ORIE4741 Project Final Report

Used Car Pricing

Zixin Huang (zh378), Yuqi Wang (yw2288), Yue Han (yh893)

I. Introduction

Nowadays, many dealers have agents to help them estimate the price of used cars. The agents must have sufficient knowledge about the market and it usually takes months or even years for them to be experienced enough to make the estimations. What the agents usually do is to find transactions with similar year, make and condition for reference, usually taking much time. Meanwhile, when individuals would like to sell their cars, without much information about the market, they are confused about how to set an appropriate price. In this project, we would like to develop a model that helps to evaluate the prices of used cars for the dealers and individuals.

II. Data

Feature	Type	Missing	% Missing
url	object	0	0%
city	object	0	0%
city_url	object	0	0%
price	int64	0	0%
year	float64	1487	0.27%
manufacturer	object	26915	4.89%
make	object	9677	1.76%
condition	object	250074	45.44%
cylinders	object	218997	39.79%
fuel	object	4741	0.86%
odometer	float64	110800	20.13%
title_status	object	4024	0.73%
transmission	object	4055	0.74%
VIN	object	239238	43.47%
drive	object	165838	30.14%
size	object	366256	66.55%
type	object	159179	28.93%
paint_color	object	180021	32.71%
image_url	object	26	0.005%
desc	object	30	0.005%
lat	float64	11790	2.14%
long	float64	11790	2.14%

Fig. 1. Statistics of Features

We are going to use the dataset that includes the sales data from Craigslist, which is the world's largest collection of used vehicle sales. This dataset contains more extensive and comprehensive information compared to that accessible to the agents. It includes city, year, make, condition and some other variables that are critical when estimating the car price.

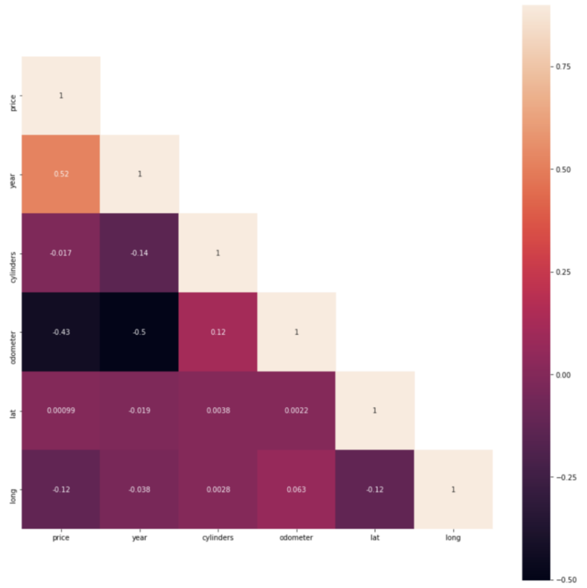

Fig. 2. Correlation between features

• Useless features: The features url, VIN, image url and desc are unique to each car. The city url does not contribute to this project. Also, we assume that paint color is not a significant factor for pricing. Thus, we also dropped these features.

• NaN value: We would like to drop the features that has more than 50% of NaN value. Size has the most NaN value, so we dropped it.

• Cylinders: The cylinder feature is of object type. We transformed it to integer.

• Price: By checking the price data, we observed some outliers such as 123,456,789. Based on common sense, we decided to drop the cars with price greater than 300,000.

• Odometer: Similar to price, odometer also has some outliers. We dropped the examples with odometer greater than 1,000,000.

• Year: We observed that most cars are produced after 1980. So we dropped the data of cars produced before 1980.

• State: We believe it will be sufficient to use the state as the indicator of location, so we extract a new column named "state" from the "city" column of the data set, which stands for the state which the city is located in. With examination we identify 180,000 missing values, which counts for almost 40% of the data. To deal with this, we notice the state can be inferred with the latitude and longitude of the location. We use the USA state latitude and longitude data set, and for each sample with missing state information, we find the state whose latitude and longitude are closest to those of the sample and use it as the inferred state. As a result, the number of missing values is reduced to 2,520, and we discard these samples.

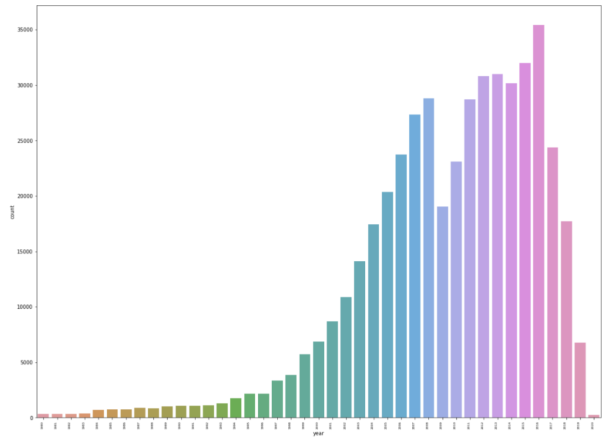

Fig. 3. Count group by state

A. *Feature Expansion: Household Income Data*

We believe the differences between the price levels of different states play an important role in car pricing, and household income is a major driver of the differences, so we decide to incorporate this data into our data set. Specifically, we use the latest median household income data for each state released by United States Census Bureau, and we map that to each sample with the state information processed before.

B. *Encoding*

There is a number of categorical data in our data set, so we need to encode them before further analysis. We apply one-hot encoding to columns “condition”, “cylinders”, “title status”, “type” and “drive”, and for columns “transmission” and “fuel” which primarily contain only two levels, we use a single integer to encode them.

III. Linear Models

Since the linear model is the simplest model, we choose to use the linear model first to see whether it is a good fit with our dataset. We separate 80% of the dataset into training set and 20% of the dataset into testing set so that we can evaluate the fitness of each linear model by calculating the corresponding train relative error (1-predicted y/y) on the training set and test relative error(1-predicted y/y) on the testing set. Moreover, we apply k-fold cross-validation on each linear model to ensure that all observations are used for both training and validation so that our estimates of relative errors will be more accurate. Specifically, we choose k=5.

A. *Elastic Net*

In statistics and, in particular, in the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods [1].

$$\hat{\beta} = \arg\min_{\beta}(\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|) \tag{1}$$

In our project, we apply grid search for parameter and use 5-fold cross-validation to estimate the train relative error and test relative error for this model. Specifically, we perform grid search for λ_2 with the range from 0.25 to 0.5 and λ_1 with range from 0.005 to 0.02.

B. *Bayesian Linear Regression*

Bayesian linear regression is an approach to linear regression in which the statistical analysis is undertaken within the context of Bayesian inference. When the regression model has errors that have a normal distribution, and if a particular form of prior distribution is assumed, explicit results are available for the posterior probability distributions of the model's parameters. [2]

The aim of Bayesian Linear Regression is not to find the single “best” value of the model parameters, but rather to determine the posterior distribution for the model parameters, as follows:

$$P(\beta|y, X) = \frac{P(y|\beta, X) * P(\beta|X)}{P(y|X)} \tag{2}$$

In our project, we use 5-fold cross-validation to estimate the train relative error and test relative error for this model.

C. *Summary*

We also fit the dataset with Lasso regression and Huber regression using 5-fold cross validation. The combined results are shown in the following table.

	Lasso with 5-folds	Huber with 5-folds	Elastic Net with 5-folds	BayesianRidge with 5-folds
Train Relative Error	0.4976	0.6160	0.4956	0.4976
Test Relative Error	0.4980	0.6100	0.4959	0.4980

Fig. 4. Summary of Linear Model Errors

From the table, Elastic Net with 5-folds, which has a test relative error of 0.4959, performs the best. However, since 0.4959 reflects that the accuracy of prediction by our model is only about 50%, we conclude that the linear models significantly underfit our dataset and therefore, we choose to use more complex models, nonlinear ones, to fit our dataset.

IV. Non-linear Models

As linear models stated above apparently under-fits the data, we can apply more complex non-linear models and make further analysis. Specifically, we choose to focus on tree-based methods, which are conceptually simple yet powerful tools common used in data science.

Tree-based methods partition the feature space into different regions, and then fit a simple model in each one [3]. The partition is based on a set of certain splitting rules that aim to maximize information gain. This process is repeated on each derived region in a recursive manner called recursive partitioning. The recursion is completed when the region at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions [4]. Fig. 5 shows an example of how prediction is made in a classification problem with a tree-based method.

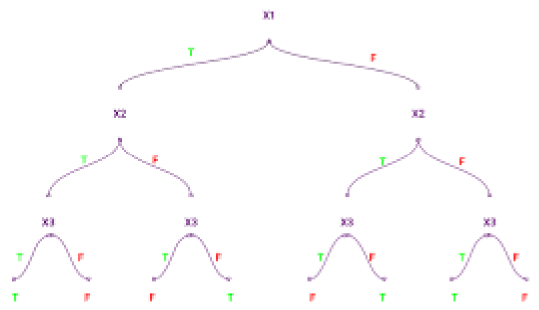

Fig. 5. Example of Using a Tree-based Method to Solve a Classification Problem [5]

With several significant advantages, tree-based methods are especially suitable for our project. First, tree-based methods are able to handle both numerical and categorical data at the same time [6]. Second, tree-based methods are white-box models and mirror human decision making more closely than other approaches, which aligns with our goal of modeling how an agent decides car prices. Third, tree-based methods perform well on large datasets (like our dataset) and are able to produce results with standard computing resources and reasonable time expenses [6].

A. Regression Tree

1) Description: As our project aims to solve a regression problem, a regression tree can be used. Mathematically, our dataset consists of p features of the car and one response y (i.e. the car price). There are N samples (x_i,y_i) for $i = 1, 2, ..., N$, with x_i = $(x_{i_1},x_{i_2},\dots,x_{i_p})$. The algorithm needs to automatically decide on the splitting variables and split points [3]. Suppose we partition the features of the cars into M regions $R_1, R_2, ..., R_M$, a regression tree predicts the price as a constant p_m in each region, that is

$$P(x) = \sum_{m=1}^{M} p_m I(x \in R_m). \tag{3}$$

If we use the quadratic loss function, i.e.

$$min\ L(y, P(x)) = \sum_{i=1}^{N}(y_i - P(x_i))^2, \tag{4}$$

it is easy to see that the best $\hat{p_m}$ is just the average of y_i in region R_m, i.e.

$$\hat{p_m} = avg(y_i|x_i \in R_m) \tag{5}$$

2) Parameter Tuning and Analysis of Results: The depth of the tree and the number of nodes are the parameters we can use as a form of regularization and avoid underfitting or overfitting. Specifically, we focus on two parameters — "max_depth" and "min_samples_split". The first parameter controls the maximum depth that the regression tree can grow to, and the second parameter refers to the minimum number of samples required to split an internal node. Overfitting occurs when "max_depth" is too large so the tree is too deep, or when "min_samples_split" is too small so the number of nodes is too large. On the contrary, underfitting occurs when "max_depth" is too small or "min_samples_split" is too large.

In our project, we apply grid search of parameters and use 5-fold cross-validation to choose the optimal parameters and evaluate whether underfitting or overfitting occurs. For "max_depth", we try 15, 20, 25 and 30; for "min_samples_split", we try 20, 30 and 40. The testing results are exhibited in Fig. 6.

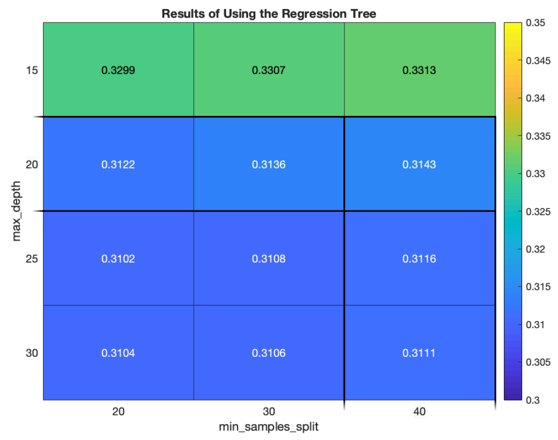

Fig. 6. Testing Results of Using the Regression Tree

As we can see from Fig. 6, the optimal parameters are 25 for "max_depth" and 20 for "min_samples_split". The best testing MRE is 0.3102. In addition, we can see from Fig. 6 that small changes to the parameters will not impact the prediction accuracy a lot.

For further analysis, we plot the histogram of the testing relative error under the optimal parameters as Fig. 7. As we can see, the relative errors of the predictions are concentrated around 0, and the frequency of occurrence decays as the relative error becomes bigger. Besides, the predictions are in general unbiased and not skewed.

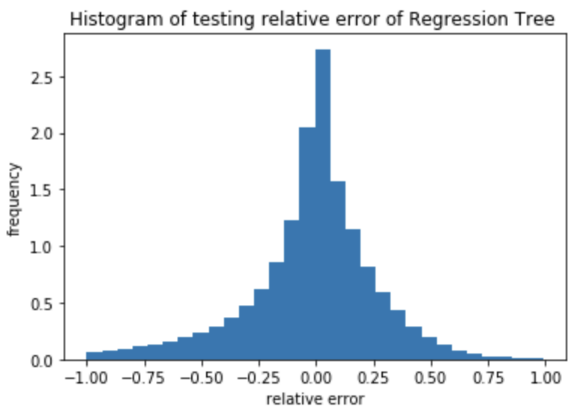

Fig. 7. Histogram of Testing Relative Error of Regression Tree

B. Gradient Boosting Regression Trees

1) Description: While the decision tree method effectively addresses the underfitting issue, we can still see much room for improvement. Boosting is one of the most powerful learning ideas that can be incorporated into regression trees, and these enhanced models are called Gradient Boosting Regression Trees (GBRTs). The motivation for boosting is to combine the outputs of many "weak" regression trees to produce a more powerful model [7].

Formally, a GBRT is a sum of regression trees induced in a forward stagewise manner, i.e.

$$GBRT_M(x) = \sum_{m=1}^{M} RT(x;\Theta_m), \tag{6}$$

where Θ_m refers to the parameters for the mth regression tree and x refers to the features of the car.

The algorithm that we use to grow a GBRT is called Forward Stagewise Additive Modeling, as show in Algorithm 1 [7].

Algorithm 1: Forward Stagewise Additive Modeling

```
1 Input: Features $x$, response $y$, objective function $J(\cdot)$,
    regression tree $RT(\cdot)$;
2 Initialize $GBRT_0(x) = 0$;
3 for $m = 1, 2, ..., M$ do
4     Compute
          $\hat{\Theta}_m = arg\min_{\Theta_m} \sum_{i=1}^{N} J(y_i, GBRT_{m-1}(x_i)+RT(x_i;\Theta_m))$;
      Set $GBRT_m(x) = GBRT_{m-1}(x) + RT(x_i;\Theta_m)$;
5 end
  Result: $GBRT_M(x)$
```

Among various kinds of implementations of GBRT, we choose three of them —XGBoost (eXtreme Gradient Boosting), Light GBM (Light Gradient Boosting Machine) and CatBoost (Gradient Boosting with Categorical Features Support). These three implementations are acknowledged to be among the state-of-the-art implementations of GBRT, and we believe they will tackle the challenges brought by the large size of our dataset. Specifically, XGBoost employs a set of advanced system optimizations and algorithm enhancements, which brings high model performances and makes it scalable on large datasets with high dimension [8]. Light GBM further improves the efficiency of XGBoost by using a novel technique for feature split called GOSS (Gradient-based One-Side Sampling), and it is proved to be able to achieve similar accuracy to XGBoost in most cases [9]. With 67% of the features of our dataset are categorical, CatBoost is an especially suitable model for us to use because of its expertise in handling categorical features. In particular, with modifications of classical gradient boosting algorithm, CatBoost fights the gradient bias brought by the discreteness of categorical features and improves the accuracy of prediction [10]. For more information, please refer to the cited materials.

2) Parameter Tuning and Analysis of Results: Each implementation of GBRT uses several parameters to control the model complexity and avoid underfitting or overfitting. Similar to the regression tree model, we perform grid search on the pivotal parameters of each implementation and use 5-fold cross-validation to find the optimal parameters and evaluate whether underfitting or overfitting exists.

For XGBoost, we focus on two parameters — "max_depth" and "min_child_weight". Similar to regression tree, "max_depth" refers to the maximum depth of a tree that is allowed, and we choose 6, 8 and 10 as candidates of this parameter. "Min_child_weight" refers to the minimum sum of instance weight needed in a child, and we choose 20, 40 and 60 as candidates. The testing results are exhibited in Fig. 8. As we can see from Fig. 8, the optimal parameters are 10 for "max_depth" and 20 for "min_child_weight". The best testing MRE is 0.2636.

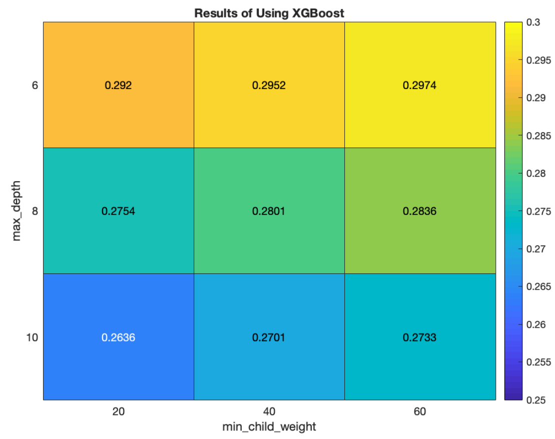

Fig. 8. Testing Results of Using XGBoost

For Light GBM, we focus on two parameters — "max_depth" and "min_child_samples". "Max_depth" has the same meaning as before, and we choose 6, 8 and 10 as candidates. "Min_child_samples" refers to minimum number of data needed in a child , and we choose 20, 40 and 60 as candidates. The testing results are exhibited in Fig. 9. As we can see from Fig. 9, the optimal parameters are 10 for “max_depth” and 20 for “min_child_samples”. The best testing MRE is 0.2529.

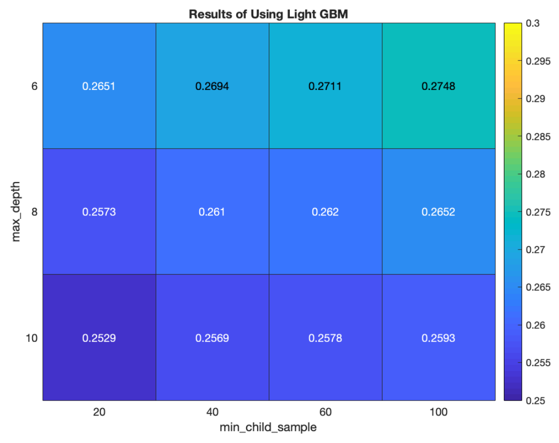

Fig. 9. Testing Results of Using Light GBM

For CatBoost, we focus on two parameters — “max_depth” and “reg_lambda”. “Max_depth” has the same meaning as before, and we again choose 6, 8 and 10 as candidates. “Reg_lambda” is the coefficient of the regularization term of the cost function, and we choose 5, 10, 20 and 50 as candidates. The testing results are exhibited in Fig. 10. As we can see from Fig. 10, the optimal parameters are 10 for “max_depth” and 50 for “reg_lambda”. The best testing MRE is 0.238.

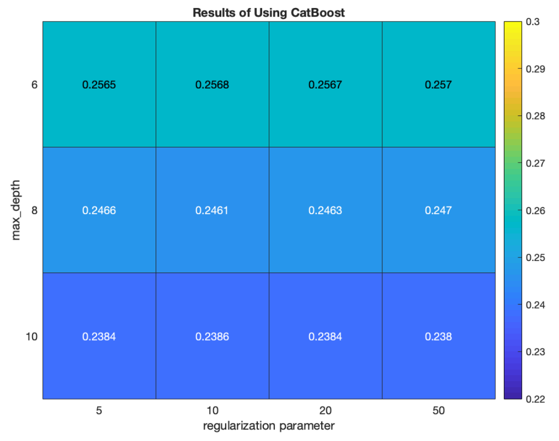

Fig. 10. Testing Results of Using CatBoost

As we can see from Fig. 8, 9 and 10, small changes to the parameters will not impact the prediction accuracy a lot. Comparing the results, we can see the prediction produced by CatBoost has the highest accuracy. This is understandable since a large proportion (67%) of the car features are categorical and CatBoost is advantageous in handling categorical data.

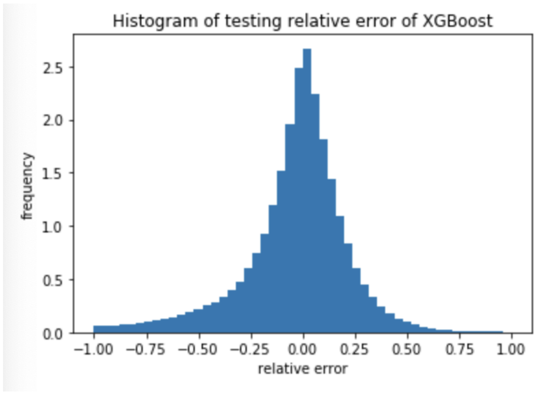

Fig. 11. Histogram of Testing Relative Error of XGBoost

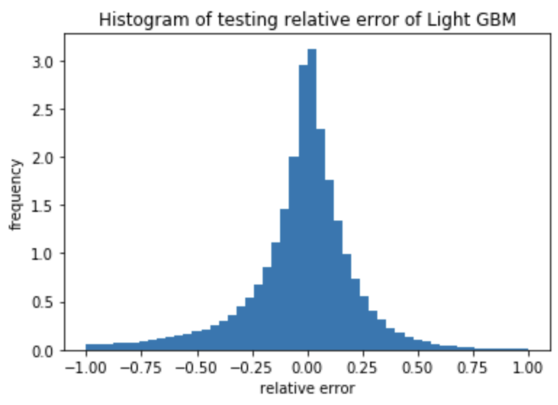

Fig. 12. Histogram of Testing Relative Error of Light GBM

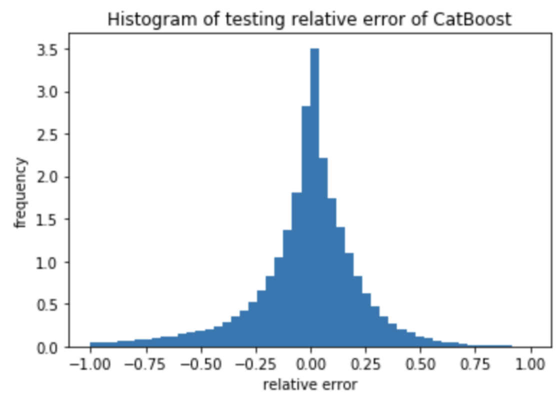

Fig. 13. Histogram of Testing Relative Error of CatBoost

Furthermore, we plot the histograms of the testing relative error for the three implementations as Fig. 11, 12 and 13. As we can see, the relative errors of the predictions are concentrated around 0, and the frequency of occurrence decays as the relative error becomes bigger. With careful observation and comparison, it is not hard to see CatBoost has the best predictive power. In general, the predictions have fatter left tail, which means underestimation of prices may happen more frequently than overestimation. However, generally no biases can be observed from the predictions.

3) Further Improvement: Fit Individual Models for Each Car Type: To further improve the predictive power of our model, we notice that the type of a car is a vital factor that we should consider. This is not only because for different types of cars the average prices are different, but more importantly, people attach importance to the features of a car in different ways if the car types are different. To elaborate, it is common that a sedan drive prefers automatic transmission to manual transmission because of the convenience, so a sedan automatic transmission might have higher price compared with a similar one with manual transmission. However, a truck driver may actually prefer manual transmission because it saves the fuel, so the price of a truck will not decrease for using manual transmission. Therefore, we consider fit individual models for each car type.

Specifically, we use GBRT to build models for each car type respectively, because GBRT is the most promising model we have used so far. Note that for cars whose type is unknown, we will not fit individual models and simply apply the predictions derived above. Similar to our previous approaches, we apply the three implementations of GBRT — XGBoost, Light GBM and CatBoost at the same time. We apply grid search to the pivotal parameters and use 5-fold cross-validation for evaluation. Considering we have 13 types in total, the sample size will decreases significantly if we train individual models for each type, so we are more cautious about the risks of overfitting and select smaller candidates for tree depth and bigger ones for regularization parameters.

For XGBoost, we find under the optimal parameters, the testing error is 0.2625, and Fig. 14 is the histogram of the testing relative error. For Light GBM, we find under the optimal parameters, the testing error is 0.2515, and Fig. 15 is the histogram of the testing relative error. For CatBoost, we find under the optimal parameters, the testing error is 0.2345, and Fig. 16 is the histogram of the testing relative error. We can observe slight improvements for all implementations. In particular, the GBRT implemented with CatBoost has the best predictive power, and Fig. 16 shows its predictions are very concentrated around 0 with thin tails.

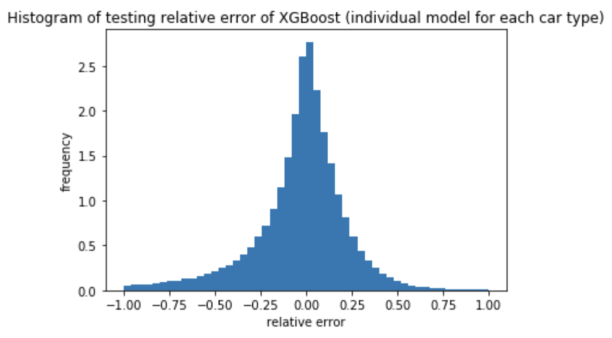

Fig. 14. Histogram of Testing Relative Errors of XGBoost with Individual Modeling for Each Car Type

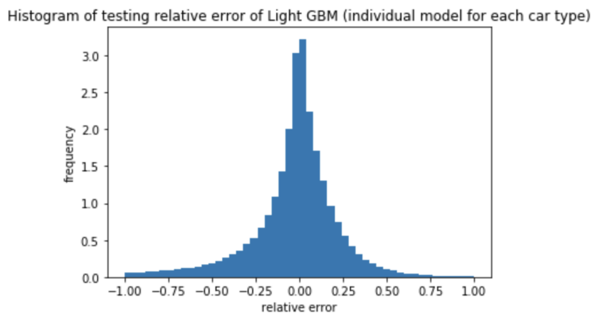

Fig. 15. Histogram of Testing Relative Errors of Light GBM with Individual Modeling for Each Car Type

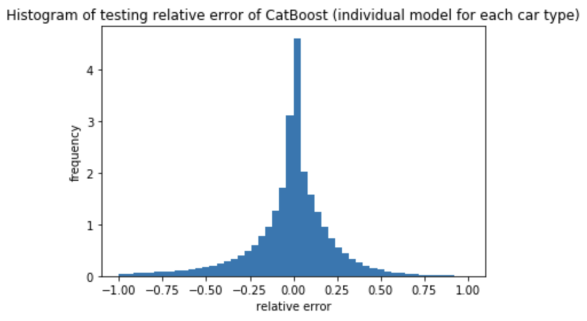

Fig. 16. Histogram of Testing Relative Errors of CatBoost with Individual Modeling for Each Car Type

C. Random Forest

1) Description: Random forests construct a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. [11] They correct for decision trees' habit of overfitting to their training set. [12]

2) Parameter Tuning and Analysis of Results: Features: The features we choose for training random forest including the most important features such as year, cylinder, odometer and one-hot encoded condition. We also included the mean and median of other numerical values in our model. Specifically, we normalized the odometer parameter by dividing by 10,0000 based on its distribution.

Parameter Tuning: We are going to select the optimal parameter for the random forest classifier. Since the dataset is very large, we performed grid search cross validation on max_depth, min_samples_split, max_features and min_weight_fraction_leaf one by one for the first 20000 samples, each time searching for the optimal hyperparameter based on the previous results we get. The final results are:

parameter	**value**
max_depth	8
min_samples_split	3
max_features	None
min_weight_fraction_leaf	0.001

parameters for random forest

Training to predict null type: When applying the random forest classifier to our dataset, we choose train different models based on their types.

There are 119387 samples of NaN type in total. For this subset, we are going to use the whole dataset as training set and predict the price of the NaN type with the training model.

Due to the large size of the sample, we will split the data to 10 folds. Then we train a model on each fold and predict the price. Afterwards, we take the median of the 10 estimation as our final prediction.

Trial	CV score	training accuracy
1	0.38544	0.54945
2	0.38647	0.47979
3	0.39468	0.44754
4	0.36924	0.50630
5	0.37370	0.45881
6	0.39688	0.45638
7	0.39731	0.44295
8	0.38642	0.49373
9	0.39732	0.46963
10	0.37173	0.50074

training statistic

Training to predict other types: For each other type, we are going to train them separately. First we need to know the sample size for each type. There are some types with relatively large sample size such as SUV and sedan. It is difficult to train a model with more than 70,000 samples. Therefore, we are going to use the stack method we used for NaN type again here. By observing the sample sizes, if the size is greater than twice of 37500, we are going to train multiple models. Otherwise, we are going to train the model with size min(37500, sample size).

Type	Sample Size	training accuracy
pickup	41553	0.28468
truck	38316	0.33002
SUV	77034	0.30136, 0.29881
hatchback	10657	0.23124
Van	8415	30120
sedan	75382	0.30649, 0.30138
coupe	16174	0.42203
wagon	9373	0.25042
Mini-van	6283	0.27175
other	9476	0.31031
convertible	7437	0.30011
bus	528	024059
offroad	700	0.14126

training statistic

Combining the prediction: After training and predicting the price for all the types, we may have multiple predictions for a single sample. Therefore, we finalize our prediction by selecting the median of all prediction on a sample. The overall in-sample accuracy based on our method is approximately 0.33821 for random forest.

3) Further Improvement:

1. Due to the limitation of running such a huge dataset on laptop, it's difficult for us to run the model on the whole dataset. Therefore, we use stacking so that almost all the samples are used in the training. We may try to run random forest on the whole dataset if there is a capable server.

2. In order to train random forest models for the 13 known type, we wrote a single loop for all known types, each model with the same hyperparameter. However, Since the sample sizes of some types differs significantly from each other, the hyperparameter might not be optimal for a particular model. It needs more time and effort to adjust the hyperparameter for every single model.

V. Model Ensemble

Model ensemble methods use multiple algorithms to obtain better predictive performance than that could be obtained from any of the constituent models alone [13]. Being one of the most commonly used ensemble methods, Stacking combines the predictions of other learning algorithms by training a combiner algorithm using all these existing predictions as inputs [14]. Stacking adds to the complexity of the overall model and causes high risk of overfitting, hence we usually use simple model or algorithm in this step.

In our project, we select 17 models with the best performance from various kinds of non-linear models we have built, i.e. Regression Trees, GBRTs and Random Forest. We apply Stacking technique to the predictions of these models and see if we can bring the predictive power to a new level.

Stacking works the best when the input predictions have comparable performance but imperfect correlations between each other. Therefore, we plot the histogram of the correlations between the predictions as Fig. 17. As we can see, most of the correlations are below 0.98 and we have some especially low correlations as below 0.85, which indicates the high probability that Stacking will bring enhancements.

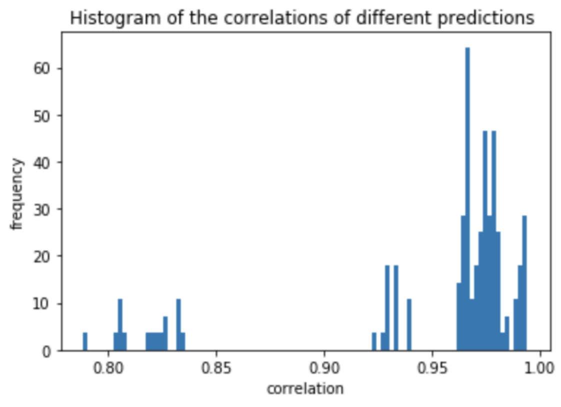

Fig. 17. Histogram of Testing Relative Error of XGBoost with Individual Modeling for Each Car Type

We first use some of the most basic methods, that is

1) Simple average of all predictions.
2) Median of all predictions.

3) Weighted average of all predictions, and the weight of each prediction is decided by the reciprocal of its testing error.

Beyond that, we try two linear models — Ridge Regression and Elastic Net Regression. We use 50% of the data for training and others for testing, and gird search is applied to find the optimal parameter.

The testing error of the five methods are as follows:

stacking method	**testing error**
Simple averaging	0.2328
Median	0.2281
Weighted averaging	0.2321
Ridge regression	0.2304
Elastic net	0.2304

testing error

As we can see, using the simple median is the most effective ensemble method in our project, and the improved testing error is 0.2280, which is lower than that of any individual model we have built so far. It is not surprising that median performs well, since it introduces some non-linearity and can eliminate potential abnormal (extremely large or small) predictions.

VI. Out-of-sample Evaluation

To evaluate how our aggregated model generalizes on out-of-sample data, we use the model to make predictions for the out-of-sample dataset that we separated at the beginning. These data have not been seen by our model before. The out-of-sample testing error is 0.24, which is comparable with the in-sample testing error 0.2280. For out-of-sample data points with no missing values, the testing error is 0.20. Moreover, we plot out the histogram of the out-of-sample testing relative errors as Fig. 18, and we can see the distribution of the errors is similar to what we get for in-sample tests. Therefore, we can safely conclude that no significant overfitting can be spotted and our model can generalize well on out-of-sample datasets.

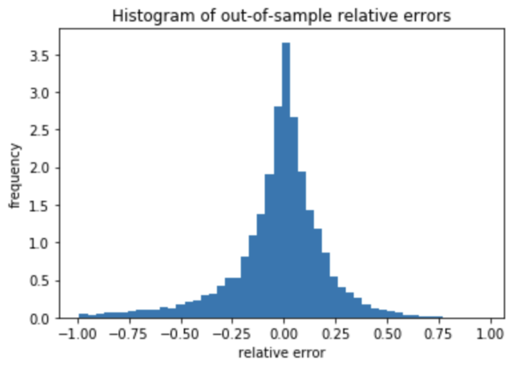

Fig. 18. Histogram of Out-of-sample Testing Errors

VII. Further Discussions

A. About the Weapon of Math Destruction

We feel quite confident that our project does not create a Weapon of Math Destruction. First, the accuracy of the predictions can be easily measured by comparing them to the true market prices. Second, we have a large amount of data and more data will be continuously available, since the used-car market is a big one and new deals are being made every day. We will be able to compare our predictions of new deals with the true prices and use the results as new feedbacks, with which we can continuously evaluate and improve our model. This clearly does not create any feedback loop. Finally, with the knowledge of economics, we know that the price of a used-car is decided simply by the market condition and the features of the car itself, since a deal is always made at a price that both the seller and the buyer are satisfied with. The predictions provided by our model can only be used as a reference and do not have the ability to influence the market prices or damage the interests of anybody.

B. Fairness

When we implement the algorithms to investigate big data problem, the available information can have bias, which may lead to wrong results and unintended negative consequences. Therefore, it is essential to evaluate the fairness for our model. By definition, a protected attribute is a feature on which discrimination is prohibited, such as race, national origin, sex, age, disability, etc. Fairness can be evaluated by unawareness. A classifier is unaware of the protected attribute if the prediction is independent of the protected attribute given other covariates. Since the attributes of our dataset do not relate with discrimination, the fairness topic is out of scope for our project. Generally, regression does not involve fairness problem in most cases, while classification does.

VIII. Summary and Outlook

In this project, we have cleaned the data and applied multiple models to the dataset in order to predict the used car price. Through the analysis of our models, we achieved nice predictions of car price. We believe that these models will benefit the dealers in car pricing. Moreover, we are expecting more data that will help to improve our existing the model as well as more models that could produce better results.

References

[1] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
[2] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011, vol. 40.
[3] J. Friedman, T. Hastie, and R. Tibshirani, “The elements of statistical learning,” in, 10. Springer series in statistics New York, 2001, vol. 1, ch. 9.
[4] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
[5] *Decision tree tutorials notes: Machine learning*. [Online]. Available: https : / / www . hackerearth . com / zh / practice / machine - learning / machine - learning - algorithms/ml-decision-tree/tutorial/.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[7] J. Friedman, T. Hastie, and R. Tibshirani, "The elements of statistical learning," in, 10. Springer series in statistics New York, 2001, vol. 1, ch. 10.

[8] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: Extreme gradient boosting," *R package version 0.4-2*, pp. 1–4, 2015.

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[10] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: Gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[11] T. K. Ho, "Random decision forests," *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pp. 278–282, 1995.

[12] H. TK, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[13] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[14] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.