MINI PROJECT REPORT

Xporters-Analysis of the Traffic Data

Eva Agrawal[a], Laetitia Clerc[b], Jean-François Gassie[b], Robin de Groot[a], and Hongyi Luo[a]

[a]EIT Data Science M1, Paris-Saclay, FR; [b]L2 Informatique, Paris-Saclay

ARTICLE HISTORY
Compiled May 3, 2020

Team name/project: VELO Xporters
THE NUMBER of the last submission on Codalab: 0.9471171511
the URL of the Youtube video: `https://youtu.be/xuWztENPmJ8`
The URL of the presentation slides (PDF):
Challenge URL: `https://codalab.lri.fr/competitions/652`
Github repository of the project: `https://github.com/jf-pnj/velo/tree/master/starting_kit`

1. Background and motivation

The world is moving fast towards globalization, and transport plays a crucial role in establishing the connections. All the transport modes, be it air, rail or road make an extensive network which connects cities and countries. Hence, the data we get from these networks is also huge and can be exploited to benefit the business processes, e.g. the supply chain can be streamlined, the best efficient routes can be identified etc.

The provided document of this challenge has a brief story about a young man. He has a lemonade stand next to a busy road. In order to know how much lemonade he should have in stock and when he should man the stand, it is beneficial to know how many cars will come by in a certain time frame, since more cars equals more potential customers.

Since tabular data is very widespread in many organisations, we thought it would be wise to familiarise ourselves further with this kind of data.

2. Data and problem description

The project we have been working on is the 'Xporters' project which is a multivariate regression based problem. It deals with predicting the traffic volume on a highway on

CONTACT Eva Agrawal. Email: eva.agrawal@u-psud.fr
CONTACT Laetitia Clerc. Email: laetitia.clerc@u-psud.fr
CONTACT Jean-François Gassie. Email:jean-francois.gassie@u-psud.fr
CONTACT Robin de Groot. Email: robin.de-groot@u-psud.fr
CONTACT Hongyi Luo. Email: hongyi.luo@u-psud.fr

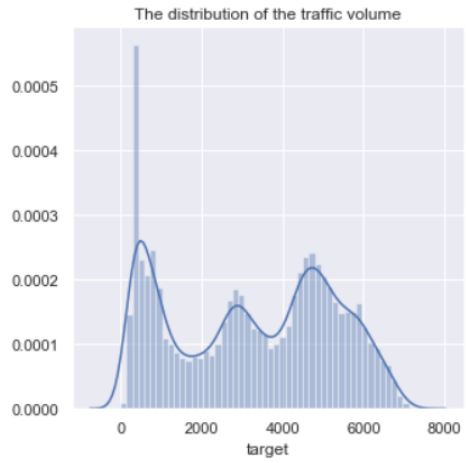

Figure 1. Traffic volume distribution

Figure 2. Traffic volume by day

a per hour, per day basis. We also study the effect of additional meteorological factors such as rain, temperature etc. making to a total of 59 features to characterise the data. The target variable is the total number of vehicles passing through the highway at an hour in the day.

The Xporters challenge is based on a small standard regression data set from the UCI Machine Learning Repository [16], formatted in the AutoML format. It uses a data set concerning the traffic volume on a highway in the USA from 2012 to 2018 [17].

Figure 1 shows the distribution of traffic volume, we can observe that the mean traffic volume is somewhere around 3000. In the Figure 2, the traffic volume is presented with respect to days in a week. From the graph we can see that the traffic gradually increases as the week progresses and then there is a sharp decrease with the start of the weekend.

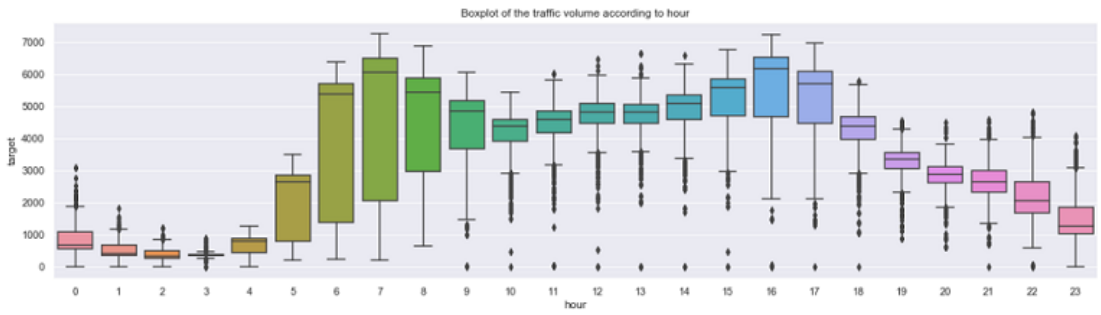

Figure 3. Traffic volume by hour

The hourly representation of traffic volume is shown in Figure 3. Most of the traffic is concentrated in early morning hours, starting from 6AM to 9AM and later there in evening the traffic becomes sparse comparatively.

3. Algorithm and model

First of all, the algorithm of the entire project is mainly concentrated in two parts: (I) Preprocessing (II) Model.

3.1. Preprocessing

The purpose of the preprocessing part is to reduce the number of dimension of the data and to clean the data to make predictive models more efficient without losing (much) accuracy. Usually, we must first complete the data exploration to ensure that there are no completely wrong or irrelevant features in the data set. However, in this project, our data was already cleaned, thus we only need to use the data manager to load.

We noticed that the total amount of the data set is more than 30,000 and the data dimension is up to dimension 59, which is obviously too high. In order to improve the efficiency of the data in the model, we adopted the following preprocessing algorithms:

3.1.1. PCA

Principal components analysis (PCA) is probably the most famous and commonly used preprocessing method, it is based on the following idea: In the process of research many factors need to be considered, but these factors may have a certain relationship with each other, so we can summarize the reference factors that are related to each other.[15] So we need to fuse the various features of the data while retaining the main information to get lower dimensional data.

3.1.2. LDA

Linear discriminant Analysis(LDA)'s idea, Unlike the PCA variance maximization theory, is to project the data into a low-dimensional space so that the same type of data is as compact as possible and the different types of data are as dispersed as possible. Thus, the LDA algorithm is a supervised machine learning algorithm. At the same time, LDA has the following two assumptions: (1) The original data is classified according to the sample mean. (2) Different types of data have the same covariance matrix. In practice, it is impossible to satisfy the above two assumptions. However, when the data is mainly distinguished by the mean value, LDA can generally achieve good results.[3]

3.1.3. LLE

Locally linear embedding(LLE) is a non-linear dimensionality reduction algorithm that can keep the original manifold structure after the dimensionality reduction of the data. The LLE algorithm believes that each data point can be constructed by a linear weighted combination of its neighbors.[4] The main steps of the algorithm are divided into three steps: (1) finding k neighboring points of each sample point; (2) calculating the local reconstruction weight matrix of the sample point from the neighboring points of each sample point; (3)The locally reconstructed weight matrix of the sample point and its neighboring points calculate the output value of the sample point.

3.2. Model

In the model phase, we will test the prediction performance of different machine learning models and compare their differences. Most of these models have been implemented in scikit-learn. The work we have done is to correctly apply them and continuously change the parameters to achieve the best prediction results. At present, we have optimized multiple models including neural networks.

As for our original idea, it is first reflected in our selection of several models. For the high dimensionality and large number of features of our data set, we first decided to use a random forest for prediction, because a random forest is often good at handling a large number of high dimensionality data.

However, interpretability is also essential. As a well interpretable algorithm, KNN can directly reflect the degree of separability within the data set and can be logically linked to the results.

The advantages of algorithms such as boosting are that they can integrate multiple weaker models to achieve a better effect. It can be used as the baseline for us to judge the models.

As a model of excellent performance in recent years, neural networks is an unmissable choice. According to experience, it often has a very good prediction rate.

In detail, we use MAE and R2 Score to evaluate our several models, because this is more stable and repeatable than the most direct accuracy rate.

3.2.1. R2 metric

In general, any model needs an indicator to evaluate its performance. Here, we use the R2 score to compare the performance of the following models.

The R2 score is called coefficient of determination in statistics, and it is used to measure the proportion of dependent variables that can be explained by independent variables to judge the explanatory power of statistical models.

Specifically, suppose we have n observations $y_1, \ldots, y_n$ in our dataset. The predicted values of our model are $f_1, \ldots, f_n$. Obviously we can define the residual $e_i = y_i - f_i$ from the bias between label and predict value. So we have a mean value $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

According to the statistical formula, we can further obtain the total sum of squares : $SS_{\text{tot}} = \sum (y_i - \bar{y})^2$ and the residual sum of squares: $SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$

Finally, we define the R2 score as $R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$.

Generally, if the R2 score is closer to 1, the model is better, if it is 0 or negative, it means that the training has no effect.

3.2.2. k-Nearest Neighbor

The idea of this method is: In the feature space, if most of the k nearest (ie, the nearest neighbors in the feature space) samples near a sample belong to a certain category, the sample also belongs to this category.[6]

3.2.3. Random Forest

A Random Forest Regressor is a regressor that contains multiple decision trees, and the output of the whole model is determined by votes from the individual trees.

```
k-Nearest Neighbor
Classify (X, Y, x) // X: training data, Y: class labels of X, x: unknown sample
for i = 1 to m do
  Compute distance d(X_i, x)
end for
Compute set I containing indices for the k smallest distances d(X_i, x).
return majority label for {Y_i where i ∈ I}
```

Figure 4. Pseudocode for KNN

3.2.4. Gradient Based Prediction

Gradient Boosting Regressor is a boosting method commonly used in regression and classification problems. It generates prediction models in the form of a set of weak prediction models (usually decision trees). The main idea is that each time, a new model is established in the gradient descent direction of the previous model loss function. Gradient Boost is a framework in which many different algorithms can be nested such as decision trees.

3.2.5. Neural Network regression

Neural networks are high-performance machine learning models that have overtook much of machine learning in recent years. They use computational models that mimic the structure and function of biological brains to estimate or approximate functions to the data. Neural networks are often multi-layer structures. Each layer has a certain number of neurons, and each neuron will automatically have different weights through convolution and other methods during training. These weights represent the importance of prediction. The epoch training can make the prediction of the neural network closer to the training set, but we should pay attention to the problem of overfitting.

3.3. Training loop

In order to train the models and select the best performing one, multiple steps should be taken. Firstly, the dataset was preprocessed using the methods we described in section 3.1. Secondly, to test what hyperparameters work best per model and per preprocessed dataset, Random Search with Cross Validation (3-Folds) was used. This process was automated to some extent through the functions *optimize_hyperparameters* and *optimize_models_dataset*, the second of which encapsulates the first one.

The results of the tests are the best hyperparameters per model, per dataset. The next step was to test the optimized models against each other per dataset. This was done using the classes *model()*, which calls the *AutomatedModelTest()* class. The results from running *model()* is the best performing model on the dataset out of the input models. [1]

[1]For the Python implementation, see `https://github.com/jf-pnj/velo/blob/master/starting_kit/README_model_AutoTest.ipynb` and `https://github.com/jf-pnj/velo/tree/master/starting_kit/sample_code_submission`

Table 1. R2 scores for the models with different pre-processing techniques

Model	**R2 Score**				
	No Pre-proc.	LDA	LLE	PCA	t-NSE
Support Vector Regressor	0.068	0.619	0.003	0.072	0.009
Linear Regression	-3.537	0.156	0.007	0.167	0.004
Nearest Neighbour Regressor	0.773	0.902	-0.087	0.774	0.688
Gradient Boosting Regressor	0.918	0.883	**0.041**	0.731	0.129
Feedforward Neural Network Regressor	0.678	0.703	0.002	0.641	0.013
Random Forest Regressor	**0.951**	**0.913**	-0.006	**0.914**	**0.715**
AdaBoost Regressor	0.819	0.775	0.022	0.494	0.026

4. Results

In the project to predict the traffic volume a number of regression models have been implemented. For evaluation we have used the standard R-squared metric which represents the the proportion of the variance for the traffic volume explained by the given independent variables. The results are submitted on Codalab challenge [18] and the leaderboard is shown in Figure 5. The R2 score for our submission is 0.947 and our position in leaderboard is 6th as highlighted below in black color.

RESULTS

#	User	Entries	Date of Last Entry	Prediction score ▲	Duration ▲	Detailed Results
1	MOTO	11	04/01/20	0.9589 (1)	0.00 (1)	View
2	Caravan	4	03/20/20	0.9547 (2)	0.00 (1)	View
3	scooter	35	02/07/20	0.9473 (3)	0.00 (1)	View
4	Truck	26	03/15/20	0.9473 (4)	0.00 (1)	View
5	AUTOCAR	25	03/23/20	0.9472 (5)	0.00 (1)	View
6	Velo-Xporters	5	04/03/20	0.9471 (6)	0.00 (1)	View
7	AUTOBUS	10	03/21/20	0.9467 (7)	0.00 (1)	View
8	Segway	6	02/28/20	0.9467 (7)	0.00 (1)	View
9	Taxi	39	02/21/20	0.9467 (7)	0.00 (1)	View
10	pavao	22	02/07/20	0.9467 (7)	0.00 (1)	View
11	automobile	28	03/31/20	0.1661 (8)	0.00 (1)	View
12	medichal	15	10/15/19	-2.6512 (9)	0.00 (1)	View
13	xporters	1	10/12/19	-2.6512 (9)	0.00 (1)	View

Figure 5. Codalab Leaderboard

The results of various models implemented in the project along with different pre-processing techniques used are summarized in table 1. Here, the respective R2 score [19] are recorded for every model implemented before and after applying the pre-processing technique. We can see that every method has it's own effect on the data and the performance sometimes even decreases after applying the pre-processing method.

Though we applied a range of algorithms for predicting the traffic volume, results from some of the best performing algorithms are shown through scatter plots of the real versus predicted values in figures 6 and 7. These results are obtained after applying PCA [15] to the data. The Random Forest method gave a good score of 0.95 on the validation data set, when applied without any further pre-processing than scaling the data. For the Neural Network, the datapoints in figure 7 are different from figure 6. The reason is that the Neural Network wasn't trained on scaled data, and thus these

Figure 6. Random Forest Regressor, R2 score: 0.951

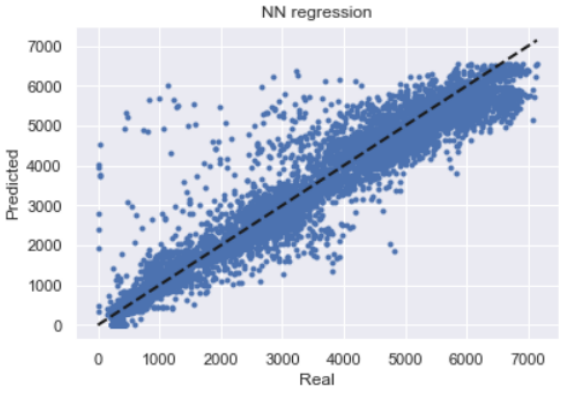

Figure 7. NN regression, R2 score: 0.923

are considered as preliminary Neural Network results.

5. Discussion and conclusion

In conclusion, out of all the models implemented for the completion of this project, the best performance has been achieved on the test set with an R2 value of 0.94 and the random forest model performed best among all the models.

In terms of pre-processing, the methods PCA and LDA are found most effective in reducing the dimensions of the data to increase learning speed and prediction accuracy. These two methods are generally most effective and widely used for pre-processing purposes due to their ability to reduce the dimensions of the data without losing (much of) the information. The support vector regression and linear regression model did not present effective results for this dataset. One main reason could be the high dimensions and features in the dataset which are not quite linearly dependent.

There were some obstacles to overcome while working on this project. The first obstacle is computational power. In order to test many models on differently prepared data, a lot of computations have to be done. Additionally, if for every model the hyperparameters need to be optimized, the computational load becomes even higher and testing thus more difficult. An improvement here could be to test the different models on a smaller subset of the data, compare their performance, and only train the best model on the full dataset. However, his has drawbacks on the confidence in the choice of model.

Another process that took a lot of time was to automate all the testing. This took a lot of time to do. Still, it taught the authors a lot of good programming skills and it made the testing of different models much easier since it is done automatically.

A model that we tried, the Neural Network, took some time to properly setup for this project. These models are a lot harder to train as they are more computationally expensive and they have many more hyperparameters to optimise, increasing the number of tests necessary. Thus, for these models, adequate time and effort are necessary. Still, a Neural Network could perform very well on this problem and therefore it is a good option to try.

There are several techniques that did not make it into this project, mostly due to time limitations and limited knowledge of the authors. The main potential probably lies in time-series based models. These kind of models are able to take the regularity

of data into account (previous day, days of the week, time of year, etc) which gives rise to their potential. An addition to common time-series algorithms, something else that could be tried is Recurrent Neural networks. These networks are also able to take the regularity in the data into account and thus have the potential to perform well on this dataset. Something that should be done is different pre-processing, such as de-trending the data, but the potential is there for these models, so it could be very worthwile.

Another part that we wished we had more time for was hyperparameter optimization of the models. The model testing process was automated, but the latest script does not have an implementation for a hyperparameter search algorithm (such as random search) and therefore there is probably still additional performance that can be 'squeezed' out of the models.

6. References

References

[1] Scikit-Learn PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html?highlight=pca#sklearn.decomposition.IncrementalPCA

[2] Scikit-Learn TSNE: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html?highlight=t%20sne#sklearn.manifold.TSNE

[3] Scikit-Learn LDA: https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html?highlight=linear%20discriminant%20analysis#sklearn.discriminant_analysis.LinearDiscriminantAnalysis

[4] Scikit-Learn LEE: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.locally_linear_embedding.html?highlight=locally%20linear%20embedding#sklearn.manifold.locally_linear_embedding

[5] Scikit-Learn random forest: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=random%20forest#sklearn.ensemble.RandomForestRegressor

[6] Scikit-Learn KNN: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html?highlight=k%20nearestneighbor#sklearn.neighbors.NearestNeighbors

[7] Scikit-Learn GradientBoostingRegressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html?highlight=gradientboostingregressor

[8] Pytorch: https://pytorch.org/docs/stable/nn.html

[9] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[10] README.ipynb from the Velo starting kit

[11] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[12] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(Feb), 281-305.

[13] Shaikh, Raheel. "Cross Validation Explained: Evaluating Estimator Performance." Medium, Towards Data Science, 26 Nov. 2018, towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85.

[14] Scikit-Learn Outlier Detection: https://scikit-learn.org/stable/modules/outlier_detection.html

[15] Scikit-Learn PCA:https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[16] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
[17] Xporters Data: README.ipynb from the Velo starting kit
[18] Codalab Challenge:https://codalab.lri.fr/competitions/652#results
[19] R2:https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html