Predicting the Resale Value of Used Cars: A Data-Driven Approach

Jiehan Wu (jw2568), Di Wang (dw624)

Table of Content

Abstract

With the difficulty of summarizing all the available information in the used car market for both dealers and customers, this paper aims at building a satisfying general predictive model to predict the value of used cars in the US market. By scraping used car data from truecar.com, cleaning the data and comparing the performance of different models, we reached the conclusion that random forest is the most suitable tool for predicting used car prices.

1. Introduction

This project aims at providing a tool to quantitatively predict the resale value of used cars with different attributes. The predictive model should come in handy for all the used car market participants, including the buyers and the dealers or other types of sellers: it will help dealers with better pricing based on the overall market, and make buyers become more informative about their buying decisions.

2. Data

We scraped our data off truecar.com through modifying available data scraping code online(Reinderien). Specifically, the data focuses on used car sales all over the United States. The whole dataset consists of 18,498 rows of data entries in total, including the following 13 entries:

- Categorical values: make[1], model, engine type
- Continuous values: price, mileage
- Discrete values: owner count, year, accident count
- Boolean values: is clean title (whether the car is lemon or is salvaged)
- Nominal values: exterior color, interior color, stype, location

It is worth noting that the pricing data we obtained is the current market selling price on truecar.com, and when making predictions, the given result will also be based on this benchmark and thus will be more comparable to the overall resale value on truecars.com.

We splitted the data randomly into 80% and 20%, for training and testing purposes respectively, and compare the performance of three different models: linear regression, random forest,and decision tree.

3. Data Cleaning

In this section, we will assess the quality of the dataset and conduct any necessary steps for data cleaning. We want to focus on removing errored data(outliers), removing or back-filling empty entries, and converting entries that have too many unique nominal values such as "style" to encoded values for easier processing.

3.1. Outliers

To start off, we want to remove the outliers in our price and mileage data to prevent overfitting in our training. After checking,

[1] Please note that the list for the "make" entry above is not an exhaustive list of every brand of car in the market. Rather, it is a compilation of the most commonly traded cars for generalization purposes.

mileage data does not seem to have too many crazy data points, while price data do have a couple of outliers as shown in figure 1. We limit the ceiling for price, and removed 29 data points (0.16% of total data) which are mostly luxury cars and are not meaningful for generalization.

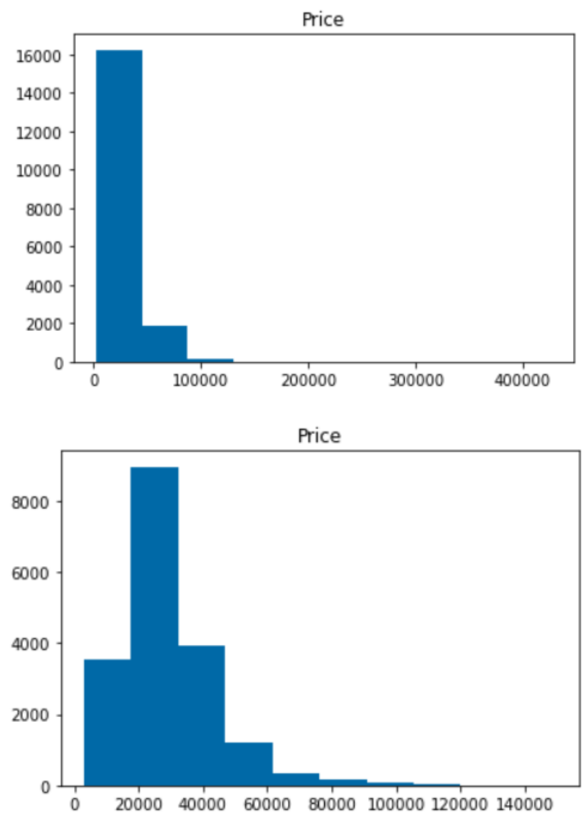


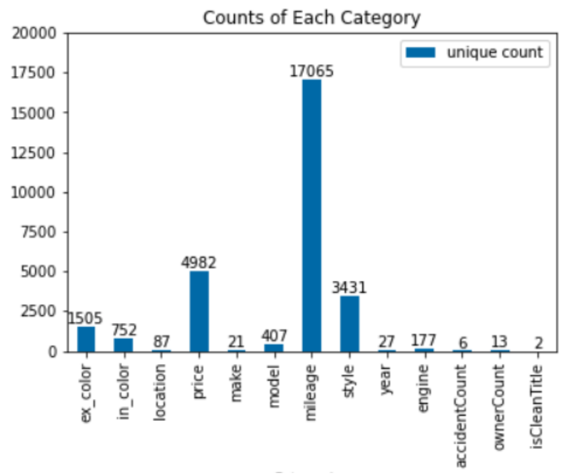
Figure 1. Comparison between pricing data with outliers (top) and without outliers (bottom)

3.2. NA values

For empty entries, we found 224 NA values in pricing data, 118 empty values in styles, and 626 unknown engines with every other column having none missing. This shows that the completeness of the dataset is pretty high. Since we have a relatively tiny amount of data missing comparing to the total number of rows of information we have, we decide to simply remove those rows with NA values. After removing rows that have missing values, we are left with 17509 rows of data, which is still a good amount to train and test on.

3.3. Nominal values

We plotted the bar plot of the count of each columns' unique entries in Figure 2, and noticed that exterior color, interior color, price, model, mileage, style and engine are the aspects that have comparatively more entries. Among those, price and mileage are continuous values and can be directly used as quantitative input for our predictive function, and thus will not pose any difficulty in analysis. On the other hand, other entries are nominal or categorical values, and we want to make some updates for easier generalization. For example, engine is converted to the size of the engine (discrete values). We converted colors such as "white metallic", "revolution white" all to just generic "white", and converted all the uncommon colors such as green and purple to "other" category. This helped us greatly reduce the unique nominal values in exterior color and interior color entries. In order to convert all those nominal values into numerical values, we used the encoding technique. For the style column, since truecars.com put values such as "manual", "EX-L AWD" and "AWD 4dr T6" in those entries, we removed some informations that are too specific to certain cars, and compressed the entries to be only "manual", "automatic", "AWD" and "FWD", and used many hot encoding on these entries. For the make and model columns, we used the label encoder in skilear.processing and converted all the unique entries in make and model column to an indexed number for easier processing. For location column, 43% of the data has ambiguous or inaccurate location information and thus might not provide reliable information for the training purpose, and it could potentially introduce noise or biases if the location entry is included. After attempting to assigning alternative values such as zip code or state code, we realized that it is extremely difficult with the ambiguous location data provided[2], and thus we decide to remove this column.



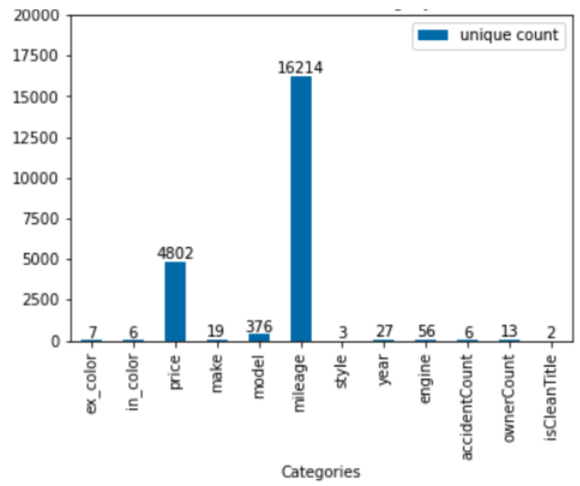


Figure 2. Unique Count of Each Data Category before (top) and after processing (bottom)

Other than the features above, we added an offset term 1 to the features.

[2] For example, there are a lot of entries called "middletown", and there are just way too many states has the city middletown.

4. Data Visualization

After the data is cleaned, we attempt to visualize the data. First, we will use the correlation matrix to have an overview on the data.

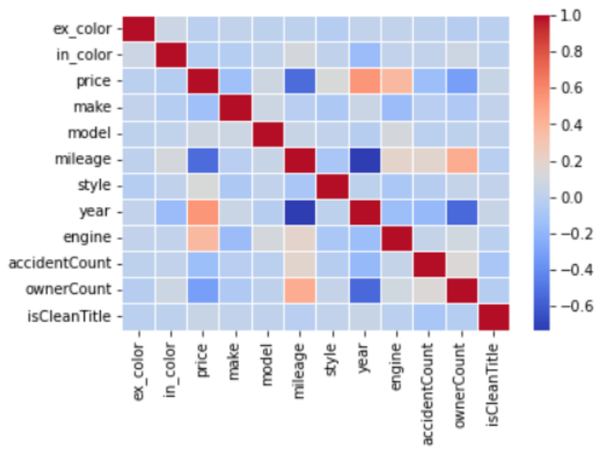


Figure 3. The correlation matrix for different attributes

The correlation matrix offers valuable insights and helps in predicting car prices. For example, analyzing the price column reveals that mileage and owner count have a negative correlation with the price, while the year of production and engine type have a positive correlation. This implies that newer cars are more expensive, while cars with more owners and more miles traveled are less expensive. These observations align with our common expectations. Thus, we can conclude that mileage, year, engine, accident count and owner count are the factors that play a bigger role in the pricing of the cars.

To go into individual attributes that are mentioned above, we took accident count as an example. From Figure 4, we color coded the cars with less accidents with lighter color, and cars with more accidents with darker ones, and it has a visible trend that the dark colored dots are sinking more in the bottom of the graph comparing to the lighter colored dots, suggesting that cars with less accidents are more valuable in the market. Another more commonly known trend is that cars that has higher mileage generally have a lower price, which is verified by Figure 5.

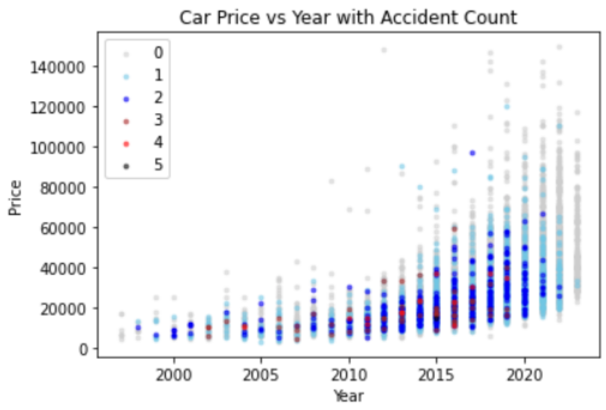


Figure 4. Scatter plot of car price vs year with accident count

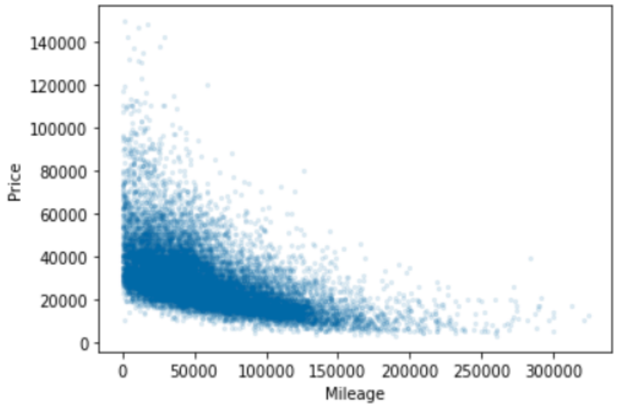


Figure 5. Scatter Plot of Price and Mileage

5. Analysis and Conclusion

5.1. Linear Regression

We had our initial assumption from experience that car prices are possibly linearly related to their mileage and year, since from our visualization we can see a clear trend that when the mileage are higher, the price of the car are usually lower; on the other hand, when the year is higher, usually the price of the car is higher. Starting from this assumption, we decide to first pick linear regression to fit our data.

In linear regression, we want to minimize the mean square error of the model, which gives the following objective function:

$$\min \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

After using the statsmodels.api package in python, we were able to run OLS and find the MSE for the linear regression model. The predicted pricing data is shown in Figure 6, with a training RMSE of 9217.71, and a test RMSE of 9583.48. This model has training adjusted R square of 0.567, and testing adjusted R square of 0.565, which means that the model is able to explain 56.7% of in sample data and 56.5% of out of sample data.

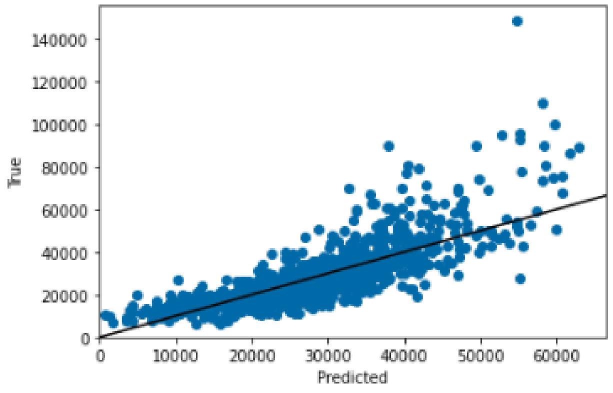


Figure 6. Actual pricing data against predicted price from linear regression

This result is somewhat satisfying since it was able to explain more than half of the data, but adding in regularizors might be able to better capture the pattern in high dimension datas. In attempt to achieve better result, we decide to add in regulization in the process as demonstrated in the following.

5.2. Linear Regression with Regularization

For regularization, we will use the two most common method: Ridge and Lasso.

5.2.1 Ridge Regression

Ridge adds a shrink penalty to the linear regression model and will attemp to shrink all the coefficients as close to 0 as possible. It estimates the predictor w using the values that minimize the following objective function:

$$\min \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda ||w||^2$$

Lambda in the penalty term determines the relative importance of the shrink penalty comparing to the original objective function of the linear regression, which is the MSE of the model. In order to find out the most suitable lambda, we used the built in GridSearchCV in sklearn.model_selection and cross validation, and found out the optimal lambda = 1.

We ran the ridge regression with the Ridge() function in sklearn.linear_model. This yielded a training RMSE of 8866.40, a testing RMSE of 8474.01. With ridge regression, around 61.2% of the in sample data can be explained, and 62.8% of out of sample data can be explained by the model.

5.2.2 Lasso Regression

Comparing to ridge, lasso has feature selection and will force some parameters to 0, and will reduce model complexity and overfitting. With this property, lasso has the following objective function:

$$\min \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda \sum_{i=1}^{n} |w_i|$$

Similar as in Ridge regression, the lambda here is the determinant of how severe the penalty would be comparing to the MSE the model achieves. After running the cross validation, we found out that the optimal lamda = 0.62.

Lasso is built in sklearn.linear_model package in python. After running lasso regression, we get a training RMSE of 8866.40, and a test RMSE of 8474.11. This model ends up with a training adjusted R square of 0.612, and testing adjusted R square of 0.628, which means that predictor is able to explain 61.2% of in sample data and 62.8% of out of sample data.

5.3. Random Forest

All the models above assumed a linear relationship between the independent variable and the pricing data, while this is not necessarily true. Another issue is that there might be interactions between independent variables. For example, there is a reasonable relationship between mileage and year: cars from more recent year would tend to have less mileage. In order to address the possibility of non-linear relationships and the interactions between features, we decide to implement the random forest model.

The random forest model stacks together a bunch of decision trees that learns from a subset of data. We removed the offset term for random forest implementation since the offset term is only used for linear regression and will not provide any information at each decision tree branch. In order to implement this, we used the random forest regressor in sklearn.ensemble. We set the number of decision tree estimators to be 100, and this achieved a train RMSE of 1817.74 and a test RMSE of 5312.92. This is the best fit so far, with the model being able to explain 98.3% of the training data and 86.6% of the testing data, which is satisfying performance.

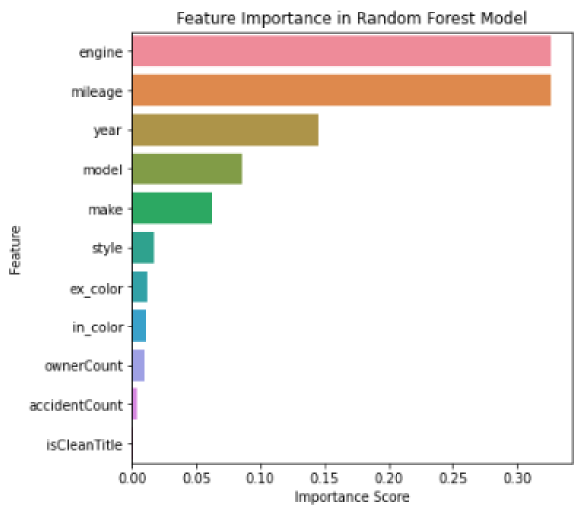


Figure 7. The importance of each feature in random forest

Figure 7 illustrated the importance of each feature after implementing random forest, and from the graph we can see that engine type, mileage and year plays the most important role in predicting the price, while if the car has a clean title seem to not matter at all. This might be due to the fact that there are very few cars that are marked as not having clean title, and thus this feature become less important in the big picture.

5.4. Summary and conclusion

To have a more clear comparison between all the models above, we summarized the RMSE and adjusted R square for each model into the following table:

Model	Train RMSE	Test RMSE	Train R square	Test R square
Linear	9217	9583	56.7%	56.5%
Ridge	8866	8474	61.2%	62.8%
Lasso	8866	8474	61.2%	62.8%
Random Forest	1818	5313	98.3%	86.6%

The RMSE for ridge and lasso is identical here due to the number of decimals we chose to kept, while in reality, ridge had slightly better performance if we compare their MSE[3]. Overall, random forest has the best performance since it has the lowest RMSE and highest adjusted R square. This is due to its ability to address non-linear relationships between the features and the pricing data, and to capture the interactions between features. Additionally, random forest also provide an estimate of the importance of different features. It is worth noting that even thought all the models finished running within seconds, random forest did run slightly slower than the other models due to the computational complexity, which is a worthy trade-off given the much more accurate result we obtained from random forest.

6. Discussion

With the results being stated above, there are certain limitations and space for further research for this project.

[3] We ran the code several times to split the data differently into training and testing dataset, and the difference in MSE for ridge and lasso differ each time, but generally ridge performs better than lasso by some amount.

6.1. Limitations

6.1.1 Bias from data sourcing

One such limitation is that the data used for model training was scraped from the truecars website, which may not provide truly fair and objective pricing for used cars. This is because sales are not guaranteed to occur at the listed prices, and there may be mispricing. Therefore, it is important to acknowledge the potential for bias in the model and to consider alternative sources of data in future research. However, it should be noted that despite the limitations, the model still provides a useful benchmark based on the currently available pricing in the market.

6.1.2 High dimensional dataset

We used label encoder for a lot of our nominal data columns during data cleaning process, and this resulted in around 500 dimensions of data. Usually this will cause computational complexity and might require more powerful algorithms. However, in our case, the linear regression model (with and without regularization) and the random forest model all finished running in a few seconds, which means that the OLS model, ridge, lasso regression and random forest are able to handle our current dimension of dataset. Have this caused an issue, we might need to consider changing the label encoder to many hot vectors or even one hot vectors.

6.2. Future Improvements

6.2.1 Alternative source of location data

In the data cleaning stage, we removed the location data since 43% of the columns includes ambiguous and inaccurate data. If time allows, some further data scraping can be done to collect more specific location data, and incorporate this factor into the model training. After all, location might be a quite important factor in valuating the car prices.

6.2.2 Time series analysis

The car pricing is sensitive to seasonal factor. For example, there are certain period of the year when used cars tend to be cheaper. In our project, we basically assumed that the car pricing are static and would not move in time, while in reality time would be an important factor. In future research, rolling windows and time series analysis might be incorporated to count in this factor.

6.2.3 Economic regime effect

This project is done in April, 2023. Bearing this in mind, the economy is just recovering from the hit of COVID. The data we have obtained from the truecars.com might be under the influence of economy crisis during 2022-2023, which means the result might not be applicable under different economy regimes.

7. References

Reinderien (Reinderien). "Web Scraping Cars." Code Review Stack Exchange, 23 Dec. 2020, https://codereview.stackexchange.com/questions/259549/web-scraping-cars.

TrueCar. (Scraped Data.) TrueCar.com. Accessed April 20, 2023. https://www.truecar.com.