Linear Statistical Models
Project report

Martin Groman

9.1.2018

Contents

1 Introduction

In this report I have picked to analyze an automobile data set from year 1985, which contains 26 different attributes with 205 instances from unspecified area (data available at [1]) . My main goal is to compare different model selection algorithms when building a reliable prediction model. I will also comment the ability to interpret certain models, but my main interest lies in prediction.
In the second chapter I describe my dataset and transformations I have done in order to be able to run multiple regression and build predictive model. I also briefly comment collinearity and its potential issues in my data.
In the third chapter I study various model selection methods. Each subsection is dedicated to specific model selection algorithm or analysis. Also, I provide brief discussion over the chosen method, stating its advantages and disadvantages.
To briefly preview my results, I would just say, I was most satisfied with performance of cross-validation. To add to that, I was pleasantly surprised with regression trees and with specific package *party*, which provided easy to read visual results.
However my main point of interest are prediction capabilities, which I summarize in the conclusion.

2 The data set

In this section I am going to describe my data set and more importantly used transformations with my reasoning, why I did certain adjustments.
As stated in the introduction my data set contains 26 different attributes with 205 observations. I will use attribute *price* as my dependent variable and the others as independent variables.
Before I get to any data transformations, I have to check the data set for missing values. Unfortunately, independent variable *normalized-losses* has around one fourth of observations missing, so I won't be using this variable. Additionally, I have also removed 4 observations with missing value for my dependent variable and a few missing observations from independent variables. I believe I can afford to remove these observations, as their number isn't crucial compared to how many observations I have left, so I wouldn't be worried about this problem.

2.1 Data transformations

First of all I have divided my independent variables into numerical and catogorical group, where the former group contains 16 variables and the latter 8.
Let's focus on numerical ones, because we will get back to categorical ones later. In order to fulfill 5 basic assumptions I had to perform several data transformations, including log and inverse transforms. The only transformation worth mentioning is probably of my dependent variable *log(price)*, the others will be observed during model selection, but apart from log and inverse tranformations, nothing else was used.
Furthermore, I ran into a problem, or let's say case, where the model sufficiency assumption was broken. When plotting *log(price)* vs. *compression ratio* I spotted 2 distinct clusters of data, far away from each other. The explanation is simple- the difference between these two groups is the fuel type as can be seen in **Figure 1**, i.e. depends if the car runs on gas or diesel.

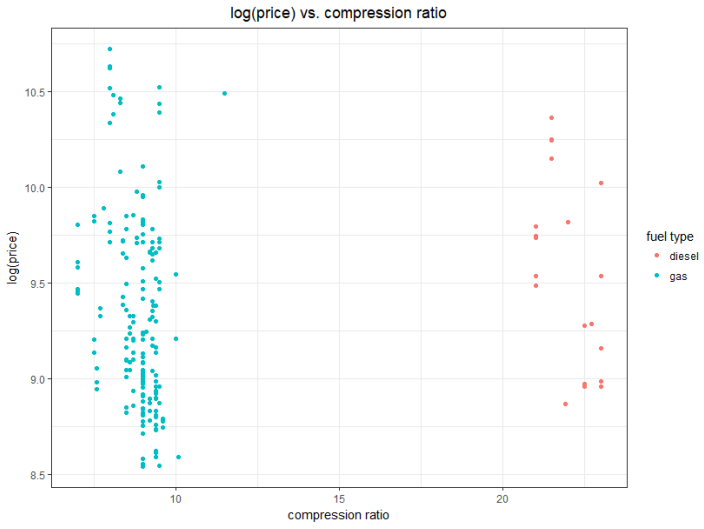

Figure 1: log(price) vs. compression ratio

Now let's take a look at the categorical variables. To be more specific there is one variable that will probably need some adjustments, and that is *make*. It has more than 20 levels and considering the fact I don't so many observations for each level, I will split this categorical variable differently in fewer levels. I have decided to group this variable into these levels - japanese, american, german and other european car brands. This should help me later in model selection phase.

2.2 Collinearity

Before I get to model selection, I would like to take a moment and see if I am dealing with any correlated independent variables.
From the **Figure 2** below, I am able to spot several highly correlated variables. These are for instance related to the dimensions of a car - length, width, weight, etc. Another example is correlation between city mileage per gallon and highway mileage per gallon, which have huge correlation of 97%, which is also not very surprising.
It is related to, what is often called the collinearity problem, because the regression coefficients will get highly unstable and sensitive to little random errors in dependent variable. Additionally, as we add or remove more independent variables from or to the model, we can observe wild fluctuations of these coefficients. This kind of instability then shows up as quite large standard error for partial regression coefficients. [3]
I will deal with the collinearity problem later, when doing model selection with different methods. Nevertheless, there aren't many solutions to this problem, I can either remove some of the correlated variables or, as I will show you later, apply principal component regression.

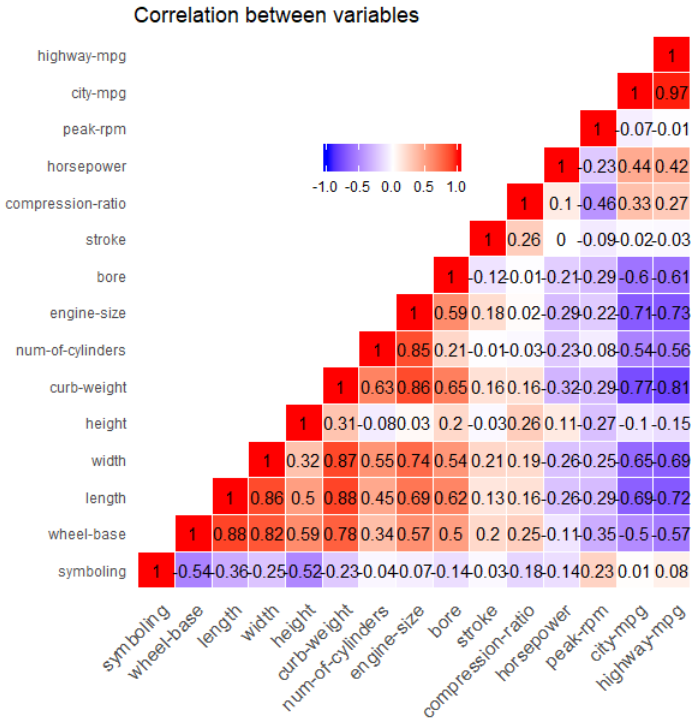

Figure 2: Collinearity heatmap (code used [2])

3 Model selection

In this chapter I will apply various model selection methods on my data. I will first provide brief description of chosen algorithm, then compute the result of each method and in the end discuss its performance and ability for future predictions.

3.1 Backward model selection

Description

It is one of the less complex algorithms, because it can be used only for nested models. Therefore, the whole algorithm starts by fitting a model with all of available variables. The next step is to drop the least significant variable, under the condition that it's not at the chosen critical level. After that, we re-fit our model and again drop the least significat variable. We repeat these steps, until all of the variables, which are left, are statistically significant. [4] [5]

Application

So, first of all I try to fit all the variables into the model. I would also like to note, I have split the data into training and test set 100 times and first, I will just show, how it's applied and then in the end say something about using this algorithm for 100 different training and test sets. The ratio for training and test set is always 80/20.
Before moving forward, I would like to briefly discuss summary of the initial fit. I have obtained suspiciously high value of R^2=0.942. Obviously, this fact doesn't mean that this model is the best, even though if we checked diagnostics plots, we might not notice anything bad, surprisingly they look very good. However, I have included so many variables, that I have basically forced the model to describe random error in the data. In other words, I have built an overfitted model, which is far too complex and is not suitable for prediction or even describing the relationships between variables themselves.[6]
Let's run the backward selection

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-1.1399	1.1413	-1.00	0.3196
symboling	-0.0230	0.0140	-1.65	0.1021
wheel_base	0.0109	0.0045	2.40	0.0175
height	-0.0192	0.0078	-2.47	0.0148
log(curb_weight)	1.2788	0.1649	7.76	0.0000
horsepower	0.0041	0.0007	5.71	0.0000
highway_mpg	0.0049	0.0035	1.39	0.1663
as.factor(make)german	0.3387	0.0553	6.13	0.0000
as.factor(make)japanese	0.0418	0.0408	1.02	0.3073
as.factor(make)other	0.1346	0.0572	2.35	0.0201
as.factor(body_style)hardtop	-0.4716	0.1235	-3.82	0.0002
as.factor(body_style)hatchback	-0.2562	0.0750	-3.41	0.0008
as.factor(body_style)sedan	-0.2251	0.0764	-2.94	0.0038
as.factor(body_style)wagon	-0.2896	0.0847	-3.42	0.0008
as.factor(engine_location)rear	0.6638	0.1551	4.28	0.0000

Table 1: Backward selection

To my surprise, this model still has very high R^2=0.92 and diagnostics plots still look pretty solid. So I might have rushed my decision about overfitting in previous paragraph. Why is R^2 so high? Well, I tried to fit only 2 first predictors, horsepower with wheelbase and using these two only I already received R^2=0.8 so then adding more variables raised it to the mentioned value.
However, R^2 is not the only deciding factor when modelling, I also checked the diagnostics. The potential problem could be seen in QQ plot (**Figure 3**), where I spot kind of long tail in the lower part. It suggests that residuals are not exactly normally distributed and possibly not ideally transformed, but I wasn't able to figure out better transformations to fix this problem.
Furthermore, when I check the residuals plot as well as QQ plot, I can spot an outlier, with observation number 58. For further prediction modelling I would remove this outlier. Otherwise, I think the rest is acceptable and wouldn't interfere anymore.

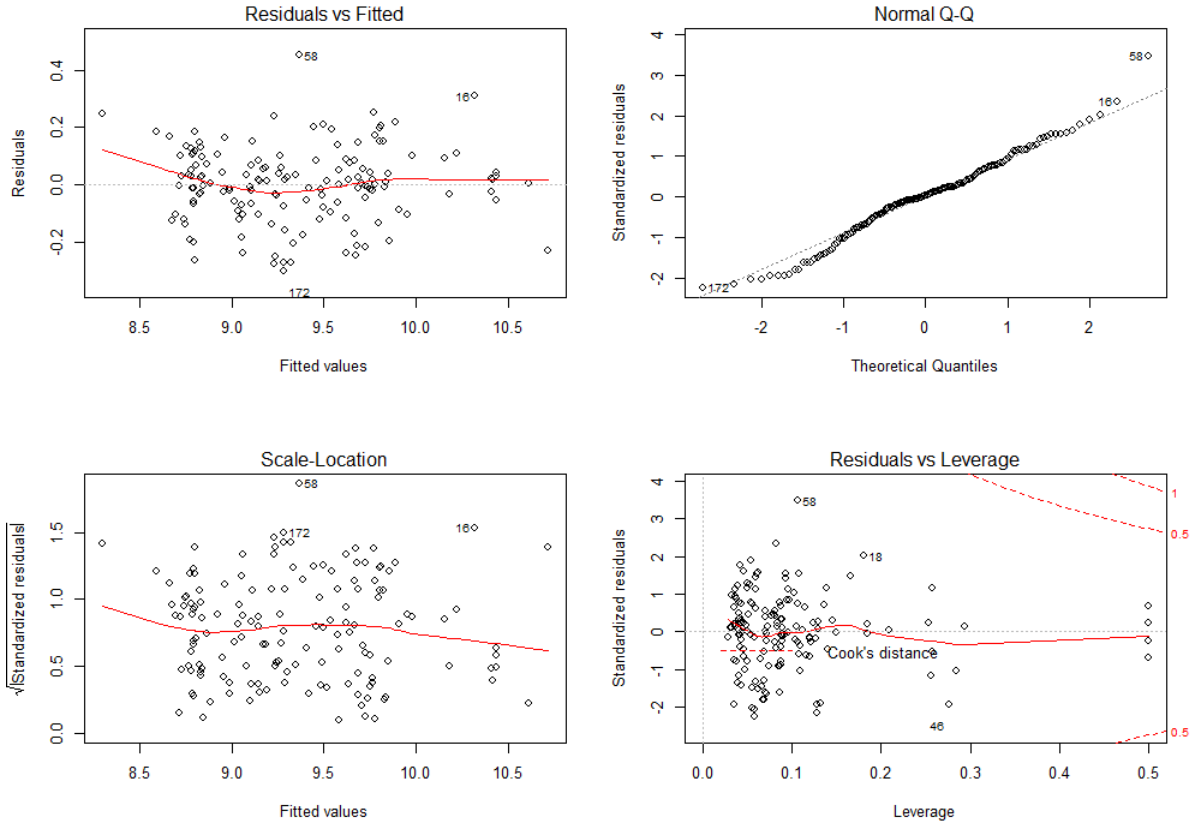

Figure 3: Diagnostics - backward selection

Let's see how good prediction this model can perform. I will calculate prediction error using this formula

$$PE = \frac{1}{n}\sum_{i=1}^{n}(y_{test} - y_{predicted})^2$$

to receive value

$$PE = 0.0434.$$

Now let's see how backward selection performs over 100 different splits in terms of prediction error (**Table 2**).

	median	minimum	maximum
Backward selection	0.0413	0.0233	0.06741

Table 2: Backward selection over 100 different splits - prediction errors

Discussion

In the beginning of this section I showed how to apply backward selection to my data. When I fitted the whole model I realized I have very high R^2 and I thought it was caused by overfitting, but when I looked deeper into the data I found out, there are some good predictors, so then adding few more variables just raised R^2 to such high level.
I also checked diagnostics to be sure I am not violating any of basic assumptions. I found a slight issue with my QQ plot, but no other transformations I tried on my data made this look better.
At the end of this section I ran backward selection 100 times to obtain some results for future comparison. I can't say, how well it performed now, but I don't expect this algorithm to be the most accurate.
To conclude backward model selection, I would like to point out that this algorithm is very old and practically not used these days, as there are far superior algorithms. I wanted to show just a basic example of model selection, which should provide sort of a stepping stone for other algorithms, so I have something to compare them with. I expect the following algorithms to have better predictive performance.

3.2 K-fold cross-validation

Description

The problem with splitting data into 2 distinct groups is that there are many ways how to split it. Moreover, one group is exclusively used for training and the other exclusively for testing and both of them are chosen by chance.
K-fold CV provides slightly different approach for model selection. The idea is to split all of the data into K equal sized subsets. We go through all enumerated models, while each of K groups is once used as a test group and used as a training group in the remaining cases. Then we compute the total prediction error sum of squares for each model and subsequently pick the model, which minimizes said error. [5]

Application

I will use 10-fold cross validation for my data. So, as said in the description, I first create 10 folds of data, then cycle through each one of them and fit all of the models. Finally I compute prediction errors for every model.
There are over 4400 models enumerated and due to high number of variables I won't paste here a table with overview, but instead just list variables that have always appeared in top 5 models. Namely- symboling, width, curb-weight, horsepower and dummy variables- dmakeother, dbodystyleconvertible, dbodystylewagon, ddrivewheelsrwd, denginelocationrear. It's worth noting, the baseline I chose for dummy variables is - dmakejapanese, dbodystylesedan, ddrivewheelsfwd, denginelocationfront.
However, I can check the top 5 models and their prediction errors across different folds, as can be seen in **Table 3**.

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	0.370	0.267	0.393	0.466	0.617	0.434	0.312	0.395	0.236	0.295	0.01962
2	0.380	0.294	0.397	0.457	0.608	0.444	0.308	0.386	0.225	0.292	0.01964
3	0.369	0.266	0.392	0.471	0.621	0.434	0.316	0.399	0.234	0.293	0.01965
4	0.377	0.293	0.395	0.461	0.612	0.443	0.311	0.391	0.222	0.289	0.01966
5	0.372	0.270	0.399	0.464	0.617	0.433	0.311	0.394	0.248	0.294	0.01970

Table 3: Prederrors in different folds and total

It is clear to see, the top 5 models have very similar prediction errors and therefore it doesn't matter that much which one we pick. Anyway the best one contains these variables- symboling, width, curb-weight, horsepower, dmakegerman, dmakeother, dbodystyleconvertible, dbodystylewagon, ddrivewheelsrwd, denginelocationrear.
Now, given the fact CV chose only certain levels of my factor variables I have to decide whether to keep only some levels in the model or remove the whole variable altogether. From variable *make* I am missing level american. So what I have decided to do is, group this catergorical variable into fewer levels, namely: american, european, japanese.

Problem is with other 2 categorical variables, which both are missing just one level, *bodystyle* is missing level hardtop and *drivewheel* is missing 4wd. I don't think I can reduce number of levels anymore in these variables, because all of them are really important. So I will try to run the CV once more with adjusted *make* variable and see how it changes. Results can be seen in **Table 4**.

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	0.472	0.343	0.368	0.662	0.594	0.377	0.368	0.436	0.319	0.359	0.02226
2	0.401	0.314	0.392	0.810	0.574	0.392	0.315	0.504	0.263	0.339	0.02230
3	0.405	0.329	0.393	0.803	0.581	0.377	0.307	0.510	0.257	0.350	0.02234
4	0.471	0.347	0.369	0.678	0.579	0.364	0.390	0.422	0.316	0.380	0.02236
5	0.451	0.320	0.395	0.800	0.561	0.348	0.349	0.474	0.275	0.346	0.02237

Table 4: Prederrors in different folds and total

Now from this result of CV I chose the model number 3, because this one already contains all levels of involved categorical variables. So variables are not exactly the same as in the last case, but quite similar, more specifically - symboling, wheel-base, width, curb-weight, horsepower, city-mpg and dummies - dmakeamerican, dmakejapanese (now with baseline being dmakeeuropean), dbodystyleconvertible, dbodystylehardtop, dbodystyle-hatchback, dbodystyle-wagon, denginelocationrear. However, by changing the categorical variable I have slightly increased prediction error and it seems to me I have a lot of variables in the model, so maybe there might be another way.
What I tried next, was to plot CVerror vs model size (**Figure 4**). If I consider models with 7-8 and more variables, the prediction error remains more or less the same. So it is useless trying to fit high number of variables, because we are increasing the complexity of model, but we do not get anything better in result in terms of prediction. Both of my previous attempts at CV had quite high number of variables, so I have decided to limit models to be enumerated to have at most 8 variables.

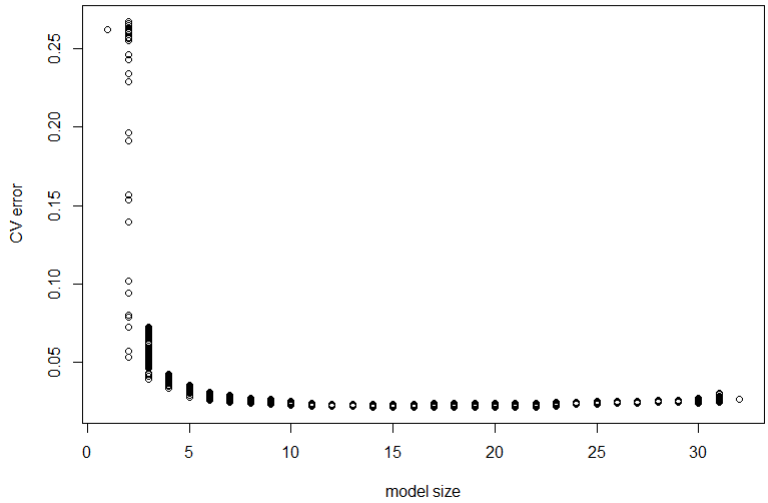

Figure 4: CV error vs model size

This time I have to deal with over 900 models that have 8 or less variables (including intercept) and the results can be seen in **Table 5**.

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	0.584	1.001	0.617	0.297	0.415	0.532	0.251	0.334	0.323	0.315	0.02419
2	0.657	0.797	0.500	0.399	0.496	0.514	0.310	0.314	0.348	0.335	0.02419
3	0.678	0.975	0.553	0.376	0.486	0.477	0.273	0.242	0.304	0.307	0.02420
4	0.648	0.857	0.514	0.436	0.525	0.488	0.261	0.315	0.269	0.377	0.02430
5	0.686	0.788	0.494	0.443	0.504	0.527	0.325	0.274	0.312	0.345	0.02434

Table 5: Prederrors in different folds and total

Let's take a look at the best model in this case. There are these variables - curb-weight, width, engine-size, horsepower, dmakeamerican, dmakejapanese (with baseline dmakeeuropean), denginelocationrear (with baseline denginelocationfront). Obviously, the prediction errors have risen once again, but I am feeling pretty confident about this model, as it is quite intuitive, it contains variables one would think are important when talking about car price.
Last thing I will do is to run CV 100 times for these 8 variables and see how well it performs (**Table 6**). Note: I have always picked the best model, so the table below describes values for the best model.

	median	minimum	maximum
CV	0.0235	0.0225	0.0244

Table 6: CV over 100 different runs - prediction errors

Discussion

It is clear to see that cross-validation is very powerful model selection tool. First of all, I tried to use it on my data without any additional thought, that's when I ran into a small problem. My models performed well in terms of prediction errors, but also created issues with ability to interpret my results.
I tried to group my categorical variables into fewer levels, but I couldn't find reasonable smaller groups for all variables. Furthermore, I observed slight increase in prediction error, but nothing too worrying.
However, I was still not satisfied with the result and I found it difficult to say what's going on in the model, because of the dummy variables, so that's why I went in an unusual direction. I restricted the number of variables that can appear in final model and in the end, I think I obtained quite interpretable model with pretty solid prediction error. Also thanks to running CV 100 times I found out, prediction errors are quite consistent and stable.
Question is, if it was worth it. I would argue yes. I am aware I have increased prediction error, but from my point of view the pros of my final model overweigh the pros of my initial model. Additionally, the increase in prediction error is not that drastic and I am quite sure, this algorithm will be one of the best in terms of prediction performance in my project.

3.3 Principal components regression

Description

The main advantage of using PCR can be seen when analyzing data suffering from multicollinearity. First, we standardize our data and then perform principal component analysis. Now, we are able to express our data in the new coordinate system determined by principal component directions. This way, the newly obtained variables will be uncorrelated. This algorithm is not perfect, though, as it has a small flaw. It doesn't take into consideration the dependent variable, when deciding which principal components should be dropped.
In the application part I will also study what influence has substituting ordinary PCA with sparse PCA. The main difference lies in fact the sparse PCA allows only some variables to have contribution to each principal component. [5],[7]

Application

First of all, I standardise my data to zero mean and unit variance and then ran the principal components regression. After having full model fit, I used stepwise selection to arrive to this model (**Table 7**).

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	9.6332	0.0814	118.27	0.0000
pc1	-0.1110	0.0225	-4.94	0.0000
pc3	-0.0662	0.0222	-2.98	0.0034
pc6	-0.1098	0.0240	-4.58	0.0000
pc9	0.0942	0.0212	4.44	0.0000
pc12	0.1105	0.0179	6.16	0.0000
pc13	-0.0986	0.0265	-3.72	0.0003
as.factor(body_style)hardtop	-0.6024	0.1362	-4.42	0.0000
as.factor(body_style)hatchback	-0.3861	0.0811	-4.76	0.0000
as.factor(body_style)sedan	-0.2493	0.0872	-2.86	0.0049
as.factor(body_style)wagon	-0.3050	0.0986	-3.09	0.0024
as.factor(engine_location)rear	1.0266	0.1685	6.09	0.0000

Table 7: Principal components regression

Once again, I receive high R^2=0.91, which doesn't surprise me anymore it was the same case with backward selection. What is also similar is the problematic QQ plot (**Figure 5**) and this time the long tail is even more obvious. Apart from this, diagnostics look pretty good. I tried looking into specific observations, which are causing this long tail, but I haven't been able to tell their common aspect or something, that could help me understand this behaviour, and furthermore I think the group is too large to be removed as outliers.

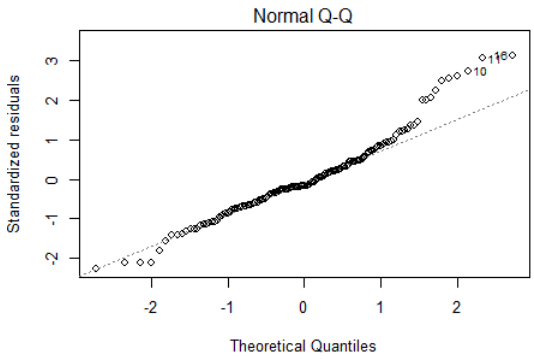

Figure 5: QQ plot - principal components regression

One more thing I would like to point out when looking at the model summary. Even though PCR chose similar categorical variables, it chose quite different numerical variables than in both previous cases. Well, we have actually lost possibility to interpret the PC variables as they almost always involve all the original x variables. I can't include the table with contributions of each variable to specific principal component due to limited space, however all my original variables are involved in every PC. Nevertheless, this is very typical for PCR, it is price we pay for dealing with collinearity, because as mentioned in description, the new variables are uncorrelated. [5]
My main point of interest is obviously prediction performance so I computed prediction error for this algorithm and got value

$$PE = 0.0484.$$

Now let's substitute ordinary PCA with sparse PCA and see how it influences my results. I will adjust my code, so that my new variables are composed of fewer original x variables. On average roughly 3 original variables contribute to new ones now. Eventually, I arrive to following model (**Table 8**)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	9.7090	0.0782	124.11	0.0000
pc4	0.1611	0.0262	6.16	0.0000
pc6	-0.1370	0.0334	-4.10	0.0001
pc8	-0.0697	0.0338	-2.06	0.0409
pc10	-0.1285	0.0282	-4.56	0.0000
as.factor(body_style)hardtop	-0.6023	0.1455	-4.14	0.0001
as.factor(body_style)hatchback	-0.4007	0.0829	-4.83	0.0000
as.factor(body_style)sedan	-0.3517	0.0818	-4.30	0.0000
as.factor(body_style)wagon	-0.4433	0.0913	-4.86	0.0000
as.factor(engine_location)rear	0.8545	0.1820	4.69	0.0000

Table 8: Principal components regression with sparse PCA

What I immediately notice is the new PC variables changed. This time I can actually check, which original x variables are involved in these variables (**Table 9**). Note: for space saving purposes I picked only variables which contributed at least once.

	PC4	PC6	PC8	PC10
symboling	0.00	0.00	0.57	0.00
num_of_doors	0.00	0.00	0.82	0.00
wheel_base	0.00	0.00	0.00	0.38
width	0.00	0.00	0.00	0.18
num_of_cylinders	0.00	0.00	0.00	0.14
stroke	0.98	-0.18	0.00	0.00
compression_ratio	0.00	0.11	0.00	0.00
peak_rpm	0.18	0.98	0.00	0.00
city_mpg	0.00	0.00	0.00	-0.11
highway_mpg	0.00	0.00	0.00	-0.89

Table 9: Contributions to specific principal components

Let's focus for a moment on **Table 9**. I can see variables that haven't appeared in previous model selection methods. What is even more surprising, the best predictor - horsepower - has not been involved in any of the new variables. On the other hand absolutely terrible predictor for car price - number of doors - bears quite big contribution to *PC8* variable.
If I take a look how well the model did in terms of prediction, I compute prediction error to get

$$PE = 0.0515.$$

Now let's run both types of PCR 100 times and compare their prediction errors.

	median	minimum	maximum
ordinary PCA	0.0312	0.0168	0.0671
sparse PCA	0.0290	0.0113	0.0664

Table 10: PCR over 100 different runs - prediction errors

Discussion

PCR is worth using when you are dealing with correlated data. There are 2 possible ways how to approach it. First, if I use ordinary principal component analysis, as I did in the beginning of this subsection, then I don't have to worry about collinearity, however I lose ability to interpret the model, because every new PC variable involve all of the original x variables.
One of the solutions how to fix this, is to use sparse principal components analysis, as shown on previous and this page. Now I am more aware what's happening in the model, plus as I have found out it performs better in prediction as can be seen in **Table 10**. The only downside I can say, was probably computation time, which was longer than in case of ordinary PCA.
The question then boils down to what are you interested in. If I don't care at all what the model says, I don't have to trouble myself with sparse PCA. However, that's usually not the case, I want to know what am doing and what's happening, so that's why I would pick rather sparse PCA over ordinary PCA. Unfortunately, then you face another dilemma, because you have to decide how to adjust the sparsity. If you allow too many original variables to be involved in the new ones, then you are losing the possibility to interpret the model once again. In my opinion, it's reasonable to let at most 3-4 variables to be of contribution, because otherwise it will only get complicated to say for sure what's going on in there.

3.4 Regression Trees

Description

Also called prediction trees, are used for representing the recursive partition. Regression trees end in terminal nodes, which we call leaves. They represent a certain cell of the partition. The whole algorithm starts at so called root node, and then by asking questions about specific features of the data we split it into several cells. Usually we ask yes/no question to keep the trees simple, but it is also possible to ask more complicated questions. The biggest advantage of using regression trees is they are quickly computed and provide clear interpretation. On the other hand, they often lack good prediction performance [8]

Application

For application on my data I used R package *party*. In the **Figure 6** below I show my generated tree.

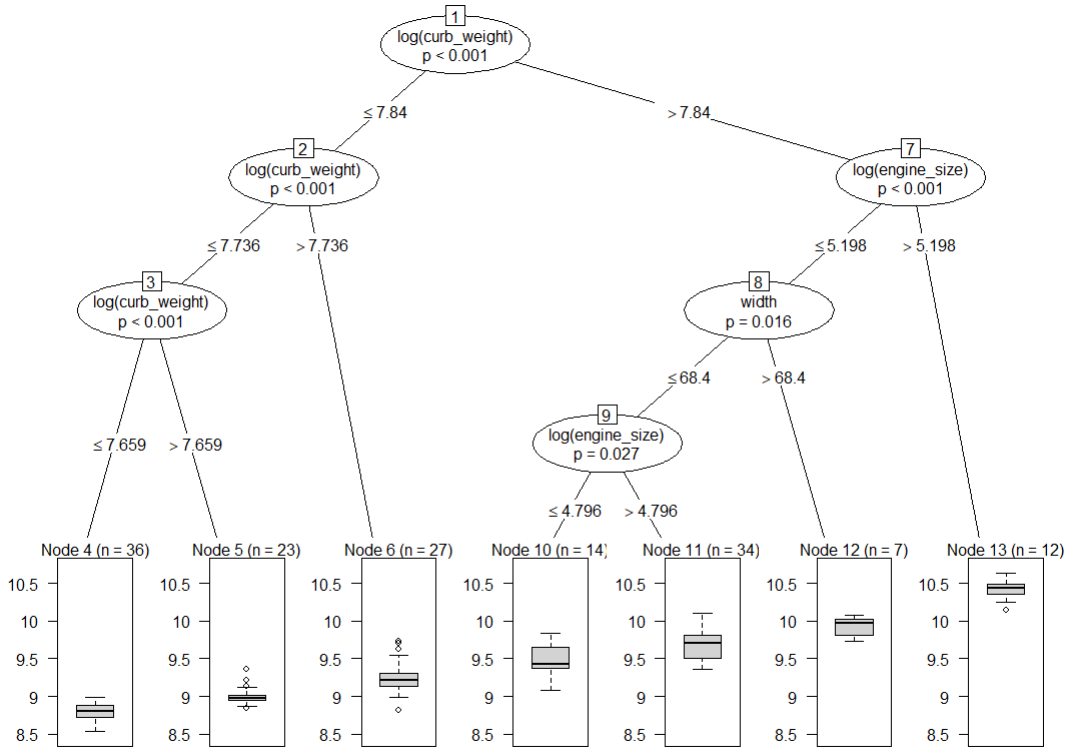

Figure 6: Regression tree (party package)

I would like to comment briefly on this figure. Just at the first glance, the most appealing property of regression trees can be seen. It is really easy to interpret and clean to read. The reason I chose this package is because it provides box plots on the leaves.
Let's try to go through one branch until the very end. So, we start by asking about the curb weight and if it's higher than 7.87 we ask about log(engine-size) and compare with the value 5.198. Now, if it's higher than we have arrived to the first leaf (node 13), it is a group of 12 observations with median log(price) slightly below 10.5. If it's lower than 5.198 we ask about the width of a car and compare to value 68.4. This way we can interpret the whole tree.

Nevertheless, this tree is not perfect. I have actually run into very common problem with regression trees. If I take a look at the whole left branch it is the same question three times in a row. It is sign of overfitting, kind of like making a small step-line instead of a straight line.
However, I cannot prune this tree anymore in this package, because I set p values to be smaller than 0.05 and therefore I already got pruned tree. I will compare this result with older packages to see what differencies they hold.
I quickly compute prediction error for this package and get

$$PE = 0.0319.$$

Now let's try to do it the usual way with *tree* package. Initially I obtain a tree of size 10. Then I plotted CV deviance vs tree size to prune the tree to one of size 7. This tree is depicted in **Figure 7**.

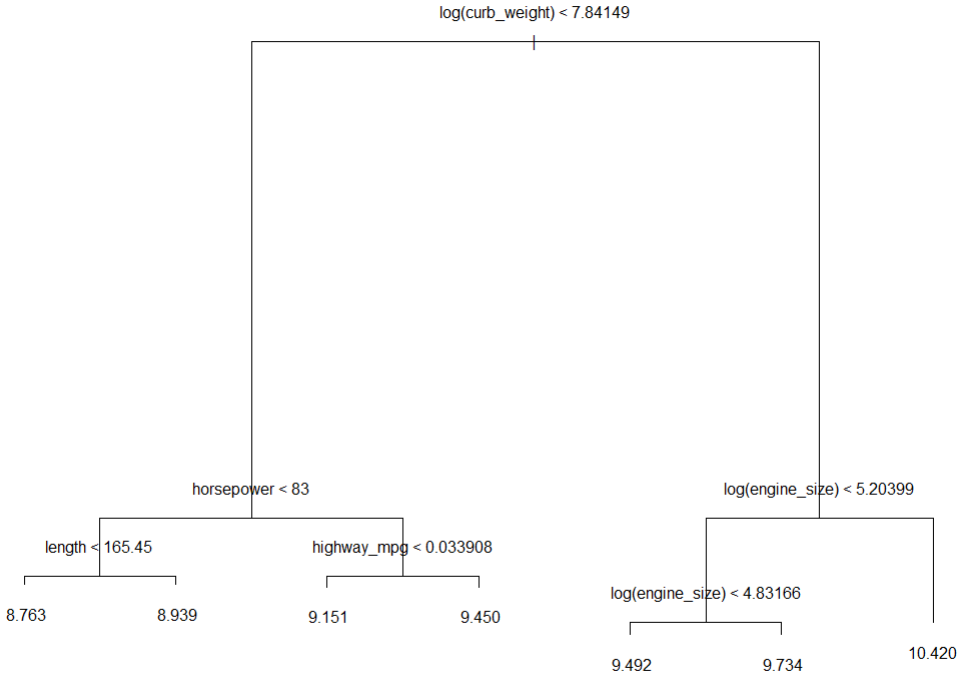

Figure 7: Regression tree (tree package)

Now for this package is typical, the longer the branch is, the bigger and also more important split in the data I have done. So as in previous tree, I can see the most important question is about the curb weight, even the value is the same. But other variables included are mostly different. And how did this specific tree perform in terms of prediction? Let's compute prediction error

$$PE = 0.0368.$$

Well, for this split into training and test set, I can see the *party* package performed better, but let's run the algorithm in both packages 100 times to see, which one is truly better for predicting and also more stable when predicting.
I created a **Table 11** for comparison of prediction errors through 100 iterations. Note: I used pruned tree for prediction in tree package and used alpha level of 0.95 for party package.

	median	minimum	maximum
party package	0.0778	0.0365	0.1933
tree package	0.0789	0.0366	0.2111

Table 11: Comparison of R packages

So from the above table I see I got pretty lucky with my data split, because I had a very low prediction error. Unfortunately, the range of possible prediction errors is quite large, so I can get really well performing models as well as very bad ones. If I should compare the mentioned packages, I don't think there is a clear winner. Medians of prediction errors are almost the same, there is a little difference in standard deviation, but I don't think it's significant.

Discussion

Regression trees are popular, because of their visual properties, everybody can see what's going on in the model and in my opinion it's one of the easiest methods to understand and to learn.
Thanks to various packages for R, you can make your prediction trees say a lot. That's what I tried to show in the start of this subsection, where I built a tree using *party* package. It chooses variables according to their significance, so the model it spits out is already pruned tree.
As we haven't used this package in lectures I was curious, how it stands in comparison with *tree* package. That's why I ran both of these options 100 times to see which one offers better prediction possibilities. In the end, as **Table 11** shows, it came out as a close tie. Both of them are quite unstable, but that's what I expected in the first place. Decision is then up to us and here I would go for the *party* package, because its visuals seemed more powerful than its opponent's.

4 Conclusion

In my project I have compared 4 different approaches to model selection with prediction as my main aspect. However, in this summarization I also take in consideration better ability to interpret some models. I have collected all prediction errors for various algorithms in **Table 12** below.

	median	minimum	maximum
Backward selection	0.0413	0.0233	0.0674
Cross-validation	0.0235	0.0225	0.0244
PCR with ordinary PCA	0.0312	0.0168	0.0671
PCR with sparse PCA	0.0290	0.0113	0.0664
Regression tree (party package)	0.0778	0.0365	0.1933
Regression tree (tree package)	0.0789	0.0366	0.2111

Table 12: Comparison of prediction errors over 100 runs of algorithm

What is the best overall model selection algorithm? That really depends on where are our expectations and interests.
Do I care a lot about being able to interpret the model? If not I can use principal components regression with ordinary PCA. It is ideal solution for data with correlated variables, but the price we pay is ability to interpret the results.
I have shown if one decides to use PCR, maybe better choice is PCR with sparse PCA. It still is not perfect in terms of being able to tell what's happening in the model, but it does better job in that than PCR with ordinary PCA.
When talking about ability to interpret the model, the best choice is definitely regression tree. I would say it is ideal tool for presentations for less educated audience. It is quick and easy to use and read. On the other hand, its stability and consistency of prediction is quite poor.
I have also studied backward selection. Its performance was not the worst, but apart from its easy use I fail to find any other positive properties of this algorithm. That's also, why it is not used anymore and was substituted with other algorithms.
Lastly, I tried model selection via cross-validation. I am most satisfied with results of this algorithm, even though I ran into specific problems with categorical variables, which might slow down the whole process. Nevertheless, it has proven to be the most consistent and stable algorithm when compared to any of these and it is not hard to say what's going on in the model either.

References

[1] Jeffrey C. Schlimmer, *Automobile data set* [online]. [cit. 2017-11-18]. Available from: http://archive.ics.uci.edu/ml/datasets/Automobile

[2] http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-data-visualization

[3] J.O. Rawlings, S.G. Pantula, D.A. Dickey. Applied Regression Analysis.

[4] https://www.stat.ubc.ca/ rollin/teach/643w04/lec/node42.html

[5] Lecture notes. Available from: http://www.math.chalmers.se/Stat/Grundutb/GU/MSG500/A17

[6] http://statisticsbyjim.com/regression/r-squared-too-high/

[7] https://blogs.sas.com/content/iml/2017/10/25/principal-component-regression-drawbacks.html

[8] http://www.stat.cmu.edu/ cshalizi/350-2006/lecture-10.pdf

Minis overview

Mini 1

In the first mini we have learnt how to deal with data transformations and how to fullfil 5 basic assumptions.
Let's focus on data transformations. Why do we transform data? Mostly because we want to fullfil mentioned assumptions, but also to interpret data better for example.
Probably the most used transformation is *log* either natural or *log10*. It is ideal for data, which is spread very wide across one of the axes and log shrinks it more together. We usually interpret this transformation as changing something multiplicative into additive, because that's the property of logs.
Then, there is inverse transformation, often used for e.g. mileage. I guess it's obvious that this transformation inverses the data points. Still pretty easy to interpret in most cases, if it's variable something per something.
Lastly, still quite common transformation is square root. It comes handy if we see some kind of squared pattern in the scatter plot. However, we might have troubles interpreting these transformations sometimes.
To conclude data transformations, I would like to emphasize, that it pays off to take time to do these transformations as it immensely helps later on with diagnostics. Also, don't use more transformations on certain variables as you lose ability to interpret, what it means.
Now let me briefly comment on 5 basic assumptions. Apart from uncorrelated errors, you can spot all of them in diagnostic and scatter plots.
Model sufficiency and constant error variance you can spot in scatter plot, that's like the first sign if you did data transformations correctly. However, even with perfect transformations, this assumptions can still remain violated, then you have to pick more complex model.
I didn't run into any problems with symmetric errors and uncorrelated errors assumptions, nevertheless, potentially problematic is always no outliers assumption. What I always did, was to take a closer look at the outlying observation and try to tell if it's justifiable to drop it or not. You have to always keep the size of your data sample in mind, because that's often the deciding factor whether to keep the outliers or not.

Mini 2

In this mini I attempted to build a primitive predictive model. I had 500 observations, split into training and test set in 50/50 ratio.
First of all I tried to find out possible data transformations and sure enough I was able to apply them to a few variables. I mostly used log transformations for variables regarding certain area, for example - sqft living, sqft lot, etc. I have also log transformed dependent variable, which in this case was house price. I would like to point out one thing, you have to be really sure it pays off to tranform dependent variable and first check scatter plots of it vs all other variables to see if fits better.

Then we tried to fit the whole model, obviously the diagnostic plots didn't look very good and even though I got high R^2, I couldn't be satisfied with this model as it didn't fullfil 5 basic assumptions.
That's why I decided to drop the least significant variables according to F-test for nested models. I kept track of these models to compare their prediction errors in the end.
Eventually I have removed around 5 variables and obtained my final model. This model had some outliers, though. I have decided to drop them as they were very few compared to the number of observations I had in total.
Then, I have tried to compute prediction errors and compare them across all models. What I have found out, is the final model and the one with size just plus one had very similar prediction errors. The others didn't perform so well in terms of prediction. In my opinion, when I have two similarly predicting models, I would go for the simpler one, because there is no reason, why to complicate things with more complex model.
In this mini I have also taken a look at collinearity between variables and I have found few highly correlated ones. There wasn't much to do how to solve it, just be careful, that not many of them appear in the final model, which could cause biased results.

Mini 3

In the mini I have studied the influence of data transformations on model selection.
First of all I tried to fit all variables untransformed to the model to see what happens and then use stepwise regression to arrive to final model. When I checked the diagnostic plots of either the full model or the final model, I could spot violations of 5 basic assumptions. Often issues was not fullfiling the constant error variance and no outliers assumptions. It was clearly caused by the lack of transformations.
So, then I tried to apply all possible transformations in order to get better results. Sure enough, I could see the difference even in the full model, but more significant change could be seen in the final model. All basic assumptions were fullfiled, maybe just few outliers, which needed to be dealt with. However, not everything can be solved with data transformations and outliers can always appear.
When I took a closer look at the two final models and ignored the diagnostics, I noticed there were small differencies in variables picked. What I think happened was that in the untransformed case, stepwise regression didn't have so many options, which predictor to choose, because most of them looked quite bad when plotted vs dependent variable. However, when I transformed the data I made this decision easier, because there were clear strong predictors.
I will just briefly skim through the transformations as I already described them in mini 1. The number one transformation is log, used for limiting the spread across one of the axes. I found it useful when dealing with area or price variables. Also commonly used is inverse transformation, ideal for variables describing variables like fuel consumption or some property per capita in population. Last one of the popular transformation is square root, which is used if we spot some kind of parabolic pattern in our data.

Mini 4

In this mini I have compared several model selection methods to see which one offers the best prediction performance.
First of all I applied all the data transformations as in mini 2. This time, though, I had to work also with categorical variables. Looking at some scatter plots I decided to bin levels of some of them into fewer groups. Also, I have transformed numerical variable *sqft-basement* into categorical variable, with levels - no basement, smaller one and bigger one. The reason behind this was, there were so many houses without basement and I felt it will work better this way.
First of all I did the stepwise regression, which later proved to be not so accurate as other methods. However it was quick and easy to use, so I used it as a comparing example with other algorithms.
Then I tried cross-validation and model selection criteria AIC, BIC, Cp. Apart from BIC, all these algorithms chose very similar models. BIC chose slightly smaller model, which also didn't perform as well with prediction errors, when compared to other criterias. Eventually it turned out, the best prediction errors were received from models by AIC, Cp and CV.
Furthermore, I tried to perform principal components regression on my data. It didn't do that bad, however, prediction error of its models was larger than one from stepwise regression, so for this data, it wasn't optimal choice.
Lastly, I build a regression tree. I didn't have any expectations of better prediction performance than the rest. Obviously it turned out to be the worst model selection tool, or to be more exact the least consistent. And even the best model of regression trees wasn't quite as good as the one chosen by selection criteria.