Pre-processing : SOLTANI Hicham (Group 2bis) et LIU Yongjie (Group)
Model : CHEN Shuangrui (Group 1bis) et XU Zhenghai (Group)
Visualization : YACHOUTI Mouad (Group 1bis) et ALCANTARA HERNANDEZ Ursula (Group 1)

Project Report

L2 MI – Mini Projet - Xporters - SEGWAY

URL Codalab: ***https://codalab.lri.fr/competitions/652***
URL Video: ***https://www.youtube.com/watch?v=Lqt8-EexUBE***
URL Github: ***https://github.com/UrsulaAlcantara/Segway***
URL Proposal: ***https://github.com/UrsulaAlcantara/Segway/blob/master/starting_kit/New_Proposal.pdf***

Pre-processing
Model
Visualization

03/04/2020

Contents

1. Background and description of the problem

A traffic prediction challenge called Xporters proposes a study of traffic , weather, prices, and many other features to explor capacity of machine learning work.

Xporters challenge is a regressor problem in which pre-processing data features must be analysed as well as detailed model regressor study may be done in order to found the best prediction results. As a team work , representations of features are necessary to an easy review.

We will resume our mean functions and methods that have leaded us to our final results. with a description of what ever sub-group has made and quickly summarize our regressor results.

2. Description and pseudo-codes

2.1 Pre-processing

In the pre-processing part we processed and used the data to allow a better analysis. To do this we had to think about how to reduce and optimize the variables. First, we create one or more copies of the data and then we eliminate the fields that are not necessary for the analysis or that do not affect the result. We can test the different data obtained. This method does not affect the result, but reduces the calculation time and limits errors. Then we transform the data using an algorithm, we chose to use the Principal Component Analysis algorithm, because it optimizes the data. This algorithm will group the linked data, so we get less variable (variable grouped by the algorithm) which will facilitate the visualization of the data. We did not use any other algorithm for our pre-processing part because we think we have reduced and optimized the data.

2.2 Model selection

We have chosen 8 models, one of them is a Voting Regression which takes opinion from 3 others. After prudent comparison our best model according to model performances are the RandomForest Regressor under pre-processing of no outliers, Voting Regression under several pre-processing conditions and GradientBoosting Regressor under raw data or no outlier data. And after cross-validation, we find they are all almost 94% correction, so we decide to use RandomForest Regressor as our final choice, for under similar condition it's lightly better than GradientBoosting Regressor, but also way more efficient than Voting Regression, because as we know, it's in fact 3 models combined together.

2.3 Best model principle

The best model we chose is RandomForest Regressor, it tries to form a tree that shows every possible situation, that's why it gains 100% correction in Training set. In view of the shortcomings of decision trees that are easy to overfit, random forest uses a voting mechanism of multiple decision trees to improve the decision tree. We assume that random forest uses m decision trees. For a tree, it is obviously not desirable to train m decision trees with a full sample every time. Full sample training ignores the rules of local samples, which is harmful to the generalization ability of the model.
Random forests de-correlate all trees through random interference, so random forests performs better than Bagging. Random Bags are not like Bagging. When constructing each tree, a random sample predictor is used before each node is split. Because in the core idea, the random forest is still the same as the Bagging tree, its variance is reduced. In addition, random forests can consider using a large number of predictors, not only because this method reduces bias, but also the local feature predictor plays an important role in the tree structure.
With no doubt, RandomForest Regression is the best model, because it can generate a clear tree structure based on feature selection of different pre-processing results. But it's also easy to be influenced, as its final decision of the decision tree is usually based on a single condition, we need only to change a few features to get a different detection. Its randomness could decrease the risk of this part. And it's still capable in training procedure, for having the ability to describe all possible division of features, it's also less easy to get over-fitting results, which makes our model more generalizing. So it suits our cases, as we have 58 features, and dozen of them are quite influential.

We will explain its detail later.

2.4 Classification And Regression Trees (CART)

The random forest will use bagging, and bagging applies Bootstrap's ideas, so let's first talk about Bootstrap and bagging.

Bootstrap:
If we have a sample of size N, we want to get m samples of size N for training. Then we can do this: First, randomly select a sample X_1 out of N samples, and then write it down, put it back, and then take out an X_2, ..., so that it is repeated N times, and you can get a new sample of N. In this new sample There may be duplicates. Repeat m times to get m such samples. In fact, it is a random sampling problem with replacement. Each sample is drawn with the same probability (1 / N) each time it is drawn. This method is useful when the sample is relatively small. For example, our sample is small, but we want to set aside a portion for verification. If the traditional method is used for train-validation segmentation, the sample is smaller and the bias will be bigger, this is undesirable. The Bootstrap method does not reduce the size of the training sample, and it can leave the verification set (because the training set has repetitions, but this repetition is random), so it has certain advantages.

How many samples can Bootstrap leave for verification?

The probability of any sample not being drawn is $(1 - 1/N)$ every time it is drawn, and a total of N times are drawn, so the probability that anyone sample does not enter the new sample is $(1 - 1/N)^N$. So in a statistical sense, it means that there are probably $(1 - 1/N)^N$ samples as verification sets. When N tends to positive infinity (N -> ∞), this value is about $1/e$, 36.8%.Using these as the validation set is called out-of-bag estimation (out-of-bag estimation)

Bagging:
The name of bagging comes from Bootstrap Aggregating. This method divides the training set into new training sets, and then builds a model on each new training set, which is irrelevant. When we finally predict, we will use the results of the m models. Integration to get the final result. The integration method is: majority voting for classification problems and the average value for regression.
Random forest (RF):

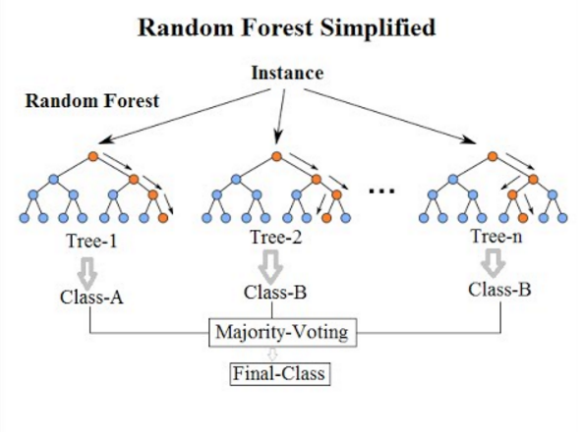

Figure 2.1 – Random forest

	model perf
KNeighborsRegressor	0.707741
SVR	-0.00177721
GaussianProcessRegressor	0.112257
ElasticNet	0.161813
DecisionTreeRegressor	1
RandomForestRegressor	0.972539
GradientBoostingRegressor	0.932471
VotingRegressor - GBoosting - DecisionTree	0.983118
VotingRegressor - GB - DT - RandomForest	0.987621
StackingRegressor - GB - DT - RF	0.982578

Figure 2.2 – Model Performance

First, RF uses the CART decision tree as a weak classifier; second, based on using the decision tree,

RF has improved the establishment of the decision tree-in the ordinary decision tree, we will have all n samples on the node Among the features, an optimal feature is selected to divide the left and right subtrees of the decision tree, but RF randomly selects a part of the sample features on the node, assuming n_sub. Then, RF selects an optimal feature among the randomly selected n_sub sample features to divide the left and right subtrees of the decision tree.

If n_sub = n, then the RF CART decision tree is no different from the ordinary decision tree. In addition, the smaller n_sub, the variance of the model will decrease, but the offset will increase. In the specific implementation of the algorithm, a suitable n_sub value is generally obtained through cross-validation tuning. A single decision tree model is easily affected by noise data, while a mixed model is not. But if you train multiple trees on the same data, it is also easy to get a strongly related tree (or the same tree), then the effect is not good;

Algorithm Random :

Input: Sample set D = $(x_1, y_2), (x_2, y_2), ...(x_m, y_m)$, training round T, weak learner algorithm h
Output: strong classifier f (x)
1: **for** t = 1 to T **do**
2: Randomly sample the training set for the t-th time, and sample a total of m times to obtain a sample set D_t containing m samples.
3: Train D_t decision tree models h_t (x) with a sample set D_t. When training the nodes of the decision tree model, select a part of the sample features from all the sample features on the node, and select an optimal feature among these randomly selected part of the sample features to make the left and right subtree division of the decision tree.

4: **end for**
5: Integrate T weak classifiers to get the final strong classifier.

2.5 Visualization

First steps:
Our sub-group visualization started studying the representation of the predicted results. We thought it was important to understand how predicted targets had been developed. We first tested with public data features but we realized poor features weren't enough to have a good representations of model results. Even if we haven't great illustrative plots, after pre-processing improvement we could have better curves to be compared. We used seaborn distplot features for the 3 data set results (see last graphic from README_modelVisual.ypinb).

Comparing model and pre-processing results
After sub-group model and pre-processing had new results and different models to be compared. We used plot of platplotlib library with X axe of Xtests of every pre-processing method (ex: shuffle , outliers, pca, svd, tsne) and Y ax, target results of each pre-processing. We superimposed every predicted target to see how results resemble between different pre-processing.

Then, we made the score Graphics of each model method compared to each pre-processing method. We matched different plot representations in a function called score_plots. We used plot, bar and horizontal bar. Thanks to this function it provided us a better analysis of model selection.

Improving scores
At the end, we questioned our features selection and decided to work with features relevance that at the beginning were ignored. This features were oil prices and holiday , that weren't chose by the best features. with plot , we see low oil prices were present before traffic decreased and the opposite with holidays, compared with time.

We chose to not put here pseudo-code of our personal functions because it wasn't appealing to see direct application of seaborn and matplotlib functions.

3. Results

3.1 Performance compared with the combination of model and pre-processing

	KNeighbors	SVR	GaussianProcess	ElasticNet	DecisionTree	RandomForest	GradientBoosting	Voting - GB - DT	Voting - GB - DT - RF
Raw data	0.545000	0.000000	0.000000	0.161000	0.880000	0.932000	0.915000	0.920000	0.930000
Remove Outliers	0.555000	0.000000	0.000000	0.163000	0.890000	0.938000	0.922000	0.926000	0.934000
PCA	0.530000	0.005000	0.000000	0.060000	0.221000	0.564000	0.489000	0.473000	0.534000
SVD	0.523000	0.005000	0.000000	0.061000	0.288000	0.573000	0.489000	0.513000	0.563000
TSNE	0.792000	0.237000	0.000000	0.000000	0.668000	0.677000	0.545000	0.709000	0.756000
VarianceThreshold	0.558000	0.000000	0.000000	0.162000	0.893000	0.938000	0.920000	0.927000	0.935000
SelectKBest	0.717000	0.000000	0.000000	0.151000	0.879000	0.934000	0.917000	0.923000	0.929000
SelectFromModel	0.611000	0.712000	0.000000	0.000000	0.886000	0.938000	0.920000	0.924000	0.933000

3.2 Visualization

Like we said in our proposal, score bar helped us to conclude that SVD and tsne models got similar predictions, like shuffle and out-liers with the best scores. But plot representations proved that voting and neighbors and voting were one of the best model method, and interestingly,decision tree regressor were badly classified.

We tested our methods using many trial, changing features and studying sklearn libraries. Our original ideas were to see targets predicted of test, traning and final results. This may helped enssentially validating our results.
Most of our results were developed in our proposal.
To enrich our studies, you can found ourvisualization, model and pre-processing codes at:

https://github.com/UrsulaAlcantara/Segway/tree/master/starting_kit/testFile .

https://github.com/UrsulaAlcantara/Segway/blob/master/starting_kit/README_modelVisual.ipynb .

4. Discussion and Conclusion

After improvement of code modeling and a detailed optimization data, our score hasn't reached a higher rating. We got 94.0. Details must be seen to improve this result and we may focus on a better processing of main data as binary features of weather or oil prices and holidays that were ignored until now. It would be better , for the next year to have in clear that the key is to develop a rich representation of the situation and to know since the beginning of the project, sub-groups need to share more information between team members.

1. Breiman L (1984) Classification and regression trees. The Wadsworth and Brooks-Cole statistics probability series. Chapman & Hall.

2. Wiki - Gradient Boosting https://en.wikipedia.org/wiki/Gradient_boosting

3. Mlxtend Sebastian Raschka Assistant Professor of Statistics at the University of Wisconsin-Madison http://rasbt.github.io/mlxtend/user_guide/regressor/StackingRegressor/

4. Understanding Random Forest https://towardsdatascience.com/understanding-random-forest-58381e0602d2

5. A TUTORIAL ON PRINCIPAL COMPONENT ANALYSIS https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

6. Visualizing Data using t-SNE http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

7. Outlier detection between statistical reasoning and data mining algorithms https://findresearcher.sdu.dk:8443/ws/files/153197807/There_and_Back_Again.pdf