How Much Does a Craigslist Car Cost? Predicting Price Using Car Listing Data

Anthony Fong
UC San Diego
San Diego, California
ajf010@ucsd.edu

Jasraj Johl
UC San Diego
San Diego, California
jsjohl@ucsd.edu

Alex Pham
UC San Diego
San Diego, California
alp075@ucsd.edu

1 Introduction

A car is one of the most important assets someone living in America can own. With the lack of reliable public transportation in many cities it is imperative that an individual buys a car for transportation. Though it can be difficult to determine how much a car is worth both to the seller and buyer. Some individuals take exceptionally good care of their property and may want that perk to be reflected in the price, whereas some users may just want something cheap and affordable regardless of the state the product is in.

There are websites such as Kelly Blue Book which can give an estimate of how much a car is worth, but this may not always be accurate as the descriptions are for the most part very general. These general descriptions may not encompass how much a seller believes their car is worth, or how much a buyer is willing to pay for a car owned by a respectable driver. As such we have decided to create a predictive algorithm that predicts a car's price based on a given set of features.

By using car sales data from Craigslist we can determine how much users generally sell their cars for given features unique to their car. Since Craigslist offerings are all from real individuals with taste based preferences, the prices are more reflective of what the general public is willing to pay as opposed to some calculated price available on sites like Kelly Blue Book.

Using the performance metrics of RMSE and R^2, we have determined that the best model for predicting car prices on Craigslist is the random forest regressor. This model had an RMSE of 4881.25 and an R^2 of 0.8214. These values far outclass the baseline, linear regression, and k-neighbors regressor model.

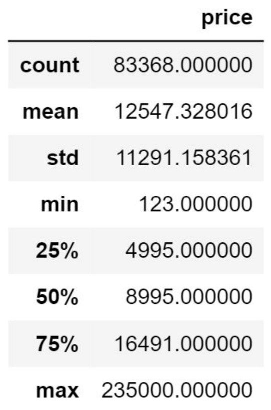
2 Dataset

This dataset was acquired from Kaggle [1]. It contains Craigslist listings of cars for sale from users. We chose to filter the data to a subset of reasonable features like price, odometer(mileage), manufacturer, model, state, and others. Some features were not included as they were either already represented by other columns or not relevant to our purposes: image_url, id, lat, VIN, region, and etc.

To clean the dataset, we removed null values and also filtered prices to exclude outliers and any vehicles with a price of less than 100 dollars.

To get a better understanding of our domain and what features may be important, we performed some exploratory data analysis.

First, we can get some useful statistics of our target, price.



	price
count	83368.000000
mean	12547.328016
std	11291.158361
min	123.000000
25%	4995.000000
50%	8995.000000
75%	16491.000000
max	235000.000000

Figure 1

It is clear that the most expensive cars are outrageously priced compared to the rest of the dataset. The difference between the 25th percentile and median is much smaller than the difference between the 75th percentile and the max.

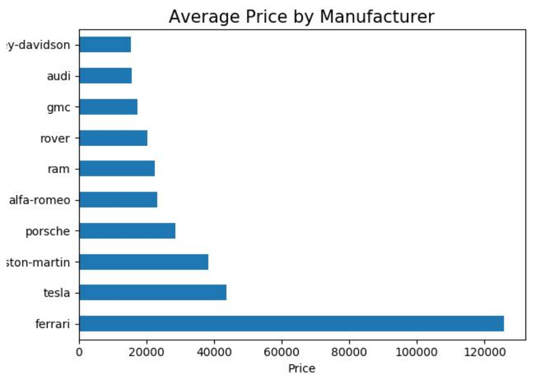


Figure 2

Figure 2 shows the top ten manufacturers based on average price. It makes logical sense that Ferrari, Tesla, and Aston-Martin are the top. It also shows that manufacturers would indeed be useful in predicting price.

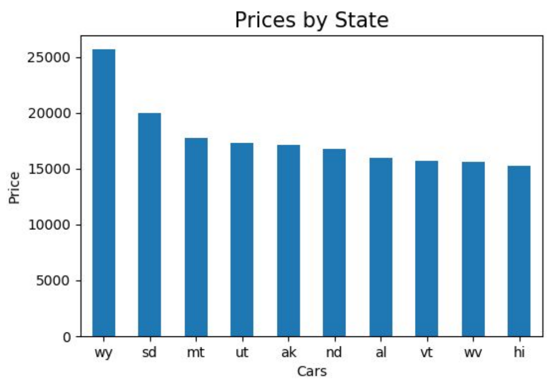


Figure 3

Figure 3 shows the top ten states based on their average price. We should also consider whether or not the state has an effect on the price. This makes logical sense as the cost of living and minimum wage can differ greatly by state.

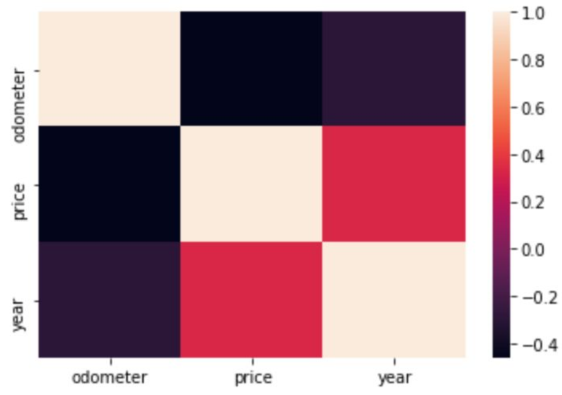


Figure 4

A heatmap is a useful tool to understand the correlation between features. Here, we can see that year correlates with price, but odometer may correlate less. This defies common sense, as the mileage of a car should have a strong effect on its pricing. We can investigate this further.

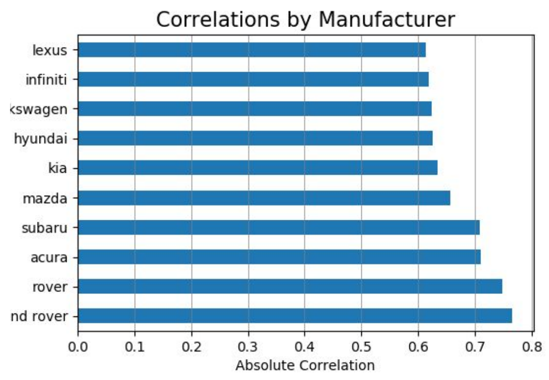


Figure 5

Figure 5 displays the top ten manufacturers based on correlation. By looking at the correlation between odometer and price within each manufacturer group, we can better understand the correlation. Most of the manufacturers have a correlation of 0.5 or higher. This is not as high as we would expect, but it shows that there is a moderate to high correlation. This means that odometer would not be a useless feature.

There also existed other categorical features that we did not directly plot. These features are intuitive to determine whether or not they would indicate price: condition, cylinders, transmission, drive, type, and etc.

3 Predictive Task

For this model, the predictive task we wanted to perform was to predict craigslist car listing prices based on previous information given in other craigslist car listings. To be able to predict the price of a car listing, we wanted to get as much relevant information as features for the task. We simplified our dataset to contain the following features: car year and odometer number, which are both quantitative feature, and categorical features of the car manufacturer, model, condition, number of cylinders (with unknown numbers as a zero vector), fuel type, title status, transmission, drive type, car type, paint color, and state of listing. We felt these features would be helpful in determining what price to sell a car as they are relevant to the status and stature of the car as well as the cost of living of a state. The price of the listings was used as the y value of training that will be predicted.

To get these features in manners that would be useful for our model we transformed them. Both year and odometer values were each min-max normalized to put them in more usable formats. The rest of the features were each one hot encoded to allow all the categories to get represented in similar manners, with the exception of car models, which was one hot encoded only for models that appeared over 100 times. This was done to limit the amount of features that would be needed with this one hot encoding, while ensuring that the more relevant car models would be recognized and evaluated by our predictive model.

To evaluate our predictive task, we would be using Root Mean Square Error (RMSE) and R^2 as metrics, and use these to compare. RMSE would help us compare between the different models, how good our predictions are as it decreases as the error for each prediction decreases, and will use R^2 to help determine the validity of our baselines as it would show the correlation between our prediction and real values, with it increasing as the correlation improves. We will evaluate our model by comparing each models' RMSE and R^2 scores, and determine which model is the best when they are each optimized and compared against each other. RMSE and R^2 score are found using the following formulas:

$$RMSE = \sqrt{\frac{\sum_{I=1}^{N}(y_i - \widehat{y}_i)^2}{N}} \qquad (1)$$

$$R^2 \text{ Score} = 1 - \frac{\sum_{I=1}^{N}(y_i - \widehat{y}_i)^2}{\sum_{I=1}^{N}(y_i - \overline{y}_i)^2} \qquad (2)$$

Where:

N: number of samples

y_i : the ith value of the actual values

$\widehat{y_i}$: the ith value of the predictions

$\overline{y_i}$: the mean value of actual value

To compare our model we used a global baseline to predict the average of all listing prices for every prediction. We then used more in depth models to compare as strong baselines and determine our strongest model for this predictive task. We used a linear regression model, a k-nearest neighbors regressor, and a random forest regressor as these models could return quantitative values as we are trying to predict in this task. With these models, we hoped to be able to get a good model to evaluate these tasks as we used a linear, instance-based, and tree method that would provide different avenues for predicting the price values.

4 Model

Four models were developed to address our predictive task. The first is a baseline model which always predicts average price, the second is linear regression, third is random forest, and our fourth model is k neighbors regressor.

4.1 Baseline

The baseline model chosen for this predictive task was a simple mean prediction model. This model would retrieve the mean price of the training data, and predict that price for every entry regardless of what feature values those entries contained. This model as expected did not perform very well scoring the highest RMSE and lowest R^2. We still, however, choose this model as our baseline since the performance metrics we are using rely on the distance between predicted points and the actual points. As such if you are planning to randomly guess values, then the mean should be the closest value to any point.

4.2 Linear Regression

The next model we employed was linear regression. This model was used as a secondary baseline. Whereas the trivial baseline was a model used to gauge performance on our predictive task as a whole, linear regression was used as the model to gauge performance on our task as a regression problem. Linear regression is the most basic model to be used in regression tasks. The model is linear and as such utilizes the basic equation of $y = mx + b$. The model requires that several assumptions about the data be true. The data must have a linear relationship, there must be multivariate normality, little collinearity, no auto-correlation, and the data must be homoscedastic. Linear regression creates predictions based on these assumptions which are not always valid.

This model has very little parameters to change and instead was optimized using standard ablation and double ablation study approaches where certain features were left out on different runs of the model to see what combination of features produced the best results. We also tried to improve this model by including additional car models. The model improved as more car models were included up to any car model that appears at least 5 times. Though with each additional car model included the training time increased. Through the ablation study it was determined that linear regression performed best when all features were present.

This model did not have scalability issues. The strength of a linear regression model is that it can be trained and tested relatively fast. The model is simple enough where it is easy to calculate any predictions made by the model by hand if needed. The issue with this model, however, is that it is overly simple and relies heavily on assumptions about the data. As a model linear regression is prone to overfitting, but through the ablation study and exploratory data analysis we believe we have removed any features that would cause this issue. This model performed quite a bit better than the trivial baseline as expected.

4.3 Random Forest Regressor

The random forest regressor model was the first model we created that was not used as a baseline. This model works by averaging the predictions made by multiple decision trees hence the name random forest. A decision tree creates a prediction by splitting nodes from a root based on the value of a feature. At each point the decision tree continues to make splits until the leaf nodes are reached which will determine what prediction is made. Each decision tree in the random forest draws from a subset of the actual data and generates splits based on the subset it has access to. This prevents the random forest from overfitting as each tree in the forest is only trained on a random subset of data.

The intuition behind this model is that the price of a car can be determined based on the answers to a few questions. These questions make up the splits of the decision trees, and given enough features or questions each decision tree would be able to make accurate predictions as to what the price of a car is. Taking the average of the predictions should then lead to a good predictor overall.

This model was tuned using a standard grid search approach. The parameters that were hypertuned in this model were n_estimators and max_depth. The n_estimators parameter determines how many trees should be in the random forest while the max_depth parameter decides how large each tree is. The grid search found that the optimal value was 16 trees with a depth of 36 leaves. The quantitative features were standard scaled, and the car model features were removed as well.

The random forest model has huge issues with scalability. The strength of the model is that it is robust as well as highly accurate. Every tree in the forest is trained on a random sample of the overall data which prevents the predictor from overfitting. This model also has an effective method for handling missing data minimizing their impact on predictions. This issue with this model is as more features are included the model has to create even more splits to ensure that one feature is not being relied on too heavily. So as the datasets and features grow larger, runtime increases exponentially. This model performed significantly better than both the trivial and non-trivial baseline in RMSE and R^2.

4.4 K-Neighbors Regressor

The final model we created is the k- neighbors regressor model. This model classifies each point by the set of features it has, and groups it with other similar points. When a new point is introduced the model checks what features the point has, and finds the k nearest points and determines the value of the new point based on the average of the k nearest ones.

The intuition here is that cars with similar properties should be grouped together. Sedans would be grouped near sedans, new cars would be grouped near new cars, and so on. So the price of a specific car should be similar to other cars with similar properties.

This model was optimized through grid search and ablation study. The parameters being hypertuned in this model were the number of neighbors to consider and the weights of each neighbor. Through the grid search and ablation study the best k neighbors regressor model is the one that removes car model as a feature, checks the 5 nearest neighbors, and uses a uniform weight distribution.

This model, like random forest, struggles with scalability. As the features increase the model run time increases exponentially since the model has to check every point to every k nearest point. This model is also highly sensitive to outliers. K-neighbors regressor can also experience overfitting if the k value is set incorrectly. Since the optimal k value was found using grid search this issue should be of no concern in the specified model. The strength of this model, however, is that it is easy to implement, makes no assumptions on the data, can continually evolve with each new data point, and is simple and intuitive. This model performed significantly better in RMSE and R^2 than the trivial baseline, but only slightly better when compared to our best linear regression model.

4.5 Model Comparison

The trivial baseline model is clearly the easiest and fastest model to create, but it performed the worst as expected. K-neighbors regressor and random forest both provided excellent results, but had significantly longer run times than linear regression. Linear regression was able to perform close to k-neighbors regressor despite being a simpler model, because of its fast run time. Having a faster training time allowed the linear regression model to utilize a larger feature selection. Whereas the random forest and k-neighbors regressor model could only handle a one hot encoding that represented car models that appear over 100 times, the linear regression model could handle a one hot encoding of car models that appear over 4 times. When stating that linear regression could handle more features we do not mean that random forest and k-neighbors has a feature limit. We merely mean that our resources did not allow for us to run the two models with too many features. The difference in model run time and parameters is what led to different model optimizations, and as a result similar performances between the linear regression model and k-neighbors regressor model.

5 Literature

We use a dataset originating from Kaggle. It is scraped frequently from Craigslist by Austin Reese and is the result of a school project [1] . Information about the dataset and how it was scraped can be found at the Kaggle page and also includes a link to the GitHub repository to the scraping script. By looking through the GitHub repository, we can understand that the way the data is scraped is by visiting the Craigslist page for every city and then grabbing data from the cars+trucks sub-category page. This is useful so we can visit the data generating process itself if need be. For example, we could see whether a column was generated by the scraping tool itself or if it was a requirement when the user was making the Craigslist post. This dataset was used to provide Kaggle users a dataset for whatever purpose they personally deem fit, and does not have a specific intended application. For our purposes, we are interested in predicting the user submitted listing price of a vehicle given its respective features.

A similar dataset focusing on car features and MSRP is also hosted on Kaggle [2]. Rather than being Craiglist postings, this dataset was scraped from Edmunds. Edmunds is a used car resale company. Sadly, a link to the scraper is not available, so we would not be able to investigate how the data was exactly gathered. The motivating features here overlap greatly with our own dataset: model, year, cylinders, transmission. Since listings are posted by the company themselves, the data here is much more consistent and clean. Most notebooks made for this dataset study price prediction like ours and utilize models like XGBoost or random forest.

There is a significant amount of work already done in the area of predicting prices of used cars. What is considered state-of-the-art for this task would be the most successful methods tested by research labs. Pal et al. have written a paper on predicting price using a similar dataset [3]. This data also focuses on used car sales, but from the German subsidiary of eBay. Their work compared two models: linear regression and random forest regression. The results show that a random forest model proved more robust to overfitting than a linear regression model. In their hyper parameter tuning, they found success with n_estimators = 500 and max_features = num_features. These results would serve as a good place to start in our own model selection and tuning, as we cannot directly use their parameter values due to a difference in datasets. Additionally, Pal et al. found the most useful features to be kilometer(mileage), brand, and vehicle type. The input data was filtered to exclude outliers and irrelevant data, much like our own model.

Another example seems to belong to a group of students in a computer science course at Stanford [4]. Their raw dataset overlapped in features with ours: price, make, milage, year, and model. They tested a variety of models using R^2 as an accuracy metric: linear regression, gradient boost, random forest, LightGBM, XGBoost, k-means linear regression, and a deep neural network. The results show that among both the training and test accuracy, random forest proved the best. They chose to use a value of 36 for their max_depth parameter. As we needed to arbitrarily pick an initial value for our own random forest, we chose to use 36 as well.

Chen et al., using Shanghai car market data, analyzed the performance between linear regression and random forest regression models [5]. This analysis focused on the change in accuracy as a result of varying the number of features and the number of samples. The results show that as the number of features and samples increases, it begins to greatly out perform linear regression.

Based on the existing related work done on this predictive task, random forest surely seems the most effective. These conclusions agreed with our own results.

6 Conclusion/Results

To evaluate our models, we used two metrics, RMSE AND R^2 score. The result of our best model, the random forest regressor was an RMSE and R^2 score of 2034.96 and 0.97 on training data and 4642.66 and 0.83 on validation data. This was far and above the best result compared to our other models that we were comparing against as baselines. It definitely did better than the simple baseline where the mean was always predicted as that had an RMSE of 11291.09 and essentially no R^2 score as there is no correlation. The best linear regression model had an RMSE and R^2 score of 6475.60 and 0.67 on training data and 6756.03 and 0.64 on validation data. The best K nearest neighbors model was better than the linear regression model, but paled in comparison to the random forest regressor as it had an RMSE and R^2 score of 5420.80 and 0.78 on training data and 6064.81 and 0.66 on validation data.

By evaluating with validation scores, we were able to see that the random forest regressor easily had the best RMSE and R^2 scores, and was much better than our other models. On test data, the RMSE and R^2 score were 4881.25 and 0.82 respectively which outperformed linear regression and k-nearest neighbors RMSE and R^2 score of 6997.28 and 0.63 and 6654.57 and 0.66. This shows that the random forest regressor was able to minimize the errors between the predicted values and the real values of the car listings the best, and was able to create the best correlation between the prediction function and the real values. This means that our predictive task using this model is able to create very relevant predictions to the actual car listing prices and the error of those predictions are much smaller than with other models and baselines we are comparing against. We were able to get this result by removing the model feature from our list of features, but kept the rest in the represented training features. This means that the model is not necessary or not particularly helpful in determining car price in this tree based regression model.

Using grid search, we were able to find the best parameters as 16 for number of estimators and 36 as the max depth. This meant that we used 16 trees to be able to determine the pricing and a max depth of 36 for the trees, so we could see that a larger amount of trees with this max depth leads to a better regressor as it is able to account for the many features of our dataset we are using to train on. This model likely succeeded as it builds trees and it is able to adjust the trees in order to be able to handle many different situations and characteristics that occur within our dataset. The trees allow the model to be robust and handle certain problems in the prediction task with the full knowledge of many varying but competent trees. This model is able to adjust to the data much better than the simple baseline as it always predicted the same value and linear regression which is a linear boundary. It is also more flexible than k-nearest neighbors which may not be able to determine good values if there is not well definable clustering or spacing of the data. This is why the random forest regressor was the best model we had and worked well for this predictive task.

7 References

[1] Reese, Austin. “Used Cars Dataset.” *Kaggle*, 3 Dec. 2020, www.kaggle.com/austinreese/craigslist-carstrucks-data.

[2] CooperUnion. “Car Features and MSRP.” *Kaggle*, 21 Dec. 2016, www.kaggle.com/CooperUnion/cardataset.

[3] Pal, N., Arora, P., Kohli, P., Sundararaman, D., & Palakurthy, S. S. (2018, April). How Much Is My Car Worth? A Methodology for Predicting Used Cars’ Prices Using Random Forest. In Future of Information and Communication Conference (pp. 413-422). Springer, Cham.

[4] Kumbar, K., Gadre, P., Nayak, V. (2019, December). CS 229 Project Report: Predicting Used Car Prices. http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26612934.pdf.

[5] Chen, C.C., Hao, L.L., Xu, C.: Comparative analysis of used car price evaluation models. In: You, Z., Xiao, J., Tan, Z. (eds) Material Science, Energy Technology and Power Engineering I (AIP 2017), vol. 1839 (2017).