CS376 Term Project

Team 11

December 14, 2018

1 Model Descriptions

1.1 Converting Data Format

After observing the data, we first found that the format of some data is quite different with others. Most features are represented as integer or float number, but contract date and construction completion date have a format of 'YYYY-MM-DD'.

We just divided contract date to contract year and contract month. For example, date 1992-07-12 will be divided to 1992 and 07. So, the number of current features is 24. We didn't modify completion date because it will be dropped in 1.2.

1.2 Dropping Features

For the next step, we have to decide features to be dropped. The first thing to consider is correlation between features. Figure 1 shows the result, and the details are in attached 'correlation.csv' file. We noticed that some features have very high correlation, so we decided to focus on correlations which are bigger than 0.9.

In the process of training data, features that highly correlated each other are not needed. i.e. we can use just one feature of them. So, we determined to drop some features using correlation result. We dropped 6 features: 1st class region id, 2nd class region id, road id, the limit number of car of parking lot, the total area of parking lot, and construction completion date.

Also, XGBoost tool provides the value of feature importance. Some features have very low importance, and we decided to drop these features too.

As a result, we will finally consider only 10 features: contract year, contract month, latitude, longitude, apartment id, floor, angle, area, the average management fee of the apartment, and average age of residents.

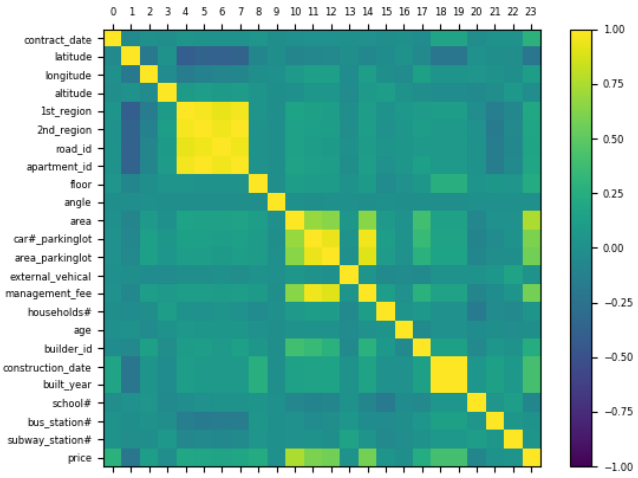

Figure 1: Correlations Between Features

1.3 XGBoost Tool

To fill the missing values, add new feautres, and build a model, we used XGBoost tool. XGBoost is an optimized distributed gradient boosting library[1], and stands for Extreme Gradient Boosting[2]. As its name says, its algorithms are based on gradient boosted trees and they are under the gradient boosting framework. In the library, we utilized xgboost.XGBRegressor, which is an implementation of the scikit-learn API for XGBoost regression.

Here's some explanations on the term. Boosting is an **ensemble method** for primarily reducing bias, and variance too in supervised learning.[3] [4] Gradient boosting[5] is similar to boosting, but the former differs from the latter on the way it creates the weak learners. It outputs a prediction model that takes the form of an ensemble of weak prediction models, which are usually decision trees.[6] Further information can be found in the XGBoost website and our course materials.

We provided some theoretical details on the ensemble learning in the appendix so the reader can refer to that.

Weak learners are classifiers, which are slightly correlated with the true classification.

[1] https://xgboost.readthedocs.io/en/latest/index.html

[2] https://xgboost.readthedocs.io/en/latest/tutorials/model.html

[3] Leo Breiman (1996). "BIAS, VARIANCE, AND ARCING CLASSIFIERS"

[4] Zhi-Hua Zhou (2012). *Ensemble Methods: Foundations and Algorithms* also defined it as a family of machine learning algorithms that convert weak learners to strong ones, which was already proved earlier in Schapire (1990)."The strength of weak learnability".

[5] The term came from Friedman, J. H. (1999). "Greedy Function Approximation: A Gradient Boosting Machine"

[6] Wikipedia, "Gradient boosting"

1.4 Filling Missing Values (Imputation)

After dropping some features, only 10 features remain. Among these, there are some features which have missing value.

1.4.1 Training Data

The number of rows in training data is 240,594, and 6,813 rows have at least one missing values. We thought that training data must have exact values, and 6,813 out of 240,594 is very little. So we determined to drop those 6,813 rows, and train the model with only 233,781 rows.

1.4.2 Test Data

In case of test data, we cannot drop rows which have missing values. We will use only 10 features to train the model, so we have to fill the missing values of those features. We simply used XGBoost tool to predict and fill them. For example, filling the number of households in the apartment, we trained a model with other 9 features in training data, and predicted the values of the number of households in test data.

1.5 Adding New Features

The one of the key point of the project is that we have to predict future data. We thought that if past data exists more, the future prediction will be more accurate. So we determined to add 6 new features, the price of one month before to six months before.

Here is more details. First, we built a model with original training data using XGBoost. I will call this model 'model A'. With model A, we predicted the price of one month before to six months before for each row. For example, contract date of the first row of training data 1992-07. Then, what would be the price if the contract date was 1992-06 or 1992-01? We predicted these prices and added them as new features in both training data and test data.

1.6 Time Series Validation

The test data is related to predict the future, so time series cross validation method would be better. Figure 2 shows how it works. We ranged training data with contract date, and did 5-fold time series cross validation.

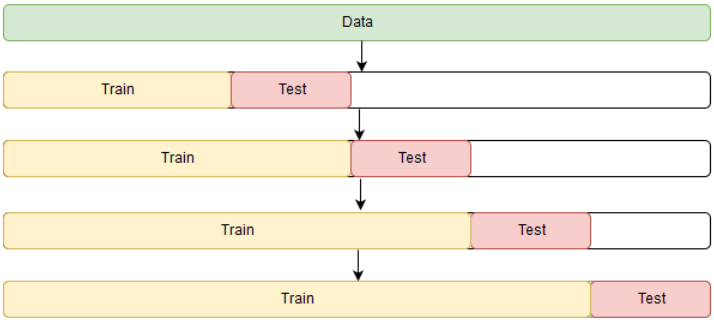

Figure 2: Time Series Cross Validation Method

1.7 Predicting Test Data

Now, the number of valid features is 16, because we added 6 more features to 10 features. with this data, we built a new model, model B, using XGBoost. Finally, with model B, we can predict the price of test data.

2 Unique Method

Our unique method is related to **feature engineering**. More specifically, we added new 6 features, the price of one month before to six months before. The details of the method are explained in 1.5.

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work.

This method is unique because we made more past data by adding new features. We thought that new features will be useful to predict future data, and the accuracy could increase.

Since some of the test data is taking a future form of training data, i.e. those will have nearly same features except the time-related feature, we decided to **drag** the data from future to past instead of predicting the price directly using the features of test data. If we add the predicted price from the **dragged** data as a new feature, our model would yield better accuracy because they have relatively high chance to be contained in the time period of the training set. Also, because XGBoost is a tree based model, predicting new future data would yield relatively low accuracy, but predicting data with similar date will be more accurate.

To summarize, the base method would be predicting test data with 10 features. On the other hand, our unique method predicts with 16 features.

3 Libraries

Note that we used Python 3.5.6.

3.1 XGBoost

Version: 0.81
Purpose: To utilize it as a base learning algorithm for our model.

3.2 Scipy

Version: 1.1.0
Purpose: It helps the process of converting our mathematical expression to python code.

3.3 sklearn

Version: 0.20.1
Purpose: To help implement our algorithm without reinventing-the-wheel that much.

3.4 Numpy

Version: 1.15.4
Purpose: To do some numerical computations for the project.

3.5 Pandas

Version: 0.23.4
Purpose: To manipulate and analyze data that used in the project.

3.6 matplotlib

Version: 3.0.2
Purpose: To plot and visualize the data.

4 Source codes

4.1 project.py

This python file is the main file that pre-processes data, builds models, and predict test data.

4.1.1 correlation_plot

This function analyzes the correlation values between features and makes plot. The result is saved in 'correlation.csv' file and 'correlation.png' file.

4.1.2 time_series_validation

We implemented time series cross validation method in this function. We ranged training data with contract year and month, did 5-fold time series cross validation, and tuned hyper-parameters.

4.1.3 predict

This function builds a model with tuned hyper-parameters and predicts the price values of test data. Note that this function does not include the unique method, but only base method. It saves the result by creating a 'predict.csv' file.

4.2 missing_value.py

This file is for filling missing values of test data. It reads 'data_test.csv' file, and create a new file named 'data_test_no_blank.csv'. The details of method to fill missing values is explained in 1.4.3.

5 Performance

5.1 Time Series Cross Validation without Unique Method

Because we have to predict the future, it would be better to use time series cross validation method. First, without unique method, Table 1 summarizes our final hyper-parameter values. Also, Table 2 shows the accuracy using time series cross validation with tuned hyper-parameters.

hyper-parameter	value
n_estimators	100
learning_rate	0.05
gamma	0
subsample	0.6
colsample_bytree	1
max_depth	10

Table 1: Hyper-parameter values without unique method

nth 5-fold	accuracy
0	0.941843
1	0.956858
2	0.956223
3	0.952972
4	0.935043
mean	0.948588

Table 2: Time series cross validation result without unique method

5.2 Time Series Cross Validation with Unique Method

Now, with unique method, Table 3 summarizes our final hyper-parameter values. Also, Table 4 shows the accuracy using time series cross validation with tuned hyper-parameters. We can see that accuracy increases when new features are added.

hyper-parameter	value
n_estimators	500
learning_rate	0.06
gamma	0
subsample	0.75
colsample_bytree	1
max_depth	10

Table 3: Hyper-parameter values with unique method

nth 5-fold	accuracy
0	0.974537
1	0.984297
2	0.983611
3	0.982967
4	0.981743
mean	0.981431

Table 4: Time series cross validation result with unique method

6 Appendix

6.1 Details on ensemble learning

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary machine learning approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use. Ensemble learning is appealing because that it is able to boost weak learners which are slightly better than random guess to strong learners which can make very accurate predictions. So, "base learners" are also referred as "weak learners". It is noteworthy, however, that although most theoretical analyses work on weak learners, base learners used in practice are not necessarily weak since using not-so-weak base learners often results in better performance. [7]

Ensemble Methods are defined by what generation processes are used and what combination schemes are applied.

How can one construct an ensemble? the way can be explained in two steps. First step is to generate base learners. In other words, we need to decide a base learning algorithm that would deal with the training data.[8] According to Krogh and Vedelsby (1995)[9], the base learners should be *accurate* and *diverse*[10] as far as possible to get a good ensemble.

On top of that, there are mainly two styles of producing base learners: a parallel style and a sequential style. In a parallel style, the generation of a base learner does not influence the generation of subsequent learners. In a sequential style, however, that influence does occur. Also, one may use multiple learning algorithms to produce heterogeneous learners, or just use one base learning algorithm to produce homogeneous base learners. [7]

The second step is to combine these base learners. This process is called combination scheme. Some examples of it are majority voting and weighted averaging. [7]

6.2 Imputation using GAIN

We tried imputation using Generative Adversarial Nets (GAN) framework. We got the idea from Yoon, J., Jordon, J., & van der Schaar, M. (2018). "GAIN: Missing Data Imputation using Generative Adversarial Nets.", tested it with

[7] Definition and explanation given by Zhi-Hua Zhou, from Nanjing University. https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/springerEBR09.pdf

[8] Things related to training data should be considered too.

[9] Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In Tesauro, G., Touretzky, D.S., Leen, T.K., eds.: Advances in Neural Information Processing Systems 7. MIT Press, Cambridge, MA (1995) 231–238

[10] Diversity of the base learners can be obtained from different channels. They include subsampling the training examples, manipulating the attributes, manipulating the outputs, injecting randomness into learning algorithms, or even using multiple mechanisms simultaneously. [7]

the help of example source file provided by author [11] , but implementing it into our code decreased the performance compared to our imputation algorithm used before, though difference was very slight. It is the suspected reason that the paper itself got comparatively better performance in terms of Area Under the Receiver Operating Characteristic Curve (AUROC), which is the same term as abbreviation AUC.

A receiver operating characteristics (ROC) curve is a two-dimensional depiction of classifier performance. To compare classifiers, we may want to reduce ROC performance to a single scalar value representing expected performance. A common method is to calculate the area under the ROC curve, abbreviated AUC (Bradley, 1997; Hanley and McNeil, 1982).[12]

In term of accuracy, we can interpret that performance difference yielded by this approach in the paper wasn't impactful enough to meaningfully alter the accuracy of our problem. Since our grading criteria is a function of accuracy and not a function of how well the imputations are done and there was a note that unique methods that have little contribution on accuracy lowers according to the criteria, we decided to use our original imputation approach, not this one for the submission.

[11] https://github.com/jsyoon0823/GAIN

[12] Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.