Derek Wright

CptS 315 Project Report

04/29/2022

I. Introduction

For this project I sought out mining information from used car sales data. With this data I wanted to find a machine learning approach that could accurately use the data to predict car sale prices, then turn around and use that for a sale price recommender. The main questions I wanted to answer were: what aspects of a sold car affected its sales price? And will non-numeric data ,such as car make and model, be easy, or even possible, to use? Both of these questions, along with determining the best machine learning method, ended up being the main challenges of my project. Some of the non-numeric data was pretty easy to convert to numeric values but most of them proved to be much more difficult. I was motivated to choose this data by my interest in cars. I'm a self-taught mechanic and am very passionate about working on and understanding cars. The used car market is a very interesting place, and it is often difficult to estimate a used car's value. There are already tools out there that accomplish my goal, such as Kelly Blue Book, and I was interested in understanding how they work on a deeper level and found that this was a great opportunity to dive in and figure it out.

The overall results of this project were not what I was hoping for. I didn't manage to find a suitable way to format the data along with a good algorithm to accurately predict used car sale prices. None of the algorithms I implemented provided accurate enough predictions, and in order to accomplish this I would need to make a few changes.

II. Data Mining Task

The first step of this data mining task was to decide what tools I wanted to use. I chose to use a collections of Python plugins: Pandas to represent the data as a Data Frame, Scikit Learn for the data mining algorithms, and NumPy for added mathematic support. The general idea is to fit the data into a Pandas Data Frame, use the Data Frame to then make feature vectors for each sale listing, and then plug the feature vectors into different Scikit Learn machine learning algorithms. The output I focused on from these algorithms was from their Score() functions, which takes the test feature vector and expected results and calculates the accuracy of the trained model. I also looked at the root mean square error (RMSE), R2 value, and mean absolute error (AME) values to get a more in depth look at the results of the training.

The main goal of my data mining task was to predict car sale values and apply those findings to make a recommender for prices based on the key features that influence the value. I had a few questions that I did want to answer along the way as well. First, I wanted to find a good way to fill in the missing values of data like the one I was using. Similarly, I also wanted to learn how I could use non-numeric data values, like a car's make and model, in a machine learning algorithm with this goal. Would it be easy? Is it even possible?

There were a few challenges that I met throughout this process. It was more difficult to fill the missing values than I initially expected. Even once the data was cut down to what I thought was relevant, and all the missing values were filled in, I found it also challenging to find suitable machine learning algorithms

for the data. This challenge was primarily due to the number of non-numeric features I wanted to use (like state of sale). The final challenge was then to make sure that the data was in a good format for the specific machine learning algorithms, as many of them will not behave properly without data following certain distributions.

III. Technical Approach

With the tools I was going to use decided, I was then able to collect the data into a Pandas Data Frame and get to work with it. The data had plenty of unnecessary columns for my needs, so before anything else I cut it down into only the columns I wanted. Many of the rows contained missing information, so I had to decide whether to drop them, or try to fill the empty spots. I opted to fill the empty slots as I felt that it would be both a better learning experience and, should I do it well, will provide more accurate results. This was also one of the questions I wanted to answer and what led to my first challenge.

In order to fill the empty cells, I needed to determine what approach I wanted to take. I considered using a nearest neighbor type approach but decided that with this data it went beyond the scope I was aiming for. Upon searching for options, I came across a similar project (source 5) with a fill approach that I liked, so I used a very similar one. For numerical values like year and odometer, I filled them in with the mean of all the rows that shared the same of another value. So, to fill a missing year of a Honda Accord, it was filled with the mean year of all Honda Accords in the data. The non-numeric data was done in the same way, but I used mode instead of mean, since the mean could not be calculated. I didn't get the chance to try another method, or to compare the results with only data that didn't need to be filled, but I fell like this method is suitable for car data since many cars that share certain attributes will often share another similar other attribute.

The next step was the then start trying out different machine learning algorithm. The challenge here was to find algorithms that worked on the type of data I had. The linear algorithms I tried did not provide satisfying results and could only really be used with the numeric data. In order to use some of the more complicated algorithms like a multilayered perceptron classifier, it was necessary to transform the data into one of many distributions. I chose to use the Scikit QuantileTransformer which transformed the data into a uniform distribution. I was then able to use this data to get better results from the machine learning algorithms.

I did not make it to the final step of making a recommender for used car sale prices because I did not find an algorithm that provided accurate enough predictions. This leads me to further question how Kelly Blue Book and other similar tools do it, and the algorithms they use. They don't seem to publicly disclose this information, which would have been a helpful guide for me in the project.

IV. Evaluation Methodology

The dataset I used was the collection of all of the past successful sales of used cars on Craigslist from the last year. The columns consisted of the various possible attributes each sales posting contained, and each row was a single posting, with all of the values that it had contained in the posting when it sold. Many of the columns were not relevant enough for predicting price, such as the VIN and url, so I dropped these columns before I used the data. Among the data I did consider to be relevant to determining price were some that consumed too much memory to use in the algorithms when turned

into dummy values. I found this method in the source 4 that I used, where you could turn non-numeric values into dummy values. This would take one column, such as state, and create a column of each possible value of state. The values of these columns would then be binary indicators. The columns that didn't play nicely with this method were the ones that had many different possible values, like model of the car. I opted to use the numeric data first to analyze the results with just them, while I tried to find the best way to use the non-numeric data without using up too much memory. Using only numeric data of course severely limits the accuracy of the predictor since more than half of the relevant values were cut out. Comparing the results of only numeric with the results of all data serves as a good way to show this. However, using the data with only the memory intensive dummy values removed did not seem to improve the accuracy substantially.

The main way I evaluated the results of each different algorithm was with the Scikit built in Score() function. It uses the test data and corresponding expected prices to determine the accuracy of the trained algorithms. As not to limit myself to only one stat for evaluating, I also looked at the RMSE, R2, and MAE values of each outcome using the trained algorithm to predict the prices of the test data set.

V. Results and Discussion

Scikit-learn classifier	Feature Vector	RMSE	R2 Value	MAE	Accuracy
Linear Discrimination Analysis	Num_features	4865091.74	- 4.546e-05	30824.82	0.020816
Linear Discrimination Analysis	All_features	25212248.54	-1.958	1064838.51	0.023060
Linear Discrimination Analysis	Num_features: QuantileTransformed	nan	1.320	78353.12	0.0284
Linear Discrimination Analysis	All_features QuantileTransformed	nan	1.214	91671.80	0.0290
Linear Regression	Num_features: QuantileTransformed – sample 50000	20579714826.28	-1946437092576	11879503241	-31.31 BAD
Linear Regression	All_features QuantileTransformed - sample 50000	Results too Extreme (> 10e17)			
Random Forrest Regressor	Num_features: QuantileTransformed	531897.768	-0.000690	21472.217	-7.909
Random Forrest Regressor	All_features QuantileTransformed	4840846.724	2.375e-06	34092.29	-0.122

The above results show that linear algorithms truly are not suitable for this type of data, at least without first transforming it to a uniform distribution. The linear algorithms with data that wasn't transformed had only 2% accuracy with numeric only data, and only 2.3% with the data including the dummy values that didn't exceed the memory limit. The Linear Regression results were shockingly bad and threw out

values that were far too extreme to even consider it viable. The random forest regressor seemed to do well at first, but after a few different runs it became clear that the accuracy is very irregular and cannot be relied upon with this data set. I tried the random forest classifier as well, but it required too much memory for even smaller sample sizes.

Converting the non-numeric values into dummy did not work well since there were far too many possible values for most of the columns. Another method that suits this data type should be used. Its hard to say whether or not my fill method worked well, but considering the algorithms were able to provide any prediction at all, I think its safe to assume that it is at least a viable option. The only way to determine if it is a good option though would be to test other methods and compare them.

VI. Lessons Learned

- What did you learn by doing this project? In the hindsight, would you have made some different decisions to improve the project further?

With the knowledge that I have gained in doing this project, I would certainly start off with other approaches. I think that using a nearest neighbor approach for filling the missing values would likely provide more accurate results, so I would try to find ones that would work for this data set and applying it. I could then compare the results with the method I used this time to determine how well this approach actually worked. Before I went into this, I hadn't really considered transforming the data to a uniform distribution, so I would definitely focus on that more the next time around. The data played a lot nicer with the algorithms after transforming it. The last change I would make is the way I deal with the non-numeric data. Using dummy values was the only method I managed to find in my search, but I am certain that there are much better ways to do it with a data set like the one I used. I think the biggest hang up in my results was from not being able to properly make use of all of the important data. This would likely make the biggest improvement and would therefore be my primary focus in a future attempt.

VII. Acknowledgements

1. Data set: https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data
2. Pandas Documentation: https://pandas.pydata.org/docs/reference/frame.html
3. Scikit-Learn Documentation: 1. Supervised learning — scikit-learn 1.0.2 documentation
4. https://www.bluegranite.com/blog/predicting-sales-with-the-aid-of-pandas
5. https://www.kaggle.com/code/jerrymazeyu/predict-car-price-by-catboost/notebook