Predictive Modelling for Used Car Pricing

Charles Wang (A16790705)

March 20, 2024

1 Introduction

One Common question and consideration when trying to both buy and sell used cars involves the price. Both buyers and sellers want to be able to obtain a "fair" price for the car they are looking for. This price comes from a variety of factors, such as make, model, condition, and miles driven. This project covers the creation of cleaning a dataset and building a model to accurately predict the price of a used car. Here is a GitHub Repository containing all of the work: Jupyter Notebook

2 Dataset

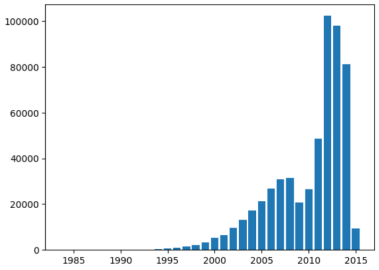
2.1 Data Source

The data here comes from a Kaggle Dataset, Vehicle Sales Data. The Dataset spans about 550,000 transactions of sales covering the year of manufacture, maker (eg. Honda), model (eg. Civic), body (eg. Sedan), trim (additional add ons such as AWD), transmission (automatic or manual), state of sale, condition of vehicle (on a scale of 1-50), odometer (miles driven), color of vehicle body, color of vehicle interior, Vehicle Identification Number (VIN), Seller, Manheim Market Report (Predicted Price from Wholesale Auctions from Manheim), Final Selling Price, and Final Sale Date. The next section covers the cleaning and ways of handling missing values, as well as the rationale for dropping a few columns.

2.2 Variable Selection and Data Cleaning

We start by dropping a few columns that are not relevant to our prediction at hand, starting with Sale Date. The goal is more so to predict how a car will sell now based on its current condition and characteristics, rather than look at sales in the past, so the time factor is ignored here. Along that same reasoning, the VIN column, and the Seller are dropped as well, since we do not want to look at individual cars or sellers to potentially influence sales price. The MMR Data is fairly interesting, and we will drop if for now since the goal is to use the vehicle's condition rather than basing a prediction off of another prediction, however will preserve it for comparison later.

Year



The Year Column breaks down the year of manufacture of each Car being sold. We can see that the data here is Left skewed, which makes sense in context. More people are interested in buying more recent cars, and as time goes on, fewer and fewer older cars are saleable rather than scrapped. Relevant to the imputation here, is that this column is complete, meaning every transaction had a year of manufacture even if they had nothing else.

Make

ford	99997
chevrolet	63269
nissan	54952
toyota	40151
dodge	30954
honda	27618
hyundai	21832
bmw	20793
kia	18296
chrysler	17483

Here's a subset of the Top 10 Vehicle Makers being sold. Notable here is the missing data, of which there were about 10,000 Cars without a maker, and corresponding Model, Body, and Trim. To fill in the missing data, the year Column was fully complete, so I took the most common car sold manufactured in the same year, and filled that in for the missing data. The next step was to also fix some naming issues, such as "Ford Truck" as the Maker rather than just Ford. Also relevant here is the model,trim, and body of which I imputed the data in largely the same way, only now I matched the Maker to sell their most common Car and under the logic that if it was more special in some way, then it would be mentioned as such. The data is not pictured here as each individual manufacturer has their own Car naming schemes, with their own models, so while they do influence price, the names themselves ultimately do not reveal too much about the data.

Transmission

transmission	
automatic	0.9638
manual	0.0362

Here is the breakdown between cars with Automatic Transmissions, and Manual Transmissions. Relevant to this column is the fact that a fairly significant amount of data was missing, 65000 entries, or almost 10 percent of the data. To not leave or try to exclude so much data, I imputed the missing values based on this article from Reader's Digest. In it, it discusses how 96 percent of Americans drive Automatic transmissions, so that is how I imputed the data. Notable here is that the numbers provided within the article seem accurate to the dataset, as the numbers are close across the whole data set even though I only imputed a fraction of it.

Color and Interior

color	
black	116295
white	111769
silver	87353
gray	86782
blue	53651
red	45629
green	11896
gold	11850
beige	9661
burgundy	9418

Figure 1: Body Color

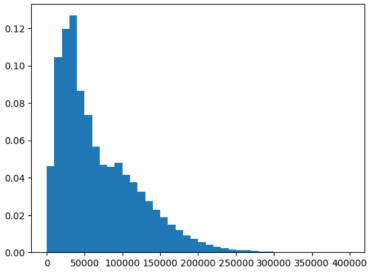
interior	
black	252307
gray	184527
beige	61766
tan	45478
brown	8924
red	1409
blue	1186
silver	1155
off-white	501
purple	347

Figure 2: Interior Color

These two get placed into the same category

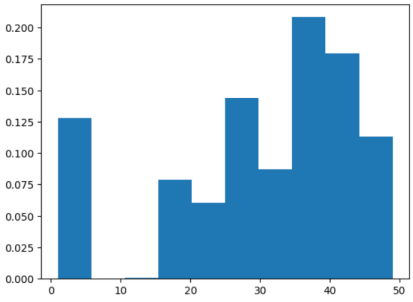
as the process for handling them largely are the same. It should be noted that these are the top 10 most common colors and interiors, as there are more uncommon options however they quickly reach only double digit counts. For colors and interior, the method of imputing missing values was similar to the Transmissions. Only this time, I used the data of the colors in the dataset itself to extrapolate odds for the frequency of each color appearing. From there, I randomly generated a color and interior based on those probabilities and used that to fill the null columns.

Odometer



This is a breakdown of the Odometer, or total number of Miles Driven. As we would expect, the data is right skewed, with fewer cars being sold as the miles driven increased. The data is categorized in 10,000 mile bins, and missing data was imputed similarly to the previous two columns: break down frequency by distribution, and then randomly imputed based on said probabilities. One thing to note though is that the data was eventually trimmed down, and outliers were placed to be at max 400000 miles, the extreme range of the data. Only something like 200 cars out of the full 550,000 were in that range, and even then, that may have been incorrect inputs as the most extreme end of the range had it at 999,999 miles. It largely did not seem to affect the performance of the final model.

Condition



Here Condition is represented on a scale from 1 to 50, and binned with intervals of size 5, and the y-axis represents the percentage presence rather than the raw count. Similarly to before, missing values were imputed from the existing data based on the probability presence. One very interesting trend is seeing that there is a fairly noticeable gap in 5-10, as it seems like most people would rather rate the car as "Very Broken/Inoperable" rather than a Very Poor quality rating. Additionally, we can see the trend what we would roughly expect, with most people rating their cars in Good condition, and from there, the count declines lower and lower until we reach that gap to very broken. It also seems like people are reluctant to label a car as nearly perfect to perfect condition.

3 Predictive Task

3.1 Task

From here, the goal after taking a look at and cleaning the data is seeing what the car will sell for based on its characteristics. This serves as a Regression Prediction Problem, as we are trying to predict final sale price.

3.2 Evaluation

Root Mean Squared Error (RMSE) serves as the metric by which the model will be evaluated. RMSE takes the prediction, subtracts it from the actual value, and then squares the value before finally taking the square root of the mean. This allows for the data to be calculated with the same units as the original, with the side effect of harshly punishing wrong predictions that are off by a significant margin. The rationale behind choosing the RMSE is that the final sales price of a car may give a very wrong metric, and encourage selling too high, or too low if the model's prediction is majorly offset. For that reason, the performance of the various models will be rated off their RMSE score on an unknown, testing dataset after being trained on a subsection of the training data. The split for this data is 60 percent training data, 20 percent validation data for HyperParameter tuning, and 20 percent testing data to evaluate the final performance. Read more here at this link

4 Models

The dataset that will be used to evaluate all models moving forward is the one with the features I have just described, as they have been cleaned and filled. The fibe models used for basic comparison are Linear Regression, Random Forest Regressor, XGB Regressor, LightGBM, and CatBoost. **Linear Regression:** The baseline model to compare more advanced options. It predicts the outcomes of the price by assigning "weights" to each variable with the ultimate goal of creating a straight line that fits the data as best as possible.

Random Forest Regressor: The Random Forest Regressor utilizes decision trees (ie breaking down the data into greater or less than certain values) in order to predict the final price of the car. The Random Forest Aspect involves drawing several of those decision trees, and averaging the results in order to create strong predictions on the data. Further Reading on Random Forest

XGB Regressor: XGB, or Extreme Gradient Boosting refers to the process of fitting and then utilizing gradient descent in order to fit better and better trees. It's another ensemble learning program, and works well to create predictions that train quickly. Further Reading on XGB Boost

CatBoost: CatBoost is another version of Gradient Boosting Decision Trees, with the benefit of also providing initial parameter tuning. It also has the benefit of allowing the use of Categorical Data, to avoid encoding and allows for the usage of GPUs in speeding up model training. CatBoost Website if you would like to use for yourself.

LightGBM: LightGBM is another Gradient Boosting Decision Tree based model and was created by Microsoft. It is based off a histogram approach to divide features into histogram bins. It notably seems to be optimized for large datasets, and progresses from each "leaf" that provides the best result rather than growing out an entire level to see each split. More Reading on LightGBM

Baseline Results: Linear Regression kicks us off with an RMSE of 3525, meaning that on average, our predictions were off by more than 3500 dollars. A position to start from, but other models do far more. Also notable is that numerical features such as Condition and Odometer had to be standardized to equalize the scale and their impact. The categorical data just needed to be OneHotEncoded, and is all the prepro-

cessing the other model types need as they are based off of tree splits rather than trying to fit a line to the data. LightGBM took the second longest to train despite its light moniker, and performed second worst with a RMSE of 3155, slightly better than Linear Regression. A long training time plus low training accuracy made me look towards other options. Random Forest was more promising with RMSE of 2248 but came with the notable downside of taking almost 10 minutes to train. This is with 32GB of RAM and an AMD R9 5900x, so although the performance is better, the longest training time out of all the models made me look elsewhere. CatBoost had good results at 2417 RMSE as a middle ground between Random Forest and LightGBM, however, XGB Regressor ended up being my final choice, Despite its RMSE of 2888, CatBoost notably already begins to HyperParameter tune while its descending, increasing the Run Time. The performance is close, and the ability to see the HyperParameters rather than them shifting was a notable advantage as the default settings for CatBoost vary based off the dataset.

4.1 HyperParameter Tunings

HyperParmeters were ultimately tuned utilizing sklearn's RandomizedSearchCV package. The time taken to iterate across 50 variations with 5 fold cross validation over the data was around 10 minutes utilizing every core on a fairly powerful setup, which was a major contributor in my decision to choose XGB Regressor over the other model choices that performed better without HyperParmeter tuning. The ultimate model chosen was

XGBRegressor(colsamplebytree=0.55, gamma=0.15, learningrate=0.2, maxdepth= 12, nestimators=500, regalpha=0.4, reglambda=0.1, subsample=0.75, njobs=-1))])

as although there was another round of Hyper Parameter tuning centered around the values we found in the first round of tuning, concerns rose for overfitting as the RMSE was 2037 dollars, but dropped only slightly to 2013 dollars after the second round of tuning.

5 Literature

The task of defining sales prices for used vehicles already exists in several forms to aid both private sellers and wholesaler car retailers. I'll address two such Kelly's Blue Book as a consumer option, and the Mannheim Market Report as the industry option.

Kelly's Blue Book

Kelly's Blue Book is a fairly popular ranking system for purchasing vehicles, especially as it breaks down the price people can expect for both trade in options, and for personal sales. They calculate their prices based off of real world sales data over the past, and makes predictions based off of the current economy and sales location. One thing to note though is that Kelly's has been noted to favor dealers and sellers, rather than purchasers of the vehicle as Kelly's suggests higher values than the tool that dealers most commonly use to assess vehicle values: the Mannheim Market Report. Further reading of issues with Kelly's Blue Book provided Here.

Mannheim Market Report

The Mannheim Market Report is one of the largest industry tools to assess wholesale trade in values for dealers. Mannheim itself is the world's largest wholesale car auction website, and their report uses over 10 million data points over 13 months to give the most accurate sales pricing

on cars. One very noteable drawback is that the MMR is not accessible to the public, and requires verification as both an authorized Car Auctioneer, your SSN, and a fee in order to view the evaluation. The tool very much so is a professional report rather than something the average consumer can access.
However, this dataset had the MMR evaluations of the vehicles as a column, so we can compare the prediction performances for both the MMR and the model created here to assess their performances. I dropped any comparison points where the MMR was null, as that may give an unfair advantage to my model as it does not have to treat those values as 0. Calculating the RMSE of the final model after Hyper Parameter tuning (trained off of 80 percent of data rather than the 60 percent previous) gives us a RMSE of 1935. In comparison, the Mannheim Market Report's RMSE was 1749, or a difference of around 200 dollars. Looking at the Mean Average Error also shows the immense impact some outlier predictions and sales had on the data, as the MAE for my final model was 1191 in comparison to the MMR of 1085 for a difference of about 100 dollars. Overall, while the MMR did perform better than my model, I had far less data to train on, and my work is available to the public to use, observe and hyper parameter tune for better performance while the MMR remains a private, black boxed, industry only tool.

6 Results

Utilizing XGB Regressor had a lower testing inaccuracy than basic Linear Regression and LightGBM, while also having a lower training time than CatBoost and a Random Forest Regressor despite the latter two's lower initial performance. Also noteworthy is that CatBoost tunes its own hyperparameters slightly when being utilized, leading to be choosing to optimize an XGB Regressor for the final results. The choice to eventually HyperParameter tune on an XGB Regressor gave the final RMSE of 1935. This is worse than the MMR, however that has both more data to work with, and is a black boxed, industry only paid assessment of car valuations.

6.1 Feature Ablation

One fairly notable feature to analyze is the condition column. While the other features, such as vehicle make, model, year, color, odometer, and state of sale are all easily answerable, the vehicle condition is the most subjective column out there. As the Kelly's Blue Book article notes, one of the largest contributors to mismatched evaluations is an incorrect quality assessment, as what the buyer, the seller, and a professional assessment rate as "good quality" can range wildly. Here, I set the entire condition column to 1, so it provided no further information on final sale price, and the RMSE was 2086. While this does mark a fairly notable decrease in prediction accuracy, it also shows that you can still use the model even with incorrect condition assessments as the average sale does not exclude the condition entirely.

6.2 Hyper Parameters

The Hyper Parameters chosen were trained on the entire dataset with a RandomSearchCV, eventually landing on the HyperParameters noted before. The overall runtime across roughly around 70 iterations with 5 fold cross validation calculated about 350 models across a range of hy-

per parameters before landing on the parameters linked earlier. This represents a fairly notable reduction of the RMSE by almost 33 percent. Any further Hyper Parameter tuning on the dataset may find it useful to use the settings previously outlined as a baseline to improve performance, but should take note in the potentially long run times, and utilize njobs=-1 to utilize every core of your CPU. I would however warn of long run times, and potentially looking into utilizing GPU based hyper parameter tuning, as my set took nearly 15 minutes with a very high end CPU