Report of MOOC

Machine Learning

Guangyue CHEN

2018/12/12

Contents

1 Methodology

1.1 Linear Model

For this problem, the linear models have bad performances, but there are also a way to build a linear model. Here we tried 'linear regression' , 'Ridge Regression' and ' Lasso Regression'. To make some features working such as 'DayOfWeek', 'StoreType' and 'Assortment', we use the ont hot function from package 'mltools'. As the results, all of these three regression have a test error feedbacked from kaggle.com which is arroud 0.4.

1.2 SVM

For the SVM method, we choose the package "e1071" to train our model, so our first step is choosing the regression type and the kernel for this regression.

1.2.1 Kernels and Type selection:

	linear	polynomial	radial	sigmoid
eps-regression	0.17834	0.15844	0.13579	0.35867
nu-regression	0.18955	0.16936	0.14461	0.46985

FigureX: Train Error with different kernels and types used in SVM regression

We can see that with the Kernel 'radial' and the type 'eps-regression', the model permance better.

1.2.2 Parameter Choosing:

Here we use the function 'tune.svm' from the package "e1071" to compare the different gamma and cost. This function use K-ford cross validation to choose the parameter. The gamma parameter defines how far the influence of a single training example reaches. The cost parameter rules the error of the cutting plane. With higher cost the train error will be lower but the test error may growth, and the parameter gamma is for the Kernel, it's also sensitive for our model.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma  cost
  0.05    3
```

FigureX: Best Parameter

1.2.3 Result:

Because of the running time of the svm model is long, we didn't pay too much attention into this model. Our result error is .

Submission and Description	Private Score	Public Score	Use for Final Score
SVM.csv just now by guangyue add submission details	0.13841	0.12382	☐

FigureX: Result

1.3 Random Forest and H2o

1.3.1 Random Forest:

Random Forest is a flexible, 'easy to use' machine learning algorithm, it porduce a good result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks.

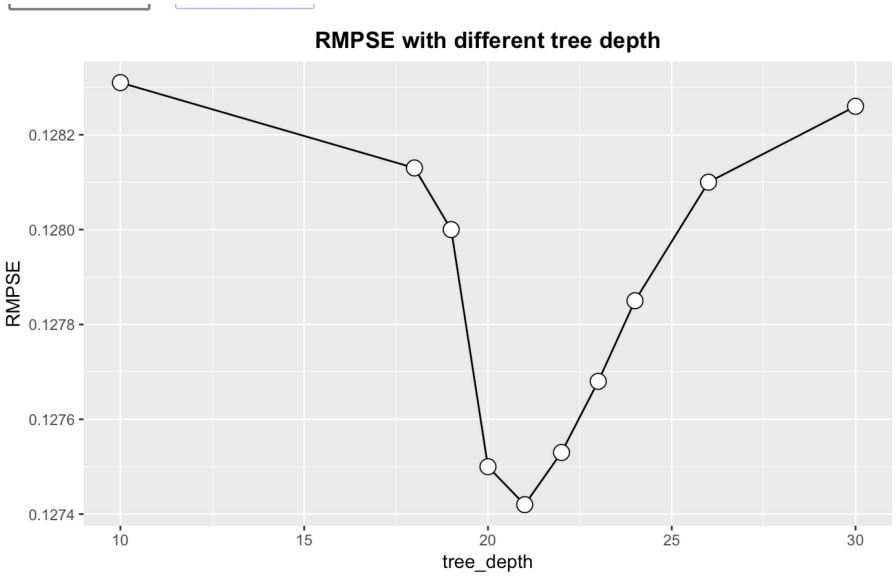
When given a set of data, Random Forest generates a forest of classification or regression trees, rather than a single classification or regression tree. Each of these trees is a weak learner built on a subset of rows and columns. It chose the features and the subset of the data(for training a tree) randomly. More trees will reduce the variance. So it could handle well the overfitting issues. For our problem, it has also a good performance.

1.3.2 H2o Random Forest:

'H2o' package use Distributed Random Forest, which is a powerful classification and regression tool. For this package, 'h2o.randomforest' run faster than the normal one in R, it can also limit the tree depth, (R's randomForest builds really deep trees), allowing for having a better predictions.

1.3.3 Parameter Choosing:

So here we should choose the parameter 'max_depth', here we use Cross-Validation to compare the test error.



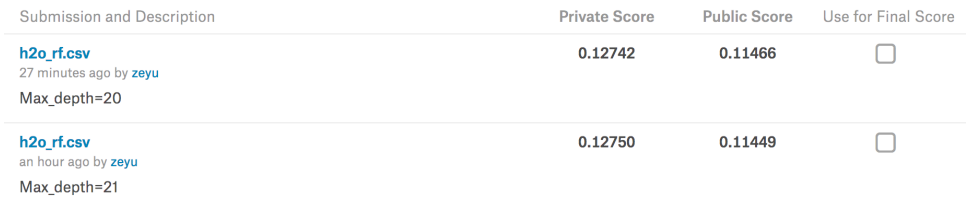
FigureX: RMPSE with different tree depth

We can see that for the test error, the models with depths 20 and 21 have the best porfermences. So we decided to build two models with the depth which are 21 and 20. Then we build a forest with a big quantity of trees, Which is 100.

1.3.4 Feature Selection:

For h2o random forest, we should load the data into h2o cluster. After our several test, we find that some features make the models perform worse. With the summary of our model, we decide to remove some features which have a low importance to our model. After we remove two features 'SchoolHoliday' and 'StateHoliday', our random forest model perform better.

1.3.5 Result:



Submission and Description	Private Score	Public Score	Use for Final Score
h2o_rf.csv 27 minutes ago by zeyu Max_depth=20	0.12742	0.11466	☐
h2o_rf.csv an hour ago by zeyu Max_depth=21	0.12750	0.11449	☐

FigureX: The Result Of Our Models

We run our best models on kaggle.com provided test data. The test errors feedbacked from kaggle.com are 0.11449 and 0.11466. So we can say that the forest with depth maximum 21 is better. And this result is already in the top 150 on kaggle.com.

2 Conclusion

According our results, Random Forest has the lowest test error feedbacked from kaggle.com. But we believe that the SVM model could perform as well as RF although it cost so much time for learning once. So for our future work, we will do more anlyses and tries on SVM.

After this project, we realize that the feature treatment has a large impact on training model quality. A correct feature selection could helps us to develop simpler and faster models. Once features are chosen and formatted correctly, the prediction error improved dramatic. Data preprocessing is the same, because, the representation and quality of data is first and foremost before running an analysis.

What's more, after this project, we have an unforgetable experience of data analysis. We know clearly the steps to treat the data and train the machine learing models, and have a clearer understanding of models such as Random Forest and Support Vector Machine.

3 Reference

1. Data source:https://www.kaggle.com/c/rossmann-store-sales/data
2. Distributed Random Forest Introduction:http://docs.h2o.ai/h2o
3. Data preprocessing and feature selection:(en chinois)https://blog.csdn.net
4. Support Vector Machine: http://uc-r.github.io/svm