Maxime Bourassa

260868968

March 9th, 2020

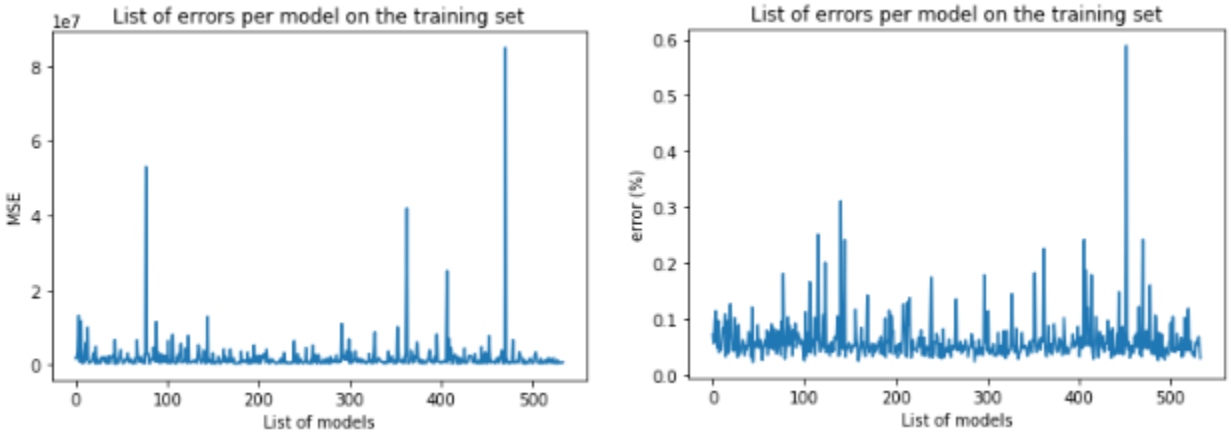
MAIS 202 – Deliverable 3

1. Final training results

Compared to my preliminary results, my final results are much better, but there still seems to be some overfitting in the model. Compared to the previous submission, first I tested on the whole dataset after downloading it directly from Kaggle. Also, I removed the make as a variable since it was redundant to the model. Then, I first applied a random forest regression model from Sickit Learn as before, but I was still getting pretty bad results with a MSE of 23590371. This is very large, but to put it into perspective as I will do with the rest of my results, on an average car price of $21085 in the data, this represents an error of 23% on the price on the validation set. This was still way to high. Therefore, I decided to review my data preprocessing. I re-arranged entries per model and then processed entries on a per model basis. Then, I only retained models which had at least 500 entries in the data, which cut the number of listings I was evaluating from 1.2million to 1million and the number of models being evaluated from 1300 to 500. Since 500 models over 1 million entries still seemed reasonable, I decided to reduce the number of models I was evaluating to get greater accuracy. This allowed me to bring my median (and average) MSE and % error significantly down both for training and validation. My results are summarized in the following graph. Each phase, training, validation and testing has two graphs, one for MSE and one for % error on the price as described before. Since I am dealing with car prices of multiple thousands of dollars, I think the % error is a more easily readable metric for measuring the accuracy of the model.

Note that for this part of the report and subsequent parts, I used a 60:20:20 division for the training, validation and testing set respectively.

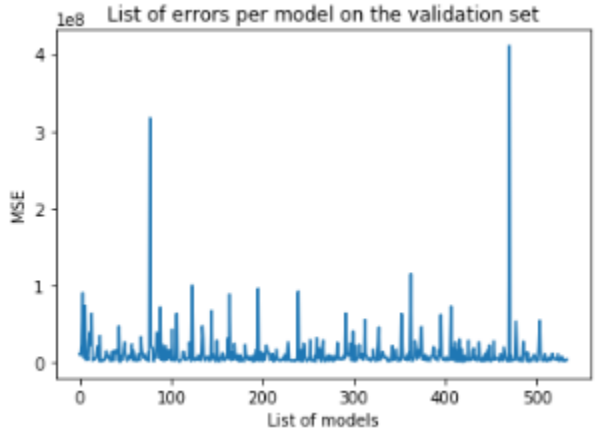
Training set:

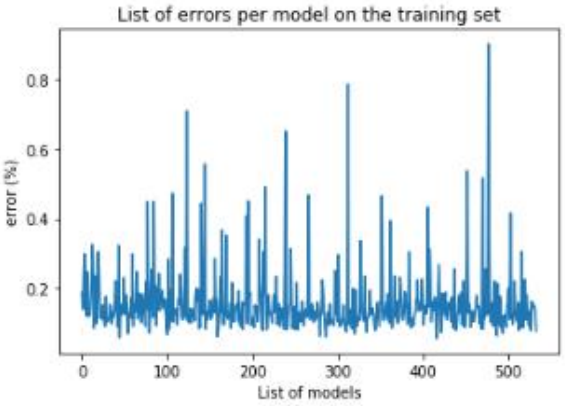


Average % error: 6%

Median % error: 5%

Validation set:

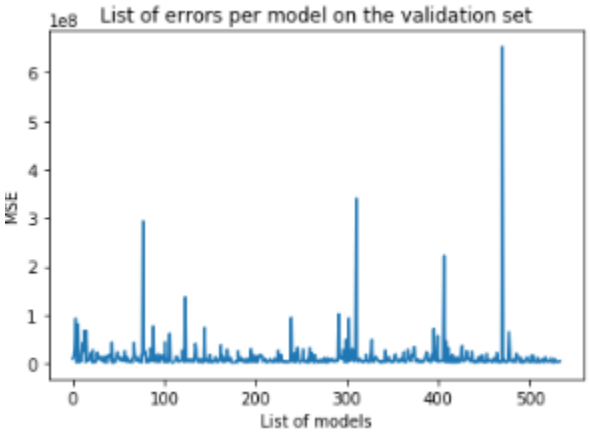


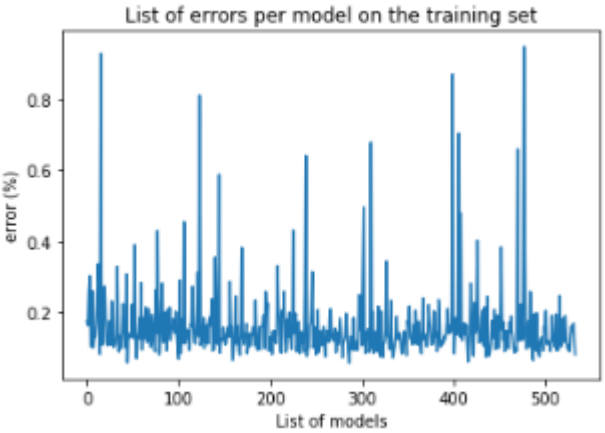


Average % error: 15%

Median % error: 13%

Test set:





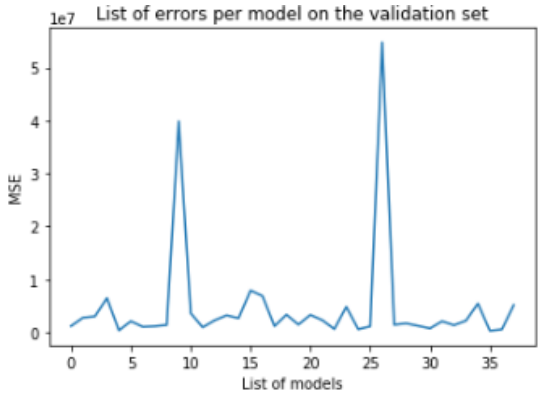
Average % error: 15%

Median % error: 13%

Note that to make the graph readable, I left the x-axis as numbers even though in reality, every x value represents a different model. Also note that the average and median % error are almost equal for the validation and testing set (they differ in their first decimal). But these are about 3 times as big as the training set % error. Unfortunately, changing the hyperparameters of the random forest regressions did not help this issue as I got worst results both in training and validation when I did not use the default parameters of the model. As seen from the graphs, some entries are consistently bad through the training validation and testing phase. These car models should be removed for the final product. Another problem though is that even though the mean and median % error are very similar in the validation and testing set; they are not the same on a per model basis. This is also a problem that could be solved by addressing my overfitting issue.

Therefore, I decided that these results were not good enough and I therefore tried yet another preprocessing technique. I decided to go with a more logical and less extreme division of my data. Instead of taking the data as a whole or dividing it into each model I decided to do as we do when we think about cars. I separated cars by make. Intuitively this makes sense since we usually associate each car make with a certain expected price range. Therefore, I redid all my preprocessing, separated the listings per make, associated numbers to each model with a given make's model consisting of consecutive numbers. I only considered makes that had at least 500 listings. Then, I use a random forest regression model for each make and got the following MSE and % error results form the training, validation and testing set.

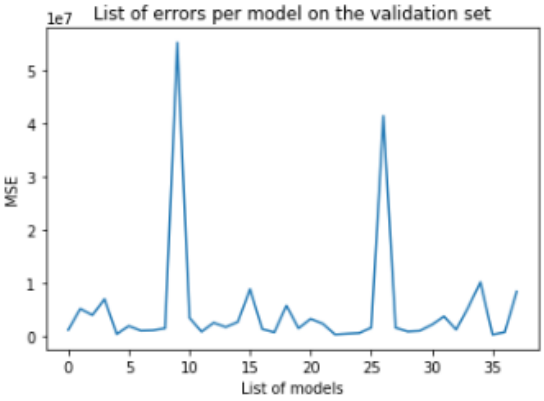
Training set:



Average % error: 8.3%

Median % error: 6.9%

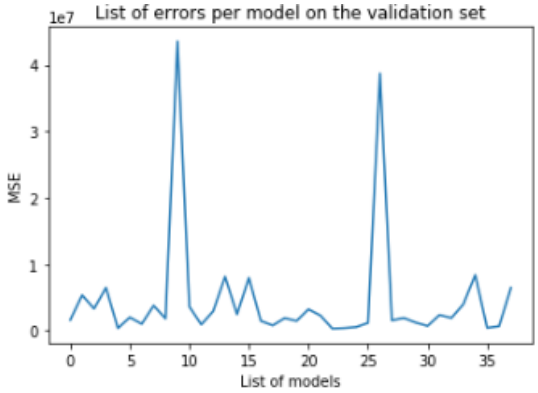
Validation set:

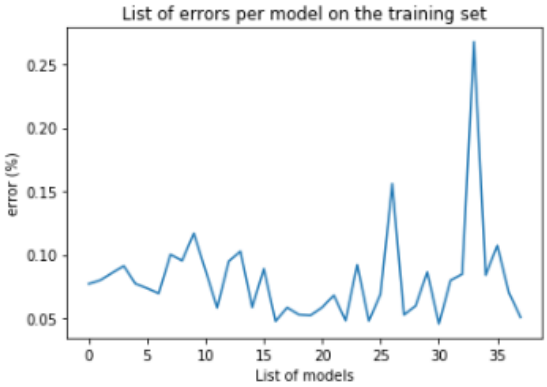


Average % error: 8.2%

Median % error: 7.1%

Test set:





Average % error: 8.1%

Median % error: 7.7%

Note that each value on the x-axis represents a different make. (which is way less dense then in the previous part since there is only 38 makes with more than 500 entries in the data)

These results are much better and are finally acceptable for presentation. In fact, the error is surprisingly almost constant between the training, validation and testing phases. And, with average and median % error under 10% these are good results. Note that if I have time, we can see clear outliers from the data that I could remove since the model is not able to find a good regression for those makes.

2. Final Demonstration Proposal

I will use a landing page type website to demo my final project. On the page, the users will be able to select a make in a drop down menu, then models from that make in another drop-down menu. Finally, the user can enter a year and a mileage and click a button to predict the price.

The page should then display three things:

- The predicted price (output of the mode)
- The closest entry in the dataset from the requested point (model, year, mileage)
- The % error from the point in the dataset

I do not have any experience with web developments. Therefore, I will learn on my own using resources such as code academy and I will then post my landing page on a free platform. And, most importantly, and will ask questions to execs or other people in the bootcamps that have more experience in web development if I have any problems I can not solve. If you have any advice on technologies I should or should not use to create my webpage, please advise me!