Predicting used cars prices with supervised methods

Picascia Sergio
Data Science and Economics
University of Milan

Abstract

Buying a new car is usually considered an important event, a renovation for the every day life. But where does the old one go? Sometimes it is given to another member of the family, sometimes we decide to trade it, in order to gain some money. We then go to a dealership or we try to sell it as a private and, in this last case, we look up on the internet at which price other similar cars are being sold. But how is this cost chosen? What are the main characteristics that are taken in consideration and how they affect it? The aim of this study is to answer this questions and, eventually, build a model which is capable of predicting the final value by itself. The data analysed has been retrieved from Craigslist and resulted in a huge dataset containing information about hundreds of thousands of cars sold in the USA. The analysis involves the use of two supervised learning methods: linear regression and decision trees. If the first one is more likely to give an interpretation of the model, understanding in which way the variables contribute to the final result, the second method usually performs better in terms of prediction precision. In particular, we will reach a test accuracy of over 98% with an ensemble learning method called random forest, while with multiple linear regression we will be able to understand which attributes have a greater impact on the response variable.

Introduction

The automotive market is one of the most important nowadays: even with the 2020 crisis due to COVID-19, 60 million cars are expected to be sold during this year. The global used car market is growing even more rapidly, primarily as a consequence of the entrance of players in countries in development, but also thanks to the increase in incomes and the decrease in car ownership periods. It is clear, then, that understanding the mechanisms of used car exchange can result in a competitive edge. Because of that, my analysis will focus on recognise which features affect the most the price of a used car and build an algorithm that will be able to predict it.

The data analysed is retrieved from a Kaggle repository which, in turn,

gathers it from Craigslist, an American advertisements website that, among other sections, has one of the widest collection of used cars for sales. The dataset came with 25 variables and more than 400 000 observations, but a cleaning was needed due to massive amount of NAs and some useless columns. For example, among the others, I removed the 'id', two columns containing URLs, 'latitude' and 'longitude' since I already had a variable 'state' and the attribute 'description', that might be included in a further text analysis. I proceeded eliminating some outliers, such as prices over $40 000 or year later than 2020, converting many variables to categorical and dropping the remaining NAs. I ended up analysing a data set with more than 100 000 cars and the following 13 variables:

- price
- year
- manufacturer
- condition
- cylinders
- fuel
- odometer
- title_status
- transmission
- drive
- type
- paint_color
- state

With the exception of 'price', 'year' and 'odometer' that are quantitative variables, all the other attributes are qualitative, some with more than 40 classes ('states' and 'manufacturer'). This brought the need to create a dummy for each distinct class.

Analysis

Before starting with modelling, I looked at the plots of 'price' against every variable: they still showed a presence of some outliers, even after my initial cleaning, and it is already possible to observe some interesting pattern, like the positive correlation between 'price' and 'year', or the negative one that came up in the plot against 'odometer' (Figure 1).

I then divided the dataset in training and test: unfortunately i didn't have a separate dataset to evaluate my model, so I performed the split on the original data. I ran a simple linear regression, including all the independent variables: the result, even in the simplicity of the model, is quite complex to interpret, due to the amount of coefficients estimated, nearly 140; this was caused by the huge number of categorical features present in the dataset that are represented

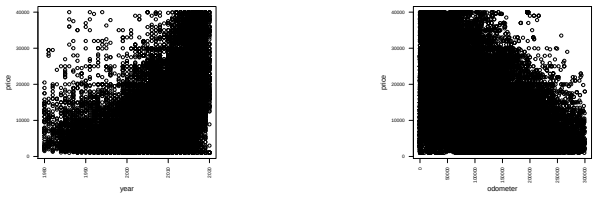
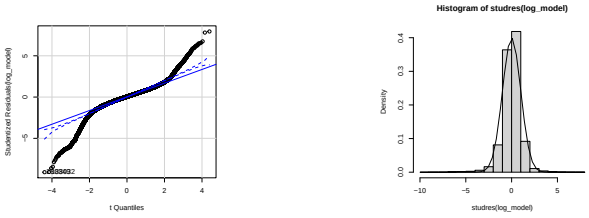


Figure 1: Price against year (left) and odometer (right).

as dummy variables in the model. However, the majority of coefficient were significant even at a 0.1% level, the adjusted R-squared is pretty high (71.8%), considering the number of variables included, the p-value for the F test is very small and the residual standard error equals 4412. I computed the VIF adjusted for categorical variables and their roots are all close to 1, showing the absence of multicollinearity. Evaluating the model on the test set, showed a high test error and an accuracy of just 69.8%.

Scaling the variables can improve the performance of a linear model, especially when there are attributes with completely different scales, like 'year' and 'odometer' in my case; but this change did not bring any advantage in my analysis, so I decided not to consider this model. Another possible approach was the logarithmic transformation of the variables: I ran the three types of model (log-log, level-log, log-level), but only one provided a significant improvement, the log-level. Applying the logarithm on the response variable 'price', resulted in a model with a slightly better adjusted R-squared (73.9%), an obvious reduction in the residual standard error (0.3794), mostly due to the scaling of the dependent variable, and a way better result on the test set: a 4% normalised RMSE and an accuracy of 97%. I then run diagnostics on this model: the outlier test showed the presence of 10 anomalous observation, the Q-Q plot highlighted the presence of many individuals at the extremities while displaying the residuals, it is clear that they follow a leptokurtic distribution (Figure 2).

Performing a k-fold cross validation, with k = 10, did not bring any ad-



(a) Q-Q plot (b) Studentized residuals

Figure 2: Regression diagnostics.

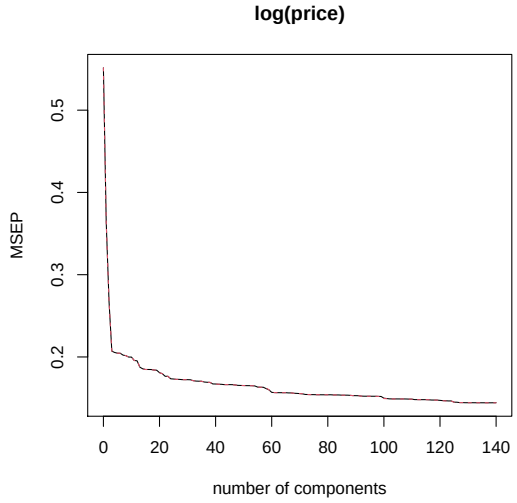


Figure 3: MSEP for Principal Component Regression.

vantage since the dataset is already large and resampling methods would not significantly improve the model. Instead, a more intuitive approach, considering the high number of variables included, would be applying subset selection methods. In particular I ran forward, backward and stepwise regression but, even in this case, the performance were similar to the best model I had found so far: the suggested number of variables was 138 on a total of 140 and the normalised RMSE was still around 4%, so I decided to discard these models too.

Shrinkage methods, like lasso and ridge regression, are useful when we want to reduce variance and operate variables selection since they shrink the coefficients towards zero. For both, I performed cross validation in order to find the best lambda, then ran the regression model and evaluated the result on the test set: if with ridge regression the results were in line with the previous models, with the lasso regression I even had a reduction of the performances.

The last attempt I made at improving my model was using dimension reduction methods, principal component regression and partial least squares, which are mainly used when there is an high correlation among the independent features. Starting from PCR, I performed the analysis on the logarithm of the price against the scaled version of the variables in my dataset. With only three principal components it is possible to explain 62.5% of the variance of price and the model achieves a prediction accuracy of 96.5%, not far away from the 97% of the linear model (Figure 3). With PLS, instead, using 3 components it is already possible to explain 69.5% of the variance of price and achieve a prediction accuracy on the test set of 96.9% (Figure 4).

When dealing with lots of categorical variables, linear regression can be outper-

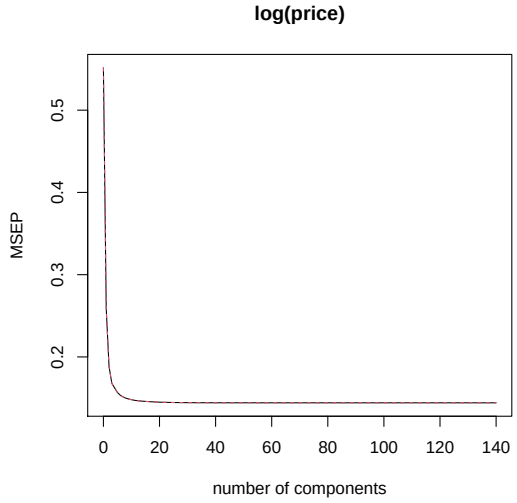


Figure 4: MSEP for Partial Least Squares.

formed by decision trees. In order to follow this approach, I had to make some changes in the dataset, one-hot enconding all the non-quantitative attributes. Training a simple algorithm on the original variables returned a regression tree with 13 terminal nodes and an accuracy of 71.8% on the test set, while applying the logarithm to the variable price I ended up with a tree with 10 terminal nodes and a 96.1% of accuracy on the test set (Figure 5).

Two ensemble methods to improve the stability and performance of decision trees are bagging and random forests. With bagging we generate multiple samples with replacement from the dataset (bootstrap), fit a learner for each sample and aggregate the results; with random forests, instead, we not only sample over the observations but also over the features, creating an even more robust learning algorithm. In fact, with the last one, I reach the highest accuracy among all the models I trained: 98.1%.

Conclusion

The objectives of the research were, first of all, to understand which attributes had the greatest impact on price values and, in the second place, to build a model capable of predicting the response attribute. The regression models explained more than 70% of the variance of the dependent variable and the interpretation of their coefficient can certainly indicate the impact of each regressor on price. Talking about the best model found, the log-level one, an important observation must be done: as a regressor increases by one unit, a percentage increase, corresponding to the value of the coefficient, is observed in the response vari-

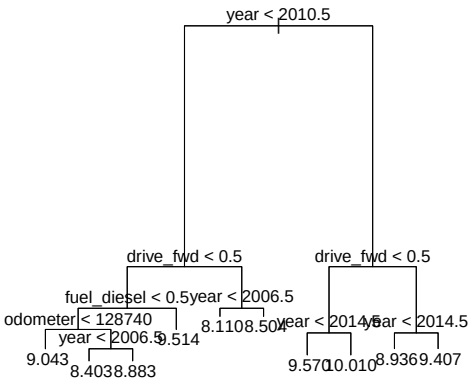


Figure 5: Simple regression tree.

able. For this model, the intercept is negative, mainly due to the fact that I did not scale the variables when estimating this model. The year has a positive coefficient (0.06), reflecting the relation already observed in the scatterplot. Almost all the manufacturer dummies are significant: positive coefficient are observed for companies like Porsche, Aston Martin or Tesla, whose cars seem to devalue less. As expectable, cars in salvage (-0.6) or fair (-0.58) condition are the ones with lower prices. Even the number of cylinders is significant, with lower prices for cars with less cylinders, while the coefficients for different fuels are all similar. A very small coefficient, -4.296e-06, is observed for odometer, but we must remember that this variables have very large values, in the order of hundreds of thousands; the negative value reflect the observation previously made looking at the scatter plot. Other interesting observation can be made on the fact that convertible cars and offroads costs generally more than sedan cars and SUVs, while dummies related to the paint colour have very low coefficient, even though some of them are still significant. Finally, variables related to the states are mostly non significant and, the ones who are significant instead, have very small negative coefficients.

Another interesting study can be performed on the loadings of the PCR and PLS models. In particular, with the last one, the explained variance was already 69.5% with just the first 3 components. Analysing the values of the loadings for this model, the first component captures mostly the variance due to year and odometer, with the first having a positive value while the second a negative one; going deeper in the other components, it is possible to find high loadings for cylinders dummies, some car types (truck, SUV), the dummy 'driverwd' and, surprisingly, for the paint colours white and silver.

At the end, with the last model, trained by random forests, the accuracy reached was very high (98.1%) and it can be certainly used in everyday predictions. A step further could be done in this sense, using the original dataset without any modification: random forests work very well even in presence of numerous missing values, therefore it could be possible to train an algorithm with decent performances on real datasets with lots of NAs.

An even deeper research, that was out of the scope of this study, could be done taking in consideration also the variable 'description' and using Natural Language Processing to perform text analysis.

R code

```
# Setup
library(tidyverse)
library(caret)
library(car)
library(glmnet)
library(pls)
library(MASS)
library(tree)
library(mltools)
library(data.table)
library(randomForest)
library(ipred)
set.seed(22)

# Import data
df <- read.csv('/Users/sergiopicascia/Desktop/vehicles.
    csv', na.strings = c('', 'NA', 'missing'))

str(df)
summary(df)
sapply(df, function(x) sum(is.na(x)))

# Removing columns in excess
vehicles <- subset(df, select = -c(id, url, region,
    region_url, model, image_url, description, county,
    lat, long, vin, size))

# Eliminating 'price' outliers
hist(vehicles$price[vehicles$price < 100000])
vehicles <- vehicles[(vehicles$price >= 1000) & (vehicles
    $price < 40000), ]

# Eliminating 'year' outliers, dropping NA
```

```r
hist(vehicles$year)
vehicles <- vehicles[(vehicles$year >= 1980) & (vehicles$
    year < 2021), ]
vehicles <- vehicles %>% drop_na(year)

# Dropping 'manufacturer' NA, recode as categorical
unique(vehicles$manufacturer)
vehicles <- vehicles %>% drop_na(manufacturer)
vehicles$manufacturer <- as.factor(vehicles$manufacturer)
vehicles <- vehicles[!(vehicles$manufacturer == 'ferrari'
    | vehicles$manufacturer == 'morgan'), ]

# Dropping 'condition' NA, recode as categorical
unique(vehicles$condition)
vehicles <- vehicles %>% drop_na(condition)
vehicles$condition <- as.factor(vehicles$condition)

# Dropping 'fuel' NA, recode as categorical
unique(vehicles$fuel)
vehicles <- vehicles %>% drop_na(fuel)
vehicles$fuel <- as.factor(vehicles$fuel)

# Eliminating 'odometer' outliers, dropping NA
hist(vehicles$odometer[vehicles$odometer < 500000])
vehicles <- vehicles[(vehicles$odometer < 300000), ]
vehicles <- vehicles %>% drop_na(odometer)

# Recode 'state' as categorical
unique(vehicles$state)
vehicles$state <- as.factor(vehicles$state)

# Dropping 'title_status' NA, recode as categorical
unique(vehicles$title_status)
vehicles <- vehicles %>% drop_na(title_status)
vehicles$title_status <- as.factor(vehicles$title_status)

# Recode 'cylinders' as categorical
unique(vehicles$cylinders)
vehicles <- vehicles %>% drop_na(cylinders)
vehicles$cylinders <- as.factor(vehicles$cylinders)

# Recode 'transmission' as categorical, dropping NA
unique(vehicles$transmission)
vehicles <- vehicles %>% drop_na(transmission)
vehicles$transmission <- as.factor(vehicles$transmission)
```

```
# Recode 'drive' as categorical
unique(vehicles$drive)
vehicles <- vehicles %>% drop_na(drive)
vehicles$drive <- as.factor(vehicles$drive)

# Recode 'type' as categorical
unique(vehicles$type)
vehicles <- vehicles %>% drop_na(type)
vehicles$type <- as.factor(vehicles$type)

# Recode 'paint_color' as categorical
unique(vehicles$paint_color)
vehicles <- vehicles %>% drop_na(paint_color)
vehicles$paint_color <- as.factor(vehicles$paint_color)

summary(vehicles)
sapply(vehicles, function(x) sum(is.na(x)))

# Plot vars against price
for (i in 1:length(vehicles[, -1])) {
 plot(vehicles[, i+1], vehicles$price, xlab = colnames(
     vehicles)[i+1], ylab = 'price', cex.axis = 0.7, las =
      2)
}

### Linear Regression
train_samples <- vehicles$price %>% createDataPartition(p
    = 0.8, list = F)
train_vehicles <- vehicles[train_samples, ]
test_vehicles <- vehicles[-train_samples, ]

lin_model <- lm(data = train_vehicles, price ~.)
summary(lin_model)
car::vif(lin_model)

lin_model_pred <- predict(lin_model, test_vehicles)
caret::R2(lin_model_pred, test_vehicles$price)
RMSE(lin_model_pred, test_vehicles$price)
RMSE(lin_model_pred, test_vehicles$price)/mean(test_
    vehicles$price)

compare <- cbind (actual = test_vehicles$price, lin_model
    _pred)
mean(apply(compare, 1, min)/apply(compare, 1, max))

# Scaled variables
```

```
train_scaled <- train_vehicles
test_scaled <- test_vehicles
train_scaled[, c(2, 7)] <- scale(train_vehicles[, c(2, 7)
    ], center = T, scale = T)
test_scaled[, c(2, 7)] <- scale(test_vehicles[, c(2, 7)],
     center = T, scale = T)

scaled_model <- lm(data = train_scaled, price ~.)
summary(scaled_model)

# Log model
train_log <- train_vehicles
train_log[, 1] <- log(train_log[, 1])
test_log <- test_vehicles
test_log[, 1] <- log(test_log[, 1])

train_log2 <- train_vehicles
train_log2[, c(1, 2)] <- log(train_log2[, c(1, 2)])
train_log2[, 7] <- log(train_log2[, 7] + 1)

train_log3 <- train_vehicles
train_log3[, 2] <- log(train_log3[, 2])
train_log3[, 7] <- log(train_log3[, 7] + 1)

log_model <- lm(data = train_log, price ~.) # Log-level
summary(log_model)
car::vif(log_model)

log_model2 <- lm(data = train_log2, price ~.) # Log-log
summary(log_model2)

log_model3 <- lm(data = train_log3, price ~.) # Level-log
summary(log_model3)

log_model_pred <- predict(log_model, test_log)
caret::R2(log_model_pred, test_log$price)
RMSE(log_model_pred, test_log$price)
RMSE(log_model_pred, test_log$price)/mean(test_log$price)

compare_log <- cbind (actual = test_log$price, log_model_
    pred)
mean(apply(compare_log, 1, min)/apply(compare_log, 1, max
    ))

# Regression Diagnostics
outlierTest(log_model)
```

```
qqPlot(log_model)

hist(studres(log_model), freq = F)
xfit <- seq(min(studres(log_model)), max(studres(log_
    model)), length=40)
yfit <- dnorm(xfit)
lines(xfit, yfit)

# K-fold Cross Validation
cv10_control <- trainControl(method = 'cv', number = 10)
log_model_cv <- train(data = train_log, price ~., method
    = 'lm', trControl = cv10_control)
summary(log_model_cv)

log_model_cv_pred <- predict(log_model_cv, test_log)
RMSE(log_model_cv_pred, test_log$price)/mean(test_log$
    price)

# Backward Regression
back_model <- train(data = train_log, price ~., method =
    'leapBackward', tuneGrid = data.frame(nvmax = 1:139),
    trControl = cv10_control)
back_model$results
back_model$bestTune

# Forward Regression
forw_model <- train(data = train_log, price ~., method =
    'leapForward', tuneGrid = data.frame(nvmax = 1:139),
    trControl = cv10_control)
forw_model$results
forw_model$bestTune

# Stepwise Regression
step_model <- train(data = train_log, price ~., method =
    'leapSeq', tuneGrid = data.frame(nvmax = 1:139),
    trControl = cv10_control)
step_model$results
step_model$bestTune

step_model_pred <- predict(step_model, test_log)
RMSE(step_model_pred, test_log$price)/mean(test_log$price
    )

# Ridge Regression
x <- model.matrix(data = train_log, price ~.)
```

```
test_x <- model.matrix(data = test_log, price ~.)
ridge_cv <- cv.glmnet(x, train_log$price, alpha = 0)
lambda <- ridge_cv$lambda.min
plot(ridge_cv)

ridge_model <- glmnet(x, train_log$price, alpha = 0,
    lambda = lambda)
ridge_pred <- predict(ridge_model, s = lambda, newx =
    test_x)
RMSE(ridge_pred, test_log$price)/mean(test_log$price)
compare_ridge <- cbind (actual = test_log$price, ridge_
    pred)
mean(apply(compare_ridge, 1, min)/apply(compare_ridge, 1,
     max))

# Lasso Regression
lasso_cv <- cv.glmnet(x, train_log$price, alpha = 0)
lambda2 <- lasso_cv$lambda.min
plot(lasso_cv)

lasso_model <- glmnet(x, train_log$price, alpha = 1,
    lambda = lambda2)
lasso_pred <- predict(lasso_model, s = lambda2, newx =
    test_x)
RMSE(lasso_pred, test_log$price)/mean(test_log$price)
compare_lasso <- cbind (actual = test_log$price, lasso_
    pred)
mean(apply(compare_lasso, 1, min)/apply(compare_lasso, 1,
     max))

# Principal Component Regression
pcr_model <- pcr(log(price) ~., data = train_scaled,
    validation = 'CV')
summary(pcr_model)
validationplot(pcr_model, val.type = 'MSEP')
validationplot(pcr_model, val.type = 'R2')
predplot(pcr_model)

test_scaled_log <- test_scaled
test_scaled_log[, 1] <- log(test_scaled_log[, 1])

pcr_pred <- predict(pcr_model, test_scaled_log, ncomp =
    3)
RMSE(pcr_pred, test_scaled_log$price)
RMSE(pcr_pred, test_scaled_log$price)/mean(test_scaled_
    log$price)
```

```
compare_pcr <- cbind(actual = test_scaled_log$price, pcr_
    pred)
mean(apply(compare_pcr, 1, min)/apply(compare_pcr, 1, max
    ))

coefplot(pcr_model, ncomp = c(1, 2, 3))

# Partial Least Squares
pls_model <- plsr(log(price) ~., data = train_scaled,
    validation = 'CV')
summary(pls_model)
validationplot(pls_model, val.type = 'MSEP')
validationplot(pls_model, val.type = 'R2')
predplot(pls_model)

pls_pred <- predict(pls_model, test_scaled_log, ncomp =
    3)
RMSE(pls_pred, test_scaled_log$price)
RMSE(pls_pred, test_scaled_log$price)/mean(test_scaled_
    log$price)

compare_pls <- cbind(actual = test_scaled_log$price, pls_
    pred)
mean(apply(compare_pls, 1, min)/apply(compare_pls, 1, max
    ))

# Tree Based Methods
train_vehicles <- as.data.table(train_vehicles)
test_vehicles <- as.data.table(test_vehicles)
oh_train <- data.frame(one_hot(train_vehicles))
oh_test <- data.frame(one_hot(test_vehicles))

tree_model <- tree(data = oh_train, price ~.)
summary(tree_model)
plot(tree_model)
text(tree_model, pretty = 0)
tree_model

tree_pred <- predict(tree_model, oh_test)
compare_tree <- cbind (actual = oh_test$price, tree_pred)
mean(apply(compare_tree, 1, min)/apply(compare_tree, 1,
    max))

# Tree with log(price)
train_log <- as.data.table(train_log)
```

```
test_log <- as.data.table(test_log)
oh_train_log <- data.frame(one_hot(train_log))
oh_test_log <- data.frame(one_hot(test_log))

tree_log_model <- tree(data = oh_train_log, price ~.)
summary(tree_log_model)
plot(tree_log_model)
text(tree_log_model, pretty = 0)
tree_log_model

tree_log_pred <- predict(tree_log_model, oh_test_log)
compare_tree_log <- cbind (actual = oh_test_log$price,
    tree_log_pred)
mean(apply(compare_tree_log, 1, min)/apply(compare_tree_
    log, 1, max))

# Bagging
bag_model <- bagging(formula = price ~., data = oh_train_
    log, nbagg = 100, coob = T)
bag_model
bag_pred <- predict(bag_model, oh_test_log)
compare_bag <- cbind (actual = oh_test_log$price, bag_
    pred)
mean(apply(compare_bag, 1, min)/apply(compare_bag, 1, max
    ))

# Random Forest (it takes a lot of time to run!)
rf_model <- randomForest(data = oh_train_log, price ~.)
rf_model
rf_pred <- predict(rf_model, oh_test_log)
compare_rf <- cbind (actual = oh_test_log$price, rf_pred)
mean(apply(compare_rf, 1, min)/apply(compare_rf, 1, max))
```