Kaggle Competition - Two Sigma Connect: Rental Listing Inquiries

BIA 2016Fall
Yi Luo
Jun Wang
Xiaozhou Yu

Abstract

In this project, our goal is to make predictions about how much interest will a new rental listing on RentHop receive. It's a Kaggle competition co-hosted by Two Sigma and RentHop. We first apply exploratory analysis to the dataset given, then cross-validition statistic and stacking methods are used to improve model performance.

1 Background and Challenge

1.1 Background

Summary

This is a Kaggle competition co-hosted by Two Sigma and RentHop where competitiors will apply analysis featuring rental listing data from RentHop.

`https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries`

The goal is to predict the number of inquiries a new listing receives based on the listingĂŹs creation date and other features. Doing so will help RentHop better handle fraud control, identify potential listing quality issues, and allow owners and agents to better understand rentersĂŹ needs and preferences.

Evaluation

Submissions are evaluated using the multi-class logarithmic loss. Each listing has one true class. For each listing, the model will give a set of predicted probabilities (one for every listing). The formula is then,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij}) \tag{1}$$

where N is the number of listings in the test set, M is the number of class labels (3 classes), log is the natural logarithm, y_{ij} is 1 if observation ii belongs to class jj and 0 otherwise, and p_{ij} is the predicted probability that observation ii belongs to class jj.

1.2 Challenges

First of all, there are many factors that can impact the interest level of a new listing such as room number, manager id, price and features. Each feature comes in different format. Thus, data cleaning and exploration is necessary before further analysis. We also need an advanced model that can deal with various types of variables and deliver accurate predictions.

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

Second, we need to figure out how to deal with the randomness of the data and create new features that can convey key information from training data to testing data. As we'll illustrate later, a technique called cross-validation statistics will be used to address this problem.

In addition, although general tree based models usually give very good performances, we still need parameter tuning in pursuing a better performance. Ensemble methods such as stacking also plays a vital role in improving prediction accuracy, at the expense of heavy computing burden and longer iterations.

2 Data Cleaning and Exploratory Data Analysis

2.1 Data Cleaning

- Covert column Created to datetime format; extract month and day information as new features.
- Count the distinct values of each instance in column photos and description, add as new features.
- Deal with categorical features, label encode columns such as display address, manager id, building id, street address.
- Compute the top 200 most frequent terms in column feature, add dummy variables to represent the apperance of the term in each record.
- Compute average price by room number.

2.2 Data Exploration - Insights into manager id

At first glance, the average interest level seems to differ substantially with respect to column manager id. As shown below, the top 50 managers (with the most listings) have very different high/medium/low distributions.

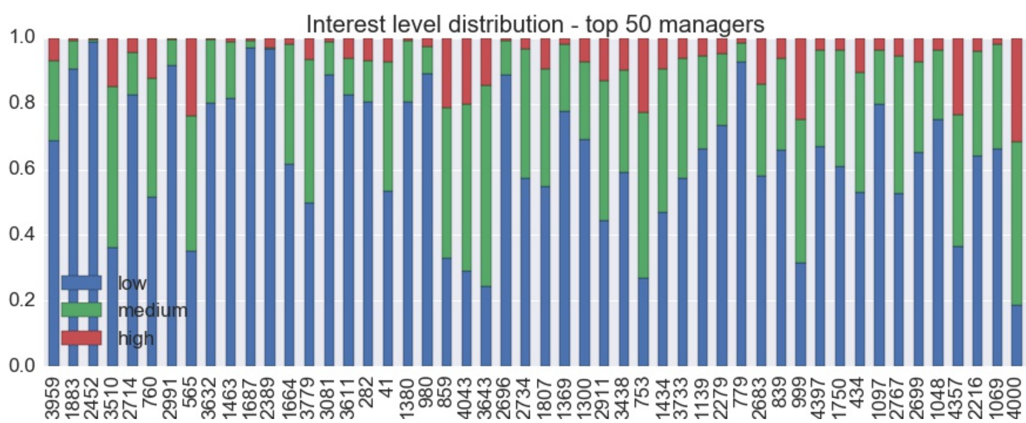

Figure 1: Interest level distribution over manager id

A closer look at the cumulative count of listings shows that he Pareto principle, i.e. the 80/20 rule, applys here. As 20% of the managers are roughly responsible for roughly 80% of the records in the dataset. Next, we can begin feature engineering focusing on manager id.

3 Cross-Validation statistics

3.1 Summary

A technic called Cross-Validation statistics (CV statistics) is applied to improve the performance of our interest level classifier. According to the output, a significant improvement is detected by applying cross-validation statistics. The idea of cross-validation statistics is generated from cross-validation.

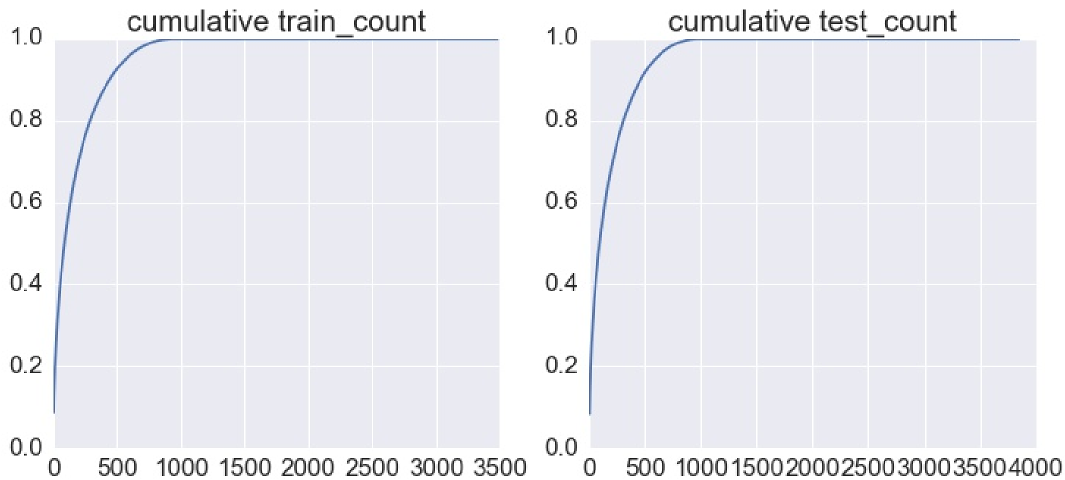

Figure 2: Cumulative count of listings over manager id

3.2 Introduction

Cross-validation is a tool to estimate the variance of your performance metric due to randomness in the data. The reason we use cross-validation is that it is difficult to report an accurate performance using only one training dataset and one test dataset. It is likely that different performance figure will be generated by the exact same method and parameters. Because of the randomness of data, the performance figure may vary a lot unless everyone use the exact same split. The idea of cross-validation is to randomly split the dataset to some equal size sub-datasets. The first sub-dataset will be used as the test dataset while other sub-datasets will be used as training datasets. Then, same procedure will be repeated n times (n equal to the number of sub-datasets) and every sub-dataset is used exactly once as the test dataset. Afterwards, the mean and standard deviation of n performance figures generated in each test is provided as the final prediction. Although cross-validation will magnify the computational cost and therefore slow down the whole program especially when datasets are large, it is still a widely-used tool.

3.3 How does CV statistics work

Just like cross-validation, the cross-validation statistics needs to split the whole dataset into some equal-size sub-datasets and then make the execution. The idea is to split the training dataset and one sub-dataset will be assigned the posterior probability from other sub-datasets.

By applying this technique, the program will be trained how to deal with low-frequency observations as well as new observations which only occurs in test dataset. In our project, prediction is made partially based on manager id, many of which occurs at a low frequency, some of the manager ids even only occurs once. Prediction under such circumstances may not be reliable, to cope with the low-frequency problem, we applied cross-validation statistics.

To be specific, the training dataset is split into five sub-datasets: a, b, c, d, e. The posterior probability calculated from other four sub-datasets will be assigned to the remaining sub-dataset. Firstly, we assign posterior probability from sub-dataset a, b, c, d to sub-dataset e. If an observation (for example, manager id: 001) occurs in a or b or c or d but not in e. The posterior probability will not be assigned to any place. However, if an observation occurs in e but not in a or b or c or d, value NaN will be assigned to that observation. Then the procedure is repeated just like the cross-validation.

As a result, in the training dataset, the NaN values represent the observations only occurs in one sub-dataset. During training, the model will learn how to deal with these NaNs and achieve better performance under these NaNs. Therefore, when there are new observations in the test dataset, the model could handle them better and make more accurate predictions. The cross-validation statistics could be treated as an improved Bayesian encoding. For the Bayesian encoding, such low frequent samples will be assigned prior probabilities. However, they are different from the samples that really hold probabilities close to prior.

4 Stacking

4.1 Introduction and Applications

Stacking (sometimes called stacked generalization) involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although in practice, a single-layer logistic regressionămodel is often used as the combiner.

Stacking typically yields performance better than any single one of the trained models.[1] It has been successfully used on both supervised learning tasks (regression, [2] classification and distance learning [3]) and unsupervised learning (density estimation).[4] It has also been used to estimate bagging's error rate.[5] It has been reported to out-perform Bayesian model-averaging.[6] The two top-performers in the Netflix competition utilizedăblending, which may be considered to be a form of stacking.

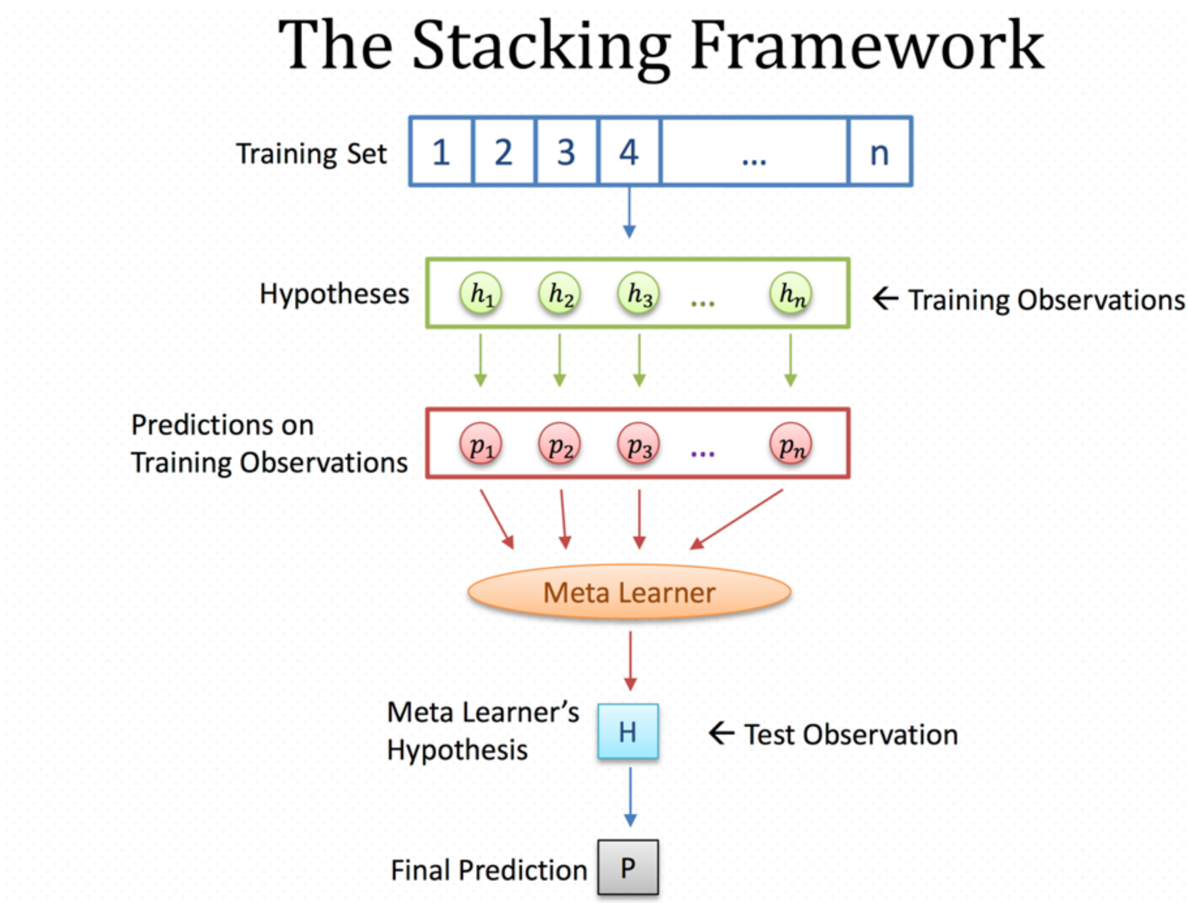

Figure 3: The stacking framework

The stacking framework

- The training set for the meta-level classifier is generated through a leave-one-out cross validation process.

- The learned classifiers are then used to generate predictions.
- The meta-level dataset consists of examples of the form

  $$((y_i^1, ..., y_i^n), y_i),$$

  where the features are the predictions of the base-level classifiers and the class is the correct class of the example in hand.

4.2 Mathematical insight into stacking ensemble

If an ensemble has M base models having an error rate e < 1/2 and if the base modelsĂŹ errors are independent, then the probability that the ensemble makes an error is the probability that more than M/2 base models misclassify the example. The simple idea behind stacking is that if an inputĂŞoutput pair (x, y) is left out of the training set of , after training is completed for , the output y can still be used to assess the modelĂŹs error. In fact, since (x, y) was not in the training set of h_i,h_i(x) may differ from the desired output y. A new classifier then can be trained to estimate this discrepancy, given by y ĂŞ h_i(x). In essence, a second classifier is trained to learn the error the first classifier has made. Adding the estimated errors to the outputs of the first classifier can provide an improved final classification decision.

4.3 How does stacking ensemble work

Stacking takes place in two phases.

In the first phase, each of the base level classifiers takes part in the j-fold cross validation training where a vector is returned in the form

$$((y_i^1, ..., y_i^m), y_i),$$

where y_ii is the predicted output of the m the classifier and y_j is the expected output for the same.

In the second phase this input is given for the Meta learning algorithm which adjusts the errors in such a way that the classification of the combined model is optimized. This process is repeated for k-fold cross validation to get the final stacked generalization model.

4.4 Advantages of stacking ensemble methods

It is found that stacking method is particularly better suited for combining multiple different types of models. Stacked generalization provides a way for this situation which is more sophisticated than winner-takes-all approach. Instead of selecting one specific generalization out of multiple ones, the stacking method combines them by using their output information as inputs into a new space. Stacking then generalizes the guesses in that new space. The winner-takes-all combination approach is a special case of stacked generalization.

The simple voting approaches have their obvious limitations due to their abilities in capturing only linear relationships. In stacking, an ensemble of classifiers is first trained using bootstrapped samples of the training data, producing level-0 classifiers. The outputs of the base level classifiers are then used to train a Meta classifier. The goal of this next level is to ensure that the training data has accurately completed the learning process. For example, if a classifier consistently misclassified instances from one region as a result of incorrectly learning the feature space of that region, the Meta classifier may be able to discover this problem. Using the learned behaviors of other classifiers, it can improve such training deficiencies. [8]

4.5 First rst-layer mode used

4.5.1 XGBoost

XGBoost stands for eXtreme Gradient Boosting. The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms.

XGBoost Features The library is laser focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features.

Model Features The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:

- Gradient Boosting algorithm also called gradient boosting machine including the learning rate.
- Stochastic Gradient Boosting with sub-sampling at the row, column and column per split levels.
- Regularized Gradient Boosting with both L1 and L2 regularization.

System Features The library provides a system for use in a range of computing environments, not least:

- Parallelization of tree construction using all of your CPU cores during training.
- Distributed Computing for training very large models using a cluster of machines.
- Out-of-Core Computing for very large datasets that donĂŹt fit into memory.
- Cache Optimization of data structures and algorithm to make best use of hardware.

Algorithm Features The implementation of the algorithm wasăengineered forăefficiencyăof compute time and memoryăresources. A design goal was to make the best use ofăavailable resourcesăto train the model. Some key algorithm implementationăfeatures include:

- Sparse Aware implementation with automatic handling of missing data values.
- Block Structure to support the parallelization of tree construction.
- Continued Training so that you can further boost an already fitted model on new data.

XGBoost is free open source software available for use under the permissive Apache-2 license.

The two reasons to use XGBoost are also the two goals of the project:

- XGBoost Execution Speed: Generally, XGBoost is fast. Really fast when compared to other implementations of gradient boosting.
- XGBoost Model Performance: XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform. The XGBoost library implements the gradient boosting decision tree algorithm.

This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines. Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. This approach supports both regression and classification predictive modeling problems.

4.5.2 SVM

Inămachine learning,ăsupport vector machines (SVMs, also support vector networks[1]) areăsupervised learningămodels with associated learningăalgorithmsăthat analyze data used forăclassificationăandăregression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilisticăbinaryălinear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called theăkernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

4.5.3 Random Forests

Random forestsăor random decision forestsăare an ensemble learningămethod foräclassification,äregressionăand other tasks, that operate by constructing a multitude ofădecision treesăat training time and outputting the class that is the modeăof the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit ofăoverfittingăto their training set.

4.5.4 Extremely Randomized Trees

Adding one further step of randomization yields extremely randomized trees, or ExtraTrees. These are trained using bagging and the random subspace method, like in an ordinary random forest, but additionally the top-down splitting in the tree learner is randomized. Instead of computing the locally optimal feature/split combination (based on, e.g., information gain or the Gini impurity), for each feature under consideration, a random value is selected for the split. This value is selected from the feature's empirical range (in the tree's training set, i.e., the bootstrap sample)

4.5.5 multilayer perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in aădirected graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (orăprocessing element) with a nonlinearăactivation function. MLP utilizes aăsupervised learningătechnique calledăbackpropagationăfor training the network. MLP is a modification of the standard linearăperceptronăand can distinguish data that is notălinearly separable.

4.5.6 k-NN

k-NN is a type ofăinstance-based learning, orălazy learning, where the function is only approximated locally and all computation is deferred until classification. Theăk-NN algorithm is among the simplest of allămachine learningăalgorithms. Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, whereădăis the distance to the neighbor. The neighbors are taken from a set of objects for which the class (foräk-NN classification) or the object property value (foräk-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. A peculiarity of theăk-NN algorithm is that it is sensitive to the local structure of the data. The algorithm is not to be confused withăk-means, another popularămachine learningătechnique.

5 Results and Discussions

5.1 Summary

At the end, the two-layer stacking model is built for interest level classification. The first-layer model includes six sub-models, which are k-nearest-neighbors, random forest, extremely randomized trees, SVM, multi-layer perceptron classifier and XG Boost. In the second-layer model, we used XGBoost that takes the running results of the first layer model as the input. The final log-loss is 0.5200, wheras the winning kernel log-loss is 0.4919.

5.2 Limitations and Possible Improvements

Even though the project has already achieved its objective, there are still some limitations.

First, the result of the best-performing model of the first-layer models decides the overall performance of the stack model. If we could do more parameter tunining in the first-layer models and implement more feature engineering to the data, we could have gained better results.

Second, as discussed before, stacking requires a large amount of computing time. Limited to if we could have more computing resources, the performance of the model could be enhanced by stacking more layers.

Last, the leak feature ĂIJimage add timeĂİ significantly improved the performance of the model. If we could master picture-processing techniques such as convolutional neural networks, then we could get more features from the photos. By adding these features to the current model, the performance could be significantly improved.

Acknowledgments

This project mainly used XGBoost developed by Chen Tianqi:

`https://github.com/tqchen/xgboost`

This project also learned from Kaggle Kernels listed below:

`https://www.kaggle.com/guoday/two-sigma-connect-rental-listing-inquiries/cv-statistics-better-parameters-and-explaination`,

`https://www.kaggle.com/den3b81/do-managers-matter-some-insights-on-manager-id`,

`https://www.kaggle.com/mmueller/allstate-claims-severity/stacking-starter/run/390867/code`.

References

[1] Wolpert, D., Stacked Generalization., Neural Networks, 5(2), pp. 241-259., 1992

[2] Breiman, L., Stacked Regression, Machine Learning, 24, 1996 doi:10.1007/BF00117832

[3] Ozay, M.; Yarman Vural, F. T. (2013). "A New Fuzzy Stacked Generalization Technique and Analysis of its Performance". arXiv:1204.0171Ăŕ

[4] Smyth, P. and Wolpert, D. H., Linearly Combining Density Estimators via Stacking, Machine Learning Journal, 36, 59-83, 1999

[5] Wolpert, D.H., and Macready, W.G., An Efficient Method to Estimate BaggingĂŹs Generalization Error, Machine Learning Journal, 35, 41-55, 1999

[6] Clarke, B., Bayes model averaging and stacking when model approximation error cannot be ignored, Journal of Machine Learning Research, pp 683-712, 2003 Sill, J.; Takacs, G.; Mackey, L.; Lin, D. (2009). "Feature-Weighted Linear Stacking". arXiv:0911.0460Ăŕ

[7] Nikunk C. Oza, and Kagan Tumer, Classifier Ensembles: Select Real-World Applications, journal Information Fusion VOLUME 9 issue 1, January, 2008 Pages 4-20

[8] Sudheep Elayidom, Sumam Mary Idikkula, and Joseph Alexander, A hybrid stacking ensemble framework for employment prediction problems. ISSN: 0975ĂŞ3273 & E-ISSN: 0975ĂŞ9085, Volume 3, Issue 1, 2011, PP-25-30