be excluded from the final feature set and this will be decided in the model training process. Besides, we use OneHotEncoder from the scikit-learn library [6] to encode some of the categorical features like *season* and *weather*, so the computation of the distance between features will be more reasonable.

2.2 Continuous Features

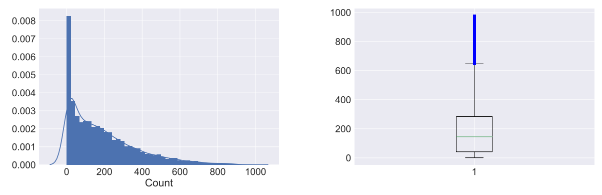
In this dataset, the continuous variables are *temp*, *atemp*, *humidity* and *windspeed*. They are the temperature, the apparent temperature, the humidity and the wind speed at a specific time. We explore the distributions of these continuous variables and find that except for *windspeed* all the other three are approximately subject to normal distribution. The reason may be that the *windspeed* has a lot of value equal to 0. We decide to create two columns of new features $windspeed\_forest$ and $windspeed\_linear$, use the RandomForestRegressor and linear_model from the scikit-learn library[6] to predict the *windspeed* based on the other known features and replace the 0 values with the predicted values. These two new features will be used to train the models to see whether if the accuracy of predictions is improved or not. Also, since the metric of every continuous feature is different, so normalisation is performed on the features.

The correlation coefficient between the continuous features and the *casual*, *registered*, and *count* is plotted as a heat map shown in Figure 2. It can be seen from the figure that the temperature and apparent temperature have positive correlations with the dependent variables while the humidity has a negative one. This proves the conclusions we have got from the previous section that the demand for biking-sharing service is higher in summer and fall and lower in days with bad weather conditions like heavy rain. Also, it can be seen from the heat map that the *temp* and *atemp* has a very high correlation between each other, so one of them may need to be dropped when training models like linear regression to avoid multicollinearity.

Based on the features that we have, we combine the values of some features and create another three new features: $temp\_humidity$, $temp\_season$ and $humidity\_season$. All these new features are generated by multiplication (e.g. $temp\_humidity = temp * humidity$) and they will be used in the model training process to see how they will affect the prediction results.

2.3 Dependent Variables

The distributions of the dependent variables *casual*, *registered* and *count* are explored as well and we find that all these distributions are right-skewed. Also, there are a lot of outliers in *count*, and all of them come from the records from 2012 due to the number of users increased a lot in this year. Since some of the machine learning models will have better performance and make more accurate


(a) Before Transformation

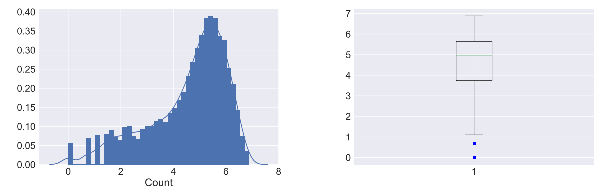

(b) After Transformation

Figure 3: Logarithmic Transformation of *count*

predictions if the dependent variable of a regression problem is normally distributed[4][3], a natural logarithm could be applied to these three dependent variables, especially the *count*, to adjust the skewness of their distributions. Most of the outliers in *count* are eliminated after skewness adjustment, so they do not have to be deleted directly and we can keep as much data records as possible.

As figure 2 shown, it is clear that *casual* and *registered* have very high correlations with *count*, so there are two ways to make predictions. The first method is to use the features to predict *count* directly, and the second method is to use the features predict *casual* and *registered* respectively and add them together to get the value of *count*. Attempts will be made to use both methods and compare their results.

Table 1: The ranking of feature imiportance

	Feature1	Feature2
1	*hour*	0.537918
2	*year*	0.081031
3	*temp*	0.067315
4	*temp_season*	0.057559
5	*atemp*	0.047905
6	*workingday*	0.041870
7	*humidity*	0.038004
8	*weekday*	0.036986
9	*temp_humidity*	0.019384
10	*season*	0.016145
11	*humidity_season*	0.015848
12	*weather*	0.012188
13	*windspeed*	0.010264
14	*holiday*	0.010264
15	*month*	0.005669
16	*day*	0.001650

3 MODELLING

At first, we utilise all features that are produced by feature engineering to build a baseline model using the Random Forest algorithm. The features we uses including *season*, *holiday*, *workingday*, *weather*, *temp*, *humidity*, *windspeed*, *hour*, *weekday*, *year*, *day*, *month*, *temp_humidity*, *temp_season* and *humidity_season*. Random Forest is able to deal with those features automatically and rank them by their feature importance, so we get the ranking as shown in the Table 1.

As we discussed earlier in feature engineering, *atemp* and *temp*, the *weekday* and *workingday*, *month* and *season* have performed a same trend respectively, thus, one of each pair will be dropped to avoid multicollinearity. Therefore, the *atemp*, *weekday* and *month* are dropped based their weaker feature importance.Next, due to the *windspeed* contains many 0 values, we try to use the *windspeed_forest* and *windspeed_linear* instead of *windspeed* for model training one by one and evaluate the accuracy of the baseline model on RMSLE, and the results show that the baseline model performs better with the *windspeed_forest*. Also, we exclude some features that have the smallest importance from the feature candidate list. Lastly, we make different combinations of the features to make a series of feature groups and train different models to fit them one by one, and finally we decide to keep two groups that have the best performance and they are shown as the following:

Table 2: The Feature Groups

Feature1	*workingday*, *weather*, *temp*, *season*, *humidity*, *windspeed_forest*, *hour*, *year*, *holiday*
Feature2	*workingday*, *weather*, *temp*, *season*, *humidity*, *windspeed_forest*, *hour*, *year*, *temp_humidity*, *temp_season*, *humidity_season*, *holiday*

Based above two feature groups and following the suggestion of feature engineering, we also made predictions on count and *log_count*. We found that forecasting the latter one could obtain better performance on training model. After that, we forecast ed the *log_registered* and *log_casual* respectively and added them together as our final predicted count value, which has better performance than previous one. Hence, in the subsequent models, we only make predictions on *log_registered* and *log_casual* to see if it could performance better or not.

3.1 Multiple Linear Regression

Multiple Linear Regression (MLR) is a regression method that can be implemented to derive parameters, which might give a scientific interpretation about the relationship between independent variable and dependent variables[5]. In this project, we choose MLR as our first model to predict bike sharing demand.

3.1.1 *Implementation of MLR.*

As the input of MLR needs to be numerical, we implement one-hot encoding method on categorical features, such as season and weather, rather than passing these features directly to the MLR model. What is more, we also normalize numerical features, such as temperature and humidity, when feed the data into MLR models.

In order to avoid overfitting and test different performance of MLR models, we implement three different MLR models in this process. The first model is MLR without regularization. The second model is MLR with Ridge regularization and the third model is MLR with Lasso regularization.

3.1.2 Result and Evaluation of MLR.
The results of each MLR model are shown in Table 2.

Table 3: The RMSLE score of different MLR models

	Feature1	Feature2
MLR	0.81614	0.80807
MLR+Ridge	0.81418	0.80800
MLR+Lasso	0.81383	0.97367

From Table 3, we can derive that the performance of models might increase with introducing regularization but not increase significantly. What is more, we can also derive that the performance of MLR models might increase after feature selection but the increment is also not quite significant. The RMSLE scores decrease after introducing new features in the MLR model and MLR with Ridge regularization model. However, the RMSLE score of MLR with Lasso regularization model increases after introducing new features. This might because Lasso regularization will automatically do feature selection, which might drop important features to some extent, while the ridge regularization might keep all features instead.

3.2 Random Forest

Random forest is a commonly ensemble learning method, which we used it to solve the regression prediction problem in this experiment. The basic idea of the random forest is to build multiple decision trees. For the regression problem, the random forest outputs the average tree of all decision trees. Random forest can solve the problem of overfitting the decision tree to the training set.Therefore, it will be implemented in this prediction task.

3.2.1 Implementation of RF.
In this experiment, we used 'RandomForestRegressor' package, which provided by the python machine learning library scikit-learn to solve the prediction problem. We consider that tree regression has been able to deal with type variables well, what is more, there is a partial correlation between the values of category variables in the data, for example, the *weather* value is settled from small to large with weather condition is from good to bed. Therefore, in the implementation processing of Random Forest and following Gradient Boosting, there is no special method to deal with the category variable.
In the Random Forest, the main parameters we considered as following: $n_estimators$, $max\_features$, $maximumdepth$ (use the defaults), $min\_samples\_spli$, $min\_samples\_leaf$. We used 'GridSearch' package with cross-validation method to adjust the features above. Among all the features, $n_estimators$ are more important. We firstly adjusted that, and then we utilized the idea of gradient descent and control the variables to adjust other parameters in turn until the parameters do not change.

The results are shown as following:

Table 4: The optimal parameters of feature 1

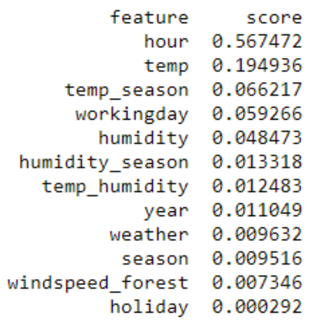
	casual_log model	registered_log model
n_estimators	1000	900
max_features	8	6
min_samples_leaf	8	6
min_samples_split	4	2

Table 5: The optimal parameters of feature 2

	casual_log model	registered_log model
n_estimators	1000	900
max_features	8	8
min_samples_leaf	8	2
min_samples_split	8	2

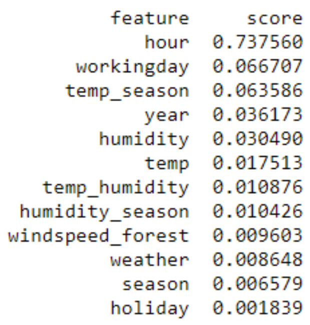
After the parameters selection, we pass our data into models with best parameters respectively and derive the results, which are shown as next part.

3.2.2 Result and Evaluation of RF.



```
         feature     score
            hour  0.567472
            temp  0.194936
     temp_season  0.066217
      workingday  0.059266
        humidity  0.048473
 humidity_season  0.013318
   temp_humidity  0.012483
            year  0.011049
         weather  0.009632
          season  0.009516
 windspeed_forest  0.007346
         holiday  0.000292
```

Figure 4: Feature importance of casual user with feature 2



```
         feature     score
            hour  0.737560
      workingday  0.066707
     temp_season  0.063586
            year  0.036173
        humidity  0.030490
            temp  0.017513
   temp_humidity  0.010876
 humidity_season  0.010426
 windspeed_forest  0.009603
         weather  0.008648
          season  0.006579
         holiday  0.001839
```

Figure 5: Feature importance of registered user with feature 2

As we can see in the above figures, the *hour* value is more relevant to the registered user. It seems that registered users have regular time which might in their commuting time to ride sharing bicycle. And the next important feature also might prove it. For casual users, the second important feature is *temp*, which means they commonly according to the temperature to decide whether they use the bike sharing service or not. But, for the registered user, the secondary important feature is *workingday* which shows they commonly use the service in the working days.In addition, the new added feature *temp_season* is the third important feature for both of them, which means the temperature relates to season and they could be important factors for casual user and registered user to decide whether they use sharing bike or not.

Table 6: The results of different RF models

Model	RSMLE score
Model for feature 1	0.40041
Model for feature 2	0.40054

Because we dedicated to increase the accuracy of model to fit data better, thus the RMSLE score is smaller and greater. From Table 4, we can derive that the performance of RF is better than MRL. What is more, parameters selection might have positive on the performance of RF. It has decreased the RMSLE score compare to without parameter selection. However, there is no much difference between those two RF models with utilizing different feature groups. In feature 1, we extract additional information, and dropped the useless feature. During this procedure, it leads to 0.40041 RMSLE score of the first RF model with the feature groups, which developed significantly comparing with MLR. However, feature engineering might have negative on the performance of RF. In feature 2, we added three features: *temp_humidity*, *temp_season*,*humidity_season* which leads to 0.40054 RMSLE score of the second RF model by applying the second feature groups. The performance of the second model decreases compared with the first RF model, which means this model using feature 1 could fit data better and the new features have not contribution in this RF model.

3.3 Gradient Boosting Regression Tree

In order to test the performance of different models, we also implement gradient boosting regression tree (GBRT) in this project, which is a boosting method that generates base models sequentially[7].

3.3.1 Implementation of GBRT.
As the input of GBRT can be numerical and categorical data, we can pass our features into GradientBoostingRegressor model directly. However, before passing features into models, we use gridsearch method to derive best parameters one by one. As the result, the best parameters of each features are shown in Table 7 and Table 8.

Table 7: The optimal parameters of feature 1

	casual_log model	registered_log model
n_estimators	450	700
min_samples_leaf	8	6
min_samples_split	28	4

Table 8: The optimal parameters of feature 2

	casual_log model	registered_log model
n_estimators	150	450
min_samples_leaf	8	1
min_samples_split	24	28

3.3.2 Result and Evaluation of GBRT.
After choosing optimal parameters for each model, we pass our data into models with optimal parameters respectively and derive the results, which are shown in Table 9.

Table 9: The results of different GBRT models

Model	RSMLE score
Model for feature 1	0.36912
Model for feature 2	0.37837

From Table 9, we can derive that the performance of GBRT is quite good, comparing with the MLR model. What is more, feature engineering might have positive influence on the performance of GBRT. In feature 1, we extract additional information, such as *hour* and *windspeed_forest* and also drop useless information, such as *month*. During this procedure, it leads to 0.36912 RMSLE score of the first GBRT model, which develop significantly comparing with MLR. However, feature engineering might have negative influence on the performance of GBRT. In feature 2, we add three features: *temp_humidity*, *temp_season* and *humidity_season*, which leads to 0.37837 RMSLE score of the second GBRT model. The performance of the second model decreases compared with the first GBRT model, which means the new features have not contribution to the model at all.

3.4 Model Stacking

Stacking is one method of ensemble learning algorithms. It can combine information from base multiple models using a meta-model. Generally speaking, stacking will get better performance when the basic models are different. The base level models are trained based on the whole data set, and then the meta-model is trained on the outputs of the base level model as features.

The main steps of stacking are as follow:

- Split the training set into K folds.
- Use the base models to fit K folds and save the outputs of the whole models.
- Fit the base models in the whole test model and t the average to combine each model output.
- Fit the L1 model in the new training set and test set.

Compared with MLP, random forests and gradient boosting tree are better models. So we regarded the two models as Level 1 models. And we used XGBoost as the Level 2 model. And from the third part, we knew that the performance of feature1 is better than that of feature2 in both random forests and gradient boosting tree. Because of that, we decided use features1 in our stacking model.

The Extreme Gradient Boosting algorithm(XGBoost) is an open-source software library created by Tianqi Chen[1]. This method is chosen because of its significant advantages:

- Compared with other Gradient Boosting algorithms, XGBoost has regularization which can help us to reduce overfitting. Generally speaking ,one of the most famous features of XGBoost is the regularized boosting technique.
- Can deal with missing values. It has an in-built routine to handle missing values.
- Can use the parallel algorithm to construct trees[1].
- It was successfully used in Machine Learning competitions[1].

The best values of parameters are as follow:

Table 10: Values of parameters

	casual_log model	registered_log model
n_estimators	94	70
subsample	0.9	0.7
max_depth	2	2
colsample_bytree	1	0.7
min_child_weight	26	6
reg_alpha	1	0

The result of stacking is 0.36800. Compared with the single model(for example, RF and GBRT), the result of the stacking is better,which means that the stacking can improve the performance of our project.

4 MODEL ANALYSIS

Firstly, the feature engineering is important before building models, which might have both positive and negative influence on models. For the feature 1, we extract more information from original data, such as *hour* and $windspeed\_forest$, which contributes significantly to the model accuracy. For the feature 2, we extract more features, such as *temp_humidity*, *temp_season* and *humidity_season*, which have negative influence on the model accuracy.

Table 11: The performance of different models in bike sharing demand

Model	Feature 1	Feature 2
MLR	0.81614	0.80807
MLR+Ridge	0.81418	0.80800
MLR+Lasso	0.81383	0.97367
RF	0.40041	0.40054
GBRT	0.36912	0.37837
Model Stacking	-	0.36800

Secondly, from above table, we can derive that stacking model performs the best on this dataset, followed by random forest and gradient boosting regression tree algorithms and the the traditional multiple linear regression algorithm performs the worst on this dataset. This might because the high dimension dataset might not be linear in the high dimension space, which leads to the worst performance of MLR model. However, stacking model might combine the advantages of different models together, which might lead to the best performance in this project.

5 FUTURE WORKS

So far, we implement feature engineering to drop non-significant features and extract three new features. What is more, we also implement different machine learning techniques to predict bike sharing demand, such as MLR, RF, GBRT and model stacking. However, there is still improvement for this project. We can draw some inferences for suggesting the future work.
Firstly, in the feature engineering process, more features can be extracted to increase accuracy of models because the performance of model using feature2 is not quite good.
Secondly, in our project, we apply log function to loss target. Another better way is to shift it 200 points right. If we can make this shift and then take the log function, the performance maybe is better (The winner of this competition did this).
Finally, the hour feature is related to time series problem, which indicates that deep learning model, such as RNN, might have better performance. As a result, deep learning model, such as RNN, can be implemented in the future works.

6 CONCLUSIONS

In conclusion, in this project, we use feature1 and feature2 to build MLP, RF, GBRT and stacking models respectively. As a result, the stacking model performs the best. In addition, the experiment shows that for this predictive problem the most important feature is *hour*, which can be used to explore the citizens' daily travel habits. After knowing citizens' daily travel habits, the enterprise policymakers might use this data to do decisions, such as where to establish bike stations and when to put more bikes, which might serve citizens better and solve urban traffic problems to some extent.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 785–794.
[2] Hadi Fanaee-T and Joao Gama. 2014. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* 2, 2-3 (2014), 113–127.
[3] Bill (https://stats.stackexchange.com/users/25212/bill). [n. d.]. What is the reason the log transformation is used with right-skewed distributions? Cross Validated. arXiv:https://stats.stackexchange.com/q/107646 https://stats.stackexchange.com/q/107646 URL:https://stats.stackexchange.com/q/107646 (version: 2017-04-13).
[4] Max Kuhn and Kjell Johnson. 2013. *Applied predictive modeling*. Vol. 26. Springer.
[5] Kar Yong Ng and Norhashidah Awang. 2018. Multiple linear regression and regression with time series error models in forecasting PM10 concentrations in Peninsular Malaysia. *Environmental monitoring and assessment* 190, 2 (2018), 63.
[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.
[7] Yanru Zhang and Ali Haghani. 2015. A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies* 58 (2015), 308–324.