Used Car Price Prediction based on Multiple Machine Learing Models

Xu Bokai Michael
Data Science
BNU-HKBU United International College
m730014072@mail.uic.edu.hk

***Abstract*—Used cars have been on a global increase these years, so it is of a great need to have an effective used cars price prediction system. This project aims at predicting used cars price by using linear models, simple neural network, tree-based models and ensemble learning and compare the performance of different models according to 4 commonly used regression evaluation indexes based on an open dataset in Kaggle.**

***Index Terms*—EDA, Linear Models, Tree-based Models, Ensemble Learing**

Supplementary Materials

Data and codes are at: https://github.com/MichaelJoker/DS-4023-Machine-Learning

I. Introduction

Automatic industry is a very essential component of world market. In automatic industry, there are two important components: new cars and used cars. The prices of new cars in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. However, due to the increased price of new cars and the incapability of customers to buy new cars due to the lack of funds, used cars sales are on a global increase. It is very common to observe that the used car sales surpasses the new ones across the globe. For example, according to OICA statistics there were around 17.5M new vehicle sales in 2017 in the USA. On the other hand, used vehicle market had a size of approximately 39.2M in the same year according to an Edmunds report [1]. Therefore it is important to study the used-car market where price is one of the most important determinants of sales.

Different factors such as the condition of the vehicle and the branch of vehicle can affect the price of the used cars. However, determining a reasonable price based on these factors of used cars is a great challenge. There is a need for a used car price prediction system to effectively determine the worthiness of the car using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best.

In this case, we need to use some models to help for the decisions. Current works mainly use one model to make prediction. Although some of the models get quite good results, it is a limit of choice. Due to these reasons, I will improve current works by trying some new models and make comparisons among them to help used-car website set a more reasonable price.

II. Related Works

Traditionally, many works like [2], [3] have already sucessfully applied regression-based methods for prediction. Also, recent years, as for the improvement of open-sourced machine learning framework, many people are trying more models such as random forest [4] and even the back propagation nueral network [5]. And lots of them have done some great jobs. However, the models are still limited, so my work is based on curretn studies, to try more machine learning models.

III. Exploratory Data Analysis (EDA)

A. Dataset and Features

The dataset is from the used cat dataset in Kaggle, which contains every used vehicle entry within the United States from a well known used-car website called Craigslist. The dataset is updated every few months, and the latest version I used is version 5, which was updated on March, 2020. The whole dataset has more than 500 thousand entries, and there are totally 25 columns , with one column ”price” as the target variable and other 24 columns as the input features. You can download the dataset here.

B. Preprocessing

As the data is scrapped from the website, so it’s more dirty than other commonly used dataset. So, the first step is to clean the data like dropping some unnecessary columns, filling the null values and so on. The first thing in the Preprocessing step is to count number of null values for each column. From Fig. 1, we can see that two columns called county (100%) and size (68.77%) have more than 50 percent null values, if an attribute has more than half of null values, then we can think that this column has not enough infomtion and is meaningless, and we can drop this column.

After looking through the details of meanings of each attribute, we can further drop more columns. Table I shows the detailed columns that should be dropped.

After dropping the columns, we need to fill the missing values and do some further preprocessing. For example, as [6] indicates, the used-car market in USA begAN in 1900 and surged in 1960, in this case, we shall analyze the year between 1960 and 2019. Also, there are some prices that are

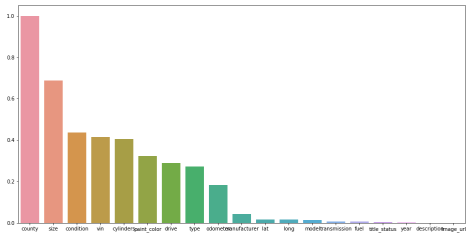

Fig. 1: Number of Null Values for Each Column

Columns	Descrtption
id	entry id
url	listing URL
region_url	online description of region
image_url	online photo of the car
description	description of a car
vin	vehicle identification number
title_status	status of the car (only one value)
model	specific car type for each manufacture, it's rare for a customer to see this attribute when buying either new car or used cars

TABLE I: Dropped Columns with Description

extremely high, as the average price of used car in America is around $25,000, so I selected interval between $400 and $40,000. What's more, there are some categorical attributes, and I filled the null and missing values with Unknown or -1.

C. Visualization

After preprocesse the data, next, I will do some visualzations. Firstly, let's see where are these used cars from. Fig. 2 shows the locations of used car in USA. I sampled 50,000 entries from the dataset and marked their locations as red points.

Fig. 2: Used Car Locations in USA

Next is to show some visualzations about the input features. Fig. 3 shows the Top 15 regions, manufacture and states that have sold used cars during these years. Fig. 4 shows the percentage of different values in different categorical attributes. Fig. 5 shows the sales of used car in each year. Finally, there are 3 figures (Fig. 6) from the left to right showing the relations between odemeters and price, year and average price, and manufacture and average price respectively.

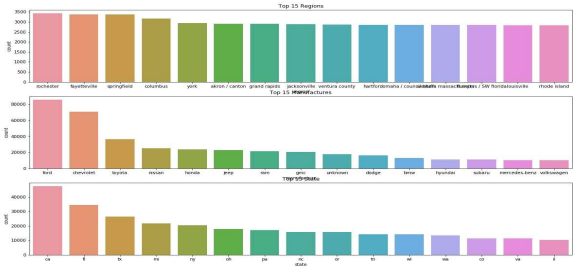

Fig. 3: Top 15 Regions, Manufacture and States

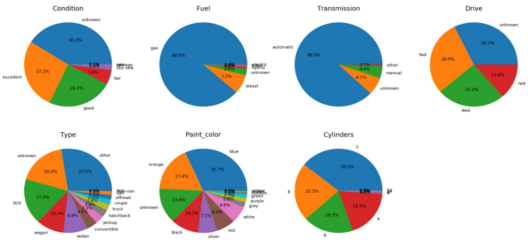

Fig. 4: Percentage of Different values in Different Categorical Attributes

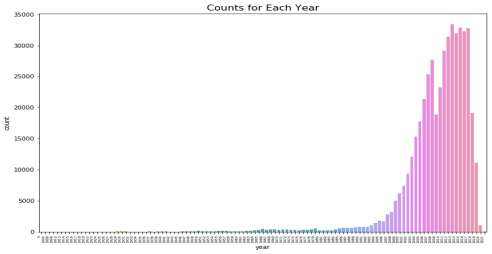

Fig. 5: Sales of Used Car in Each Year

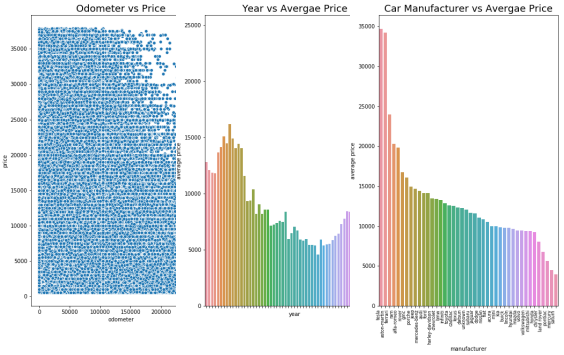

Fig. 6: Relations between Some Attributes and Price

IV. Methodology

There are more 60 models that can be applied to regression prediction probelms. In this project, I have tried some of them

and grouped them into four different categories: regression models, Neural Network, tree-based models and Ensemble models.

A. Linear Models

1) ***Linear Regression:*** Linear Regression is a traditional model in regression problems, and it has already been covered in our lectures. In this case, I will use it as my baseline model in this project. The objective function of Linear Regression is:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 \tag{1}$$

And one way to solve it is to use gradient descent method which we have already learned in class.

2) ***Support Vector Regression:*** Support Vector Regression (SVR) is a variation of Support Vector Machine (SVM). The basic idea of SVR is similar to SVM, with only slightly differences. In SVR, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance ϵ is set in approximation to the SVM which would have already requested from the problem. The primal problem of SVR is:

$$\begin{aligned} &\min \tfrac{1}{2}\|\mathbf{w}\|^2 + C\textstyle\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ &\text{s.t.} \begin{cases} y_i - \mathbf{w}^T\mathbf{x}_i - b \le \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - y_i \le \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{2}$$

Specifically, I will use Linear kernel to implement as this will be faster to achieve.

3) ***Ridge Regression:*** Sometimes the attributes have some multicollinearity, then in this case when solving the linear regression problem by least-square method, the matrix (X^TX) will inevitably be singular. Therefore, in this case, we cannot directly use $\theta = \left(X^TX\right)^{-1}X^Ty$ to get θ. Ridge Regression, which was firstly proposed by Horel and Kennard in 1970 [7], can help handle multicollinearity. Buy adding L2-regularization, Ridge regression gives the objective function in (3):

$$\|X\theta - y\|^2 + \|\Gamma\theta\|^2 \tag{3}$$

If we define the $\Gamma = \alpha I$, then by using least-square method, we can get:

$$\theta(\alpha) = \left(X^TX + \alpha I\right)^{-1}X^Ty \tag{4}$$

By choosing different α, until the function value is stable. Ridge regression is a complement to least-squares regression, which loses unbiasedness in exchange for high numerical stability, thus obtaining higher computational accuracy.

B. Neural Network

Multilayer perceptron (MLP) is a static neural structure composed of successive layers which communicate and exchange information through synaptic connections represented by an adaptative weight. The structure of multilayer network contains an input layer which is made of number of perceptions equal to the number of data attributes, on the other hand, output layer includes one perceptron in the case of regression or more when it is a task of classification and in this case the number of perceptron is equal to the number of classes to predict, all the other layers are considered as hidden [8].

Multilayer perceptron Regression (MLP Regression) is one type of MLP that optimizes the squared-loss using LBFGS (Limited-memory BFGS). The equation for MLP Regression is:

$$Y = a + b_1x_1 + b_2x_2 + \cdots + b_nx_n \tag{5}$$

It trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting [9].

C. Tree Models

1) ***Decision Tree Regression****:* Decision tree, is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. Decision tree regression is one type of decision tree algorithms that are specialized for regression problem. It used ID3 algorihtm that can be applied to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction [10]. It is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero. The Standard Deviation between target variable and predictor can be calculated as follows:

$$S(T, X) = \sum_{c \in X} P(c)S(c) \tag{6}$$

Where T is the target variable and X is the predictor, P is the probability of the predictor belongs to a specific value c and S is the standard deviation of c. Following is the procedure of Decision tree regression:

1) Calculate the standard deviation of the target.

2) The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

3) The attribute with the largest standard deviation reduction is chosen for the decision node

4) The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.

*2) **Random Forest Regression**:* Random forests [11] is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random Forest is a bagging method. Compared with traditional bagging method, Random forest not only bootstrap for the samples but also bootstrap for the input features, as we have already learned in class. The algorithms are shown as follows:

Algorithm 1 Random Forest

Input A training set S := $(x_1, y_1), \cdots, (x_n, y_n)$, features F, and number of trees in Forest B

H $\leftarrow \phi$
for $i \in 1 \cdots B$ **do**
  $S^{(i)} \leftarrow$ A bootstrap sample from S
  $h_i \leftarrow RandomTreeLearn(S^{(i)}, F)$
  $H \leftarrow H \cup h_i$
end for
return H

Algorithm 2 RandomTreeLearn

Input A training set S := $(x_1, y_1), \cdots, (x_n, y_n)$, features F

At each node:
  $f \leftarrow$ very small subset of F
  Split on best feature in f
return the learned tree

*3) **Gradient Boosting**:* Gradient boosting [12] is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The intuitively understanding of gradient boosting is: There is a residual difference between the prediction and the actual value in each iteration. The prediction is carried out in the next iteration according to the residual. And finally, all the predictions are added up to get the result. To be more specific, let us consider a gradient boosting algorithm with M stages. At each stage m ($1 \geq m \leq M$) of gradient boosting, suppose some imperfect models F_m. In order to improve F_m, our algorihtm should add some new estimator $h_m(x)$. So,

$$F_{m+1}(x) = F_m(x) + h_m(x)$$

The goal of it is to find the best F_m such that:

$$F_m(x) = F_{m-1}(x) + \arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(x_i\right) + h_m\left(x_i\right)\right) \right] \tag{7}$$

for each iteration. And the objective function of it is:

$$\arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(x_i\right) + h_m\left(x_i\right)\right) \right] \tag{8}$$

The algorithm goes as follows:

Algorithm 3 Gradien Boosting

Input training set with $\{(x_i, y_i)\}_{i=1}^{n}$. differentiable loss function $L(y, F(x))$, number of iterations M

Initialize model with a constant values:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

for $m = 1$ to M **do**
  1. Compute so-called pseudo-residuals

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n$$

  2. Fit a base learner (or weak learner, e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $r_{im})\}_{i=1}^{n}$
  3. Compute multiplier γ_m by solving the following Line search—one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)$$

  4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

end for
return $F_M(x)$

*4) **eXtreme Gradient Boosting**:* eXtreme Gradient Boosting (XGBoost) [13] is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way. XGBoost is a modified version of gradient boosting by adding the regularization and model complexity term to the objective function, so the objective function of XGBoost is:

$$\arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(x_i\right) + h_m\left(x_i\right)\right) \right] + \sum_{k} \Omega(f_k) \tag{9}$$

where $\sum_k \Omega(f_k)$ is the regularization term, A function representing the complexity of the tree. The smaller the value,

the lower the complexity, and the stronger the generalization ability, the expression is:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{10}$$

Where T is the number of leaf nodes, and w is the number of nodes. Intuitively, the target requires that the prediction error should be as small as possible, and the leaf node T should be as small as possible, and the node value w should be as moderate as possible.

Although XGBoost seems similar with gradient boosting. However, it indeed enhance the performance from both system aspect. The enhancement for system achievement includes:

1) **Parallel Calculation:** XGBoost approaches the process of sequential tree building using parallelized implementation. This is possible due to the interchangeable nature of loops used for building base learners; the outer loop that enumerates the leaf nodes of a tree, and the second inner loop that calculates the features. This nesting of loops limits parallelization because without completing the inner loop, the outer loop cannot be started.

2) **Tree Pruning:** The stopping criterion for tree splitting of Gradient Boosting is greedy and depends on the negative loss criterion at the point of split. XGBoost uses max-depth parameter as specified instead of criterion first, and starts pruning trees backward. This depth-first approach improves computational performance significantly.

3) **Hardware Optimization:** This algorithm has been designed to make efficient use of hardware resources. This is accomplished by cache awareness by allocating internal buffers in each thread to store gradient statistics. Further enhancements such as 'out-of-core' computing optimize available disk space while handling big data-frames that do not fit into memory.

D. Ensemble Methods

*1) **Bagging**:* Bootstrap aggregating (Bagging), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach. The algorithm goes as follows:

*2) **Adaptive Boosting Regressor**:* An AdaBoost regressor [14] is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases. Algorithm goes follows:

Algorithm 4 Bagging

Input $D = \left\{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$, Base learning algorithm $\mathcal{L}$, and Number of learning rounds T

1: **for** t = 1 to T **do**
2: $D_t = \text{Bootstrap}(D)$
3: $\mathcal{L}(D_t)$
4: **end for**
5: **Output:** $H(x) = \underset{y \in \{+1,-1\}}{\text{argmax}} \sum_{t=1}^{T} \mathbb{I}\left(y = h_t(x)\right)$

Algorithm 5 AdaBoosting

Input a $N-$ sample $S = \langle (x_1, y_1), \ldots, (x_N, y_N) \rangle$, a distribution D on S, a learning algorithm $\mathcal{A}$, an integer T

1: Initialize the weighted vector $w_i^1 = D(i)$ for $i = 1, \ldots, N$
2: **for** t = 1 to t **do**
3: 1. set $\mathbf{p}^{(t)} = \frac{\mathbf{w}^{(t)}}{\sum_{i=1}^{N} w_i^{(t)}}$
4: 2. Choose randomly sample $S^{(t)}$ with distribution $P^{(t)}$ from Sample; call the learning algorithm A and get the hypothesis $h_t = A_{S_{(t)}}$
5: 3. Calculate the error: $\epsilon_t = \sum_{i=1}^{N} p_i^{(t)} \left|h_t\left(x_i\right) - y_i\right|$
6: 4. Calculate $\beta_t = \frac{\epsilon_t}{(1-\epsilon_t)}$
7: 5. Set the new weights vector to be : $w_i^{(t+1)} = w_i^{(t)} \beta_t^{1-\left|h^{(t)}(x_i) - y_i\right|}$
8: **end for**
9: **Ouptut:** the hypothesis

$$h_f = HS\left(\sum_{k=1}^{T}\left(\log\frac{1}{\beta_t}\right) h_t(x) - 1/2 \sum_{k=1}^{T}\left(\log\frac{1}{\beta_t}\right)\right)$$

*3) **Stacking Methods**:* Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.

E. Evaluation Index

In regression problems, there are four commonly used evaluation indexes: explained variance, mean absolute error, root mean square error and the coefficient of determination (R^2)

*1) **Explained Variance**:* In statistics, explained variation measures the proportion to which a mathematical model accounts for the variation (dispersion) of a given data set. The formula is given by:

$$\text{Explained Var} = 1 - \frac{Var(y - \hat{y})}{Var(y)} \tag{11}$$

The max values is 1 and the nearer to 1, the better the model.

*2) **Mean Absolute Error (MAE):*** MAE is used to describe the difference between a predicted value and a true value. The smaller the number, the better the modle. Although the mean absolute error can obtain a evaluated value, you do not know whether the value represents the good or bad fitting of the model without comparison. The formula is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{12}$$

*3) **Root Mean Squared Error (RMSE):*** RMSE is used to measure the deviation between the observed value and the true value. RMSE is very sensitive to the very large or very small errors in a set of measurements, so RMSE can well reflect the precision of the measurement. The formula is given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{13}$$

*4) **R2:*** Also known as the coefficient of determination. What is judged is the degree of fitting between the prediction model and the real data. The optimal value is 1, and it can be negative at the same time. The formula is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{14}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

V. Experiments and Results Analysis

A. Model Preparation

1) Label Encoder: As some columns contains categorical values, we cannot directly use these values to train the models. we need to map the categorical labels to continuous values. One way to do this is by using LableEncoder function in python scikit-learn library.

2) Standard Scalar: Because in the original data, the scope of each variable is very different. For some machine learning algorithms, if not been standardized, the objective function will be unable to proper operate. Therefore, all features should be standardized. Another reason to do feature scaling is that it accelarate the speed of gradient descent method. In python scikit-learn library, module Preprocessing has already provide StandardScaler function for convenience.

B. Setting Parameters

In machine learning, it's of great importance to fine-turning the parameters in order to get some better results.

1) Linear regression: Use default parameters.

2) Stochastic Gradient Descent Regression: A special type of linear regression that uses Stochastic Gradient Descent method, using default parameters.

3) LinearSVR: Use grid search and find best Control coefficient C = 3.3.

4) Ridge Regression: Use cross validation and find the alpha = 10.

5) MLP Regressor: By using grid search method for training 70 minutes, get the parameters in Table II:

Parameter	Value
activation	relu
alpha	0.0001
early_stopping	True
hidden_layer_sizes	19
learning_rate	constant
learning_rate_init	0.01
max_iter	1000
power_t	0.5
solver	adam
warm_start	False

TABLE II: Parameters for MLP Regressor

6) Decision Tree Regressor: Set max depth of the tree equals 10.

7) Random Forest Regressor: Use grid search and after 18 minites training, get the best number of estimators is 300 with max depth of trees 20.

8) Gradien Boosting: By using hyper optimization [15] (link) with 5 times evaluations, after training for around 20 minitues, get the best number of estimators is 338 with max depth of trees 2. The reason to use HyperOpt is that it uses Distributed Asynchronous Hyper-parameter Optimization, which is faster than grid search function in scikit-learn library.

9) XGBoost: After using grid search and GPU accelaration for training, getting the parameters: learning_rate: 0.1, max_depth: 7, n_estimators: 140, reg_lambda: 0.5.

10) Bagging and AdaBoosting: Use default parameters.

11) Stacking: Consider the outputs from Linear regression, SDGRegressor and ridge regressor as input for the second models. Then using pretrained models Random Forest, Bagging, Boosting and MLP Regressor respectively as second models to predict the values.

C. Result

To accelarate the training speed, I randomly selected 100 thousand entries and use a 80-20 criterion to split the data into training set and testing set. Table III and IV show the 4 evluation indexes for different models of training set and testing set respectively.

Training Set Result				
Model	EVS	MAE	RMSE	R2
Linear Regression	0.3853	4859.83	6699.26	0.3853
LinearSVR	0.2366	4341.66	7624.8	0.2037
SGDRegressor	0.3853	4860.45	6699.26	0.3853
Ridge Regressor	0.3853	4859.87	6699.26	0.3853
Bagging	**0.9804**	**710.25**	**1195.67**	**0.9804**
AdaBoost	0.6278	5385.5	6235.36	0.4675
stack_bagging	0.5897	3999.44	5475.68	0.5893
stack_boosting	0.5876	4755.33	5840.6	0.5328
stack_rf	0.583	4028.21	5524.59	0.582
stack_mlp	0.5829	4019.17	5631.21	0.5657
MLP	0.7864	2765.9	3949.1	0.7864
Decision Tree	0.8326	2464.94	3496.42	0.8326
Random Forest	0.9746	889.43	1362.4	0.9746
Gradient Boosting	0.8193	2548.04	3632.09	0.8193
XGBoost	0.8898	1954.43	2836.02	0.8898

TABLE III: Training Set Result

Testing Set Result				
Model	EVS	MAE	RMSE	R2
Linear Regression	0.3469	4940.68	6846.84	0.3467
LinearSVR	0.1712	4375.65	7839.59	0.1435
SGDRegressor	0.347	4941.27	6846.51	0.3467
Ridge Regressor	0.3469	4940.71	6846.84	0.3467
Bagging	0.8637	2016.57	3128.14	0.8636
AdaBoost	0.6146	5417.19	6290.15	0.4486
stack_bagging	0.5614	4092.37	5611.08	0.5612
stack_boosting	0.5715	4821.38	5917.84	0.5119
stack_rf	0.56	4094.32	5624.67	0.5591
stack_mlp	0.5611	4068.16	5700.79	0.5471
MLP	0.7782	2777.04	3989.17	0.7782
Decision Tree	0.8003	2628.08	3785.55	0.8003
Random Forest	**0.8657**	**2012.86**	**3104.29**	**0.8657**
Gradient Boosting	0.8046	2606.67	3744.42	0.8046
XGBoost	0.8138	2556.6	3602.66	0.8093

TABLE IV: Testing Set Result

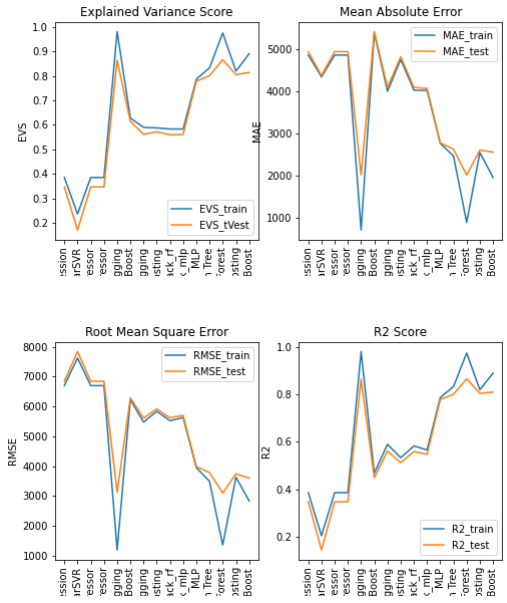

Fig. 7: Scores of Different Mdoels based on 4 Evaluation Indexes

Also, to have better intuition of the performance of different models, I drawed (Fig. 7) the scores of different mdoels based on 4 evaluation indexes.

From the results above, we can see that Linear Models does not give satisfied results. One thing to notice is Ridge Regression has the very similar result as Linear Regression as there are no multicollinearities among different features, So in this case, Ridge Regression is exactly Linear Regression.

For Tree models, as we can see that they both get high scores in the training set but lower scores in the testing set, that is: the models have overfitting problems. One possible reason behind it is that Tree models cannot predict values that out of the range of target values in the training dataset, that is, it can only do interpolation, not extrapolation.

And for Bagging, the value is extremely high, although overfitting problem still happens. For AdaBoost, performance is not quite good, but no overfitting problem happens. And for 4 stacking models, we can see that compared with three weak learners, stacking models does improve a lot, but the results are also not quite good.

VI. Conclusion and Future Work

In conclusion, I have compared different models based on 4 evaluation indexes. Linear models are simple, however, it is not quite adequite for predition on this dataset. Neural Network model is good, but needs quite a long time training and needs high computational resources. Tree-models are even better, but have overfitting problem. Ensemble learning have uncertain performance based on the choice of models. Therefore, in the real-life problem, we cannot depend on only one of few specific models, we need to compare different models using some high-level techniques.

In the future, I will try some methods like early stopping and data augmentation to avoid overfitting problems. In addition, I will try some more high-level ensemble learning methods to get a better performance. Also, I will try some deep neural models to find whether deep learning can apply to my dataset or not.

References

[1] A. Demiriz, "Used car pricing and beyond: A survival analysis framework," pp. 65–68, 2018.

[2] N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya, and P. Boonpou, "Prediction of prices for used car by using regression models," in *2018 5th International Conference on Business and Industrial Research (ICBIR)*. IEEE, 2018, pp. 115–119.

[3] M. Listiani, "Support vector regression analysis for price prediction in a car leasing application," Ph.D. dissertation, Citeseer, 2009.

[4] N. Pal, P. Arora, P. Kohli, D. Sundararaman, and S. S. Palakurthy, "How much is my car worth? a methodology for predicting used cars' prices using random forest," in *Future of Information and Communication Conference*. Springer, 2018, pp. 413–422.

[5] S. Gongqi, W. Yansong, and Z. Qiang, "New model for residual value prediction of the used car based on bp neural network and nonlinear curve fit," in *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, vol. 2. IEEE, 2011, pp. 682–685.

[6] D. Burgess Wise, *Classic American Automobiles*. [Don Mills, Ont.]: Nelson Canada, 1980.

[7] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[8] C. Arouri, E. M. Nguifo, S. Aridhi, C. Roucelle, G. Bonnet-Loosli, and N. Tsopzé, "Towards a constructive multilayer perceptron for regression task using non-parametric clustering. a case study of photo-z redshift reconstruction," *arXiv preprint arXiv:1412.5513*, 2014.

[9] B. Choubin, S. Khalighi-Sigaroodi, A. Malekian, and Özgür Kişi, "Multiple linear regression, multi-layer perceptron network and adaptive neuro-fuzzy inference system for forecasting precipitation based on large-scale climate signals," *Hydrological Sciences Journal*, vol. 61, no. 6, pp. 1001–1009, 2016. [Online]. Available: https://doi.org/10.1080/02626667.2014.966721

[10] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[14] H. Drucker, "Improving regressors using boosting techniques."

[15] J. Bergstra, D. Yamins, and D. D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” 2013.