Used Cars Price Prediction Model Training

Roxy Zhang

3/25/2022

Contents

Introduction

Since the market of used cars is growing, there is an increasing need of the information of used cars prices. Therefore, I want to build a model predicting the price using attributes of the car as predictors, and thus provide information for the potential buyers for decision making.

Our data is from Kaggle. This dataset contains 22 (1998-2019) years of used cars information. There are 5872 rows and 11 predictors, among which 5 are numeric and 6 are categorical.

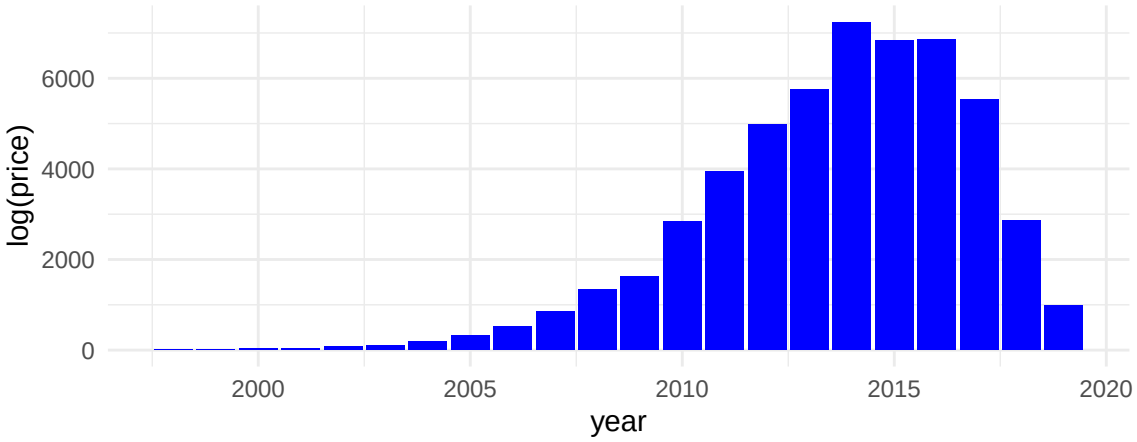
Data Cleaning

For better illustration, I transformed the original unit (100000 Indian Rupee) of response variable **price** to USD, using the exchange rate of 100000 Indian Rupee equals to 1309.75 USD on March 21, 2022. For numeric variables with unit, the units are deleted for model-building. I also delete the car model from the original `name` variable, keeping only the brand name, since the model can be greatly explained by the predictors I are using. The type `Fourth` in owner_type is actually Fourth and above. For analysis completeness, I dropped NA values. I also filtered out a row containing an extremely large value of kilometers_driven.

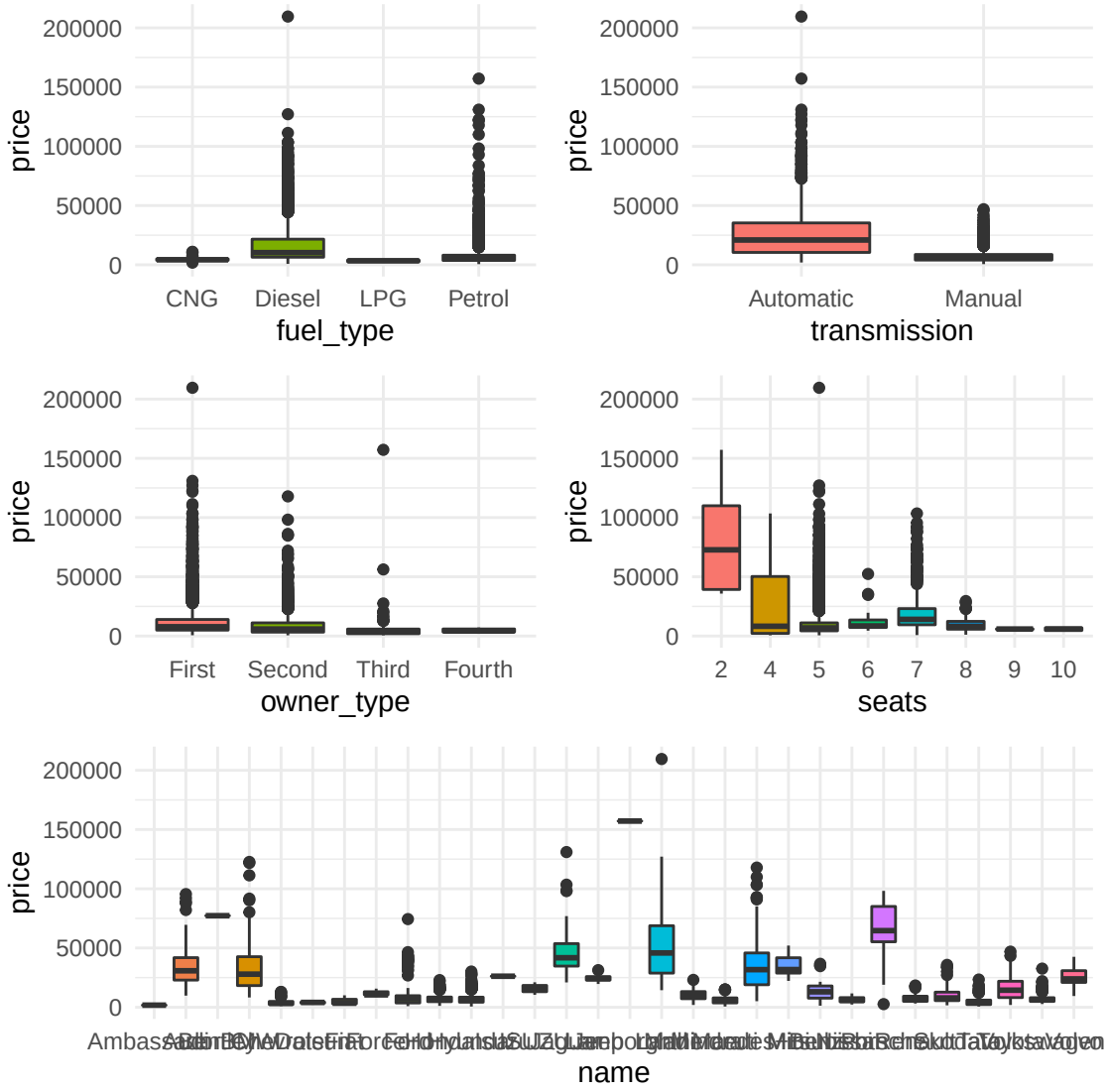
Below are the variable used:

name: The brand of the car.
location: The location in which the car is being sold or is available for purchase.
year: The year or edition of the model.
kilometers_driven: The total kilometres driven in the car by the previous owner(s) in KM.
fuel_type: The type of fuel used by the car.
transmission: The type of transmission used by the car.
owner_type: Whether the ownership is Firsthand, Second hand or other.
mileage: The standard mileage offered by the car company in kmpl or km/kg.
engine: The displacement volume of the engine in cc.
power: The maximum power of the engine in bhp.
seats: The number of seats in the car.
price: The price of the used car in US Dollar.

Exploratory Data Analysis



First, I plot the price against year. For better scaling, the price is log-transformed. There is an increasing trend of price from 1998-2014, while the price goes down rapidly since 2015.

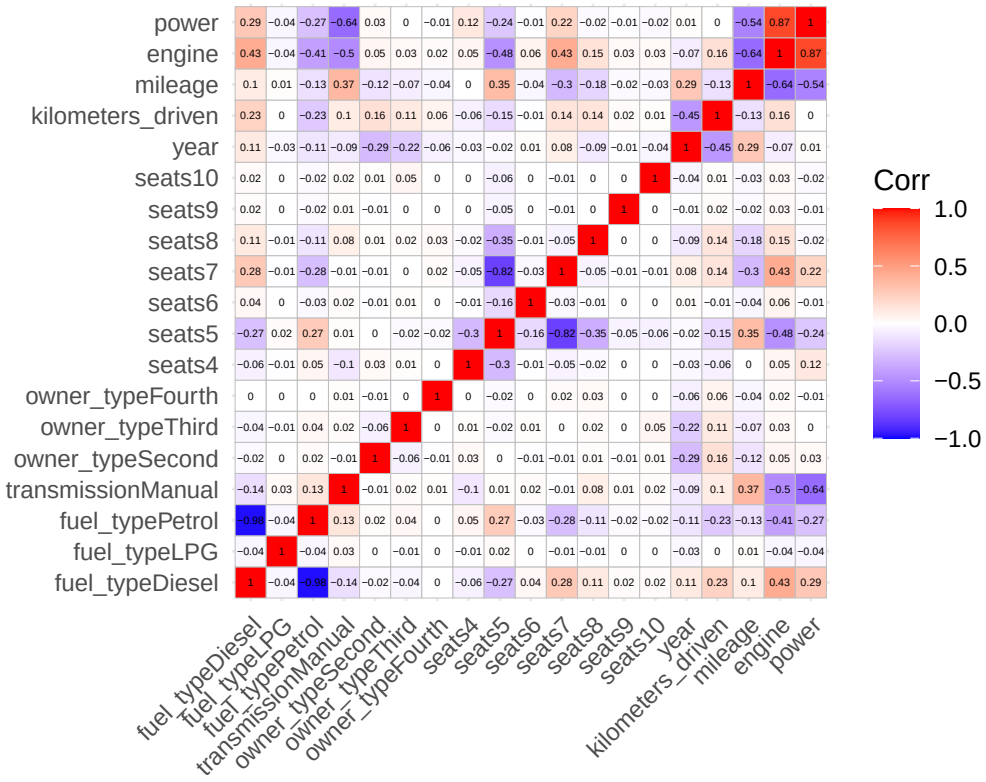


Then I look at the price distribution for 5 categorical predictors respectively. There are some interesting findings:

1. There are several attributes associated with higher price: fuel type diesel, automatic transmission, first-hand owner, and cars with 2 seats.

2. The price range for each brand varies a lot. For example, Lamborghini has a condensed price range slightly above 150k (which is not surprising), while the majority stays below 25k.

3. There is a relatively large number of outliers observed, indicating the big variance of our response variable. This may because of the volatility of used cars market, which in turn suggests the significance of our project.



For better visualization, categorical variables with over ten terms (`name`, `location`) are filtered out in the correlation plot. From the upper-right corner of the plot we can see a strong positive correlation between `engine` and `power`. Besides, `milage` is negatively associated with both `engine` and `power`.

Data Partitioning

The data is split into training data and testing data by the proportion of 0.8:0.2.

```
index_train = createDataPartition(
  y = car$price,
  p = 0.8,
  list = FALSE
)

train_df = car[index_train, ]
test_df = car[-index_train, ]

train_x = model.matrix(price ~ ., train_df)[ ,-1]
train_y = train_df$price

test_x = model.matrix(price ~ ., test_df)[ , -1]
test_y = test_df$price
test_all = model.matrix(price ~ ., test_df) # for nent prediction
```

Model Building

4 models are used to fit the training data: Lasso, Elastic Net, Partial least squares (PLS), and Multivariate Adaptive Regression Spline (MARS). Cross validation is used to select the best parameter or parameter combination for each model.

Lasso

The tuning parameter λ controls the L1 regularization, as λ increases, the number of predictors in the model decreases. Setting the candidate values of λ to be from 0.1353353 to 54.59815 with 100 steps, the best-tune λ is 2.069. All five numeric predictors are included in the final model.

Elastic Net

Elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the Lasso and Ridge methods. In addition to setting and choosing a lambda value, elastic net also allows us to tune the alpha parameter where $\alpha = 0$ corresponds to Ridge and $\alpha = 1$ to Lasso. The optimum alpha chosen is 0.25, and the optimum λ is 5.136.

Partial least squares (PLS)

Instead of finding hyperplanes of maximum variance between the response and independent variables, PLS finds a linear regression model by projecting the predicted variables and the observable variables to a new space. The tuning parameter - number of components in the final model is 18.

Multivariate Adaptive Regression Spline (MARS)

MARS is a non-parametric regression technique and can be seen as an extension of linear models, which can can be used to model nonlinear relationships and interactions between a set of predictor variables and a response variable. The best tune metrics are nprune = 12, degree = 2. There are 12 terms in the final model.

```
# models
set.seed(0324)

lasso_fit = train(price ~ .,
                  data = train_df,
                  method = "glmnet",
                  preProcess = c("center", "scale", "zv"), # zv for zero variance
                  tuneGrid = expand.grid(alpha = 1,
                                         lambda = exp(seq(4, -2, length = 100))),
                   trControl = trainControl(method = "cv"))

enet_fit = train(x = train_x,
                  y = train_y,
                  method = "glmnet",
                  preProcess = c("center", "scale", "zv"),
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(-2, 7, length = 100))),
                  trControl = trainControl(method = "cv"))
```

```
pls_fit = train(x = train_x,
                y = train_y,
                method = "pls",
                tuneGrid = data.frame(ncomp = 1:19),
                trControl = trainControl(method = "cv"),
                preProcess = c("center", "scale", "zv"))

mars_grid = expand.grid(
  degree = 1:3,
  nprune = 2:15)

mars_fit = train(x = train_x,
                 y = train_y,
                 method = "earth",
                 tuneGrid = mars_grid,
                 trControl = trainControl(method = "cv"))
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```
lasso_predict = predict(lasso_fit, newdata = test_df)
enet_predict = predict(enet_fit, newdata = test_all)
pls_pred = predict(pls_fit, newdata = test_x)
#mars_pred = predict(pls_fit, newdata = test_x)
```

```
# tuning parameters plots
#t1 =

#t2
#t3

t4 = ggplot(mars_fit)
#t4

#(t1 + t2)/(t3 + t4)
```

Model limitations

The penalty function of Lasso has several limitations. For example (not the case of this data), in the "large p, small n" case (high-dimensional data with few examples), the Lasso selects at most n variables before it saturates. Also if there is a group of highly correlated variables, then the Lasso tends to select one variable from a group and ignore the others. As is mentioned above, there are some association among `power`, `engine` and `mileage`, thus the lasso model's performance is not so well. To overcome these limitations, the elastic net adds a quadratic part to the penalty, which when used alone is ridge regression.

Conclusions

What Ire your findings? Are they what you expect? What insights into the data can you make?