Used cars price prediction

Lauren Gallego Ropero

2024-09-17

Introduction

The data set is a collection of valuable data about used cars that have been advertised on an online platform to be sold, and it includes the following information:

- The date the add was first seen by a crawler
- The name of the car
- Seller type: Private/Company
- Offer type: Offer/Request
- The car price
- A/B testing information: Control/Test
- Vehicle type
- Year of registration
- Gearbox: Manual/Automatic
- Horse Power
- Model
- Kilometers driven so far
- Month of registration
- Fuel type: Gasoline/Diesel
- Brand
- If there is any non-repaired damage in the vehicle
- Advertisement creation date
- Number of pictures included in the advertisement
- The date the add was last seen by a crawler

Import the data and take a quick glance at the variable characteristics from the summary.

```r
autos <- read.csv('cars.csv',header = T, dec = '.', sep = ',')
summary(autos)
```

```
##      index         dateCrawled            name              seller         
##  Min.   :     6   Length:52529       Length:52529       Length:52529      
##  1st Qu.: 92516   Class :character   Class :character   Class :character  
##  Median :185293   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :185527                                                           
##  3rd Qu.:278046                                                           
##  Max.   :371524                                                           
##                                                                           
##   offerType             price           abtest          vehicleType       
##  Length:52529       Min.   :   235   Length:52529       Length:52529      
##  Class :character   1st Qu.:  1350   Class :character   Class :character  
##  Mode  :character   Median :  3350   Mode  :character   Mode  :character  
##                     Mean   :  6060                                        
##                     3rd Qu.:  7800                                        
##                     Max.   :145000                                        
##                                                                           
##  yearOfRegistration   gearbox             powerPS          model          
##  Min.   :1910       Length:52529       Min.   : 20.0   Length:52529      
##  1st Qu.:2000       Class :character   1st Qu.: 73.0   Class :character  
##  Median :2004       Mode  :character   Median :105.0   Mode  :character  
##  Mean   :2003                          Mean   :117.9                     
##  3rd Qu.:2008                          3rd Qu.:150.0                     
##  Max.   :2016                          Max.   :751.0                     
##                                                                          
##    kilometer      monthOfRegistration   fuelType            brand          
##  Min.   : 10000   Min.   : 1.000      Length:52529       Length:52529      
##  1st Qu.:100000   1st Qu.: 3.000      Class :character   Class :character  
##  Median :150000   Median : 6.000      Mode  :character   Mode  :character  
##  Mean   :124007   Mean   : 6.123                                          
##  3rd Qu.:150000   3rd Qu.: 9.000                                          
##  Max.   :150000   Max.   :12.000                                          
##                                                                           
##      Damage       dateCreated         nrOfPictures   postalCode   
##  Min.   :0.000   Length:52529       Min.   :0      Min.   : 1067  
##  1st Qu.:0.000   Class :character   1st Qu.:0      1st Qu.:31137  
##  Median :0.000   Mode  :character   Median :0      Median :50374  
##  Mean   :0.099                      Mean   :0      Mean   :51479  
##  3rd Qu.:0.000                      3rd Qu.:0      3rd Qu.:72414  
##  Max.   :1.000                      Max.   :0      Max.   :99988  
##  NA's   :6697                                                     
##    lastSeen        
##  Length:52529      
##  Class :character  
##  Mode  :character  
##                    
##                    
##                    
##                    
```

Data Pre-processing

Categorical values must be transformed into factors

```r
autos$abtest <- as.factor(autos$abtest)
autos$vehicleType <- as.factor(autos$vehicleType)
autos$gearbox <- as.factor(autos$gearbox)
autos$fuelType <- as.factor(autos$fuelType)
autos$Damage <- as.factor(autos$Damage)
autos$seller <- as.factor(autos$seller)
autos$offerType <- as.factor(autos$offerType)
```

There are some columns that are unnecessary and can be omitted:

- Too many categories to handle

```r
length(unique(autos$model))
```

```
## [1] 234
```

```r
length(unique(autos$brand))
```

```
## [1] 40
```

- Same values for all observations

```r
summary(autos$nrOfPictures)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

```r
summary(autos$seller)
```

```
##     P
## 52529
```

```r
summary(autos$offerType)
```

```
##     O     R
## 52528     1
```

Delete columns

```r
del_cols <- c('index','name','model','brand','nrOfPictures','postalCode','seller','offerType')
autos <- autos [,!names(autos) %in% del_cols]
```

Missing Values

```r
na_counts <- colSums(is.na(autos))
na_counts <- data.frame(Column = names(na_counts),
                        NA_Counts = na_counts,
                        NA_prop = na_counts / dim(autos)[1])
has_na <- which(na_counts$NA_Counts != 0)
na_counts[has_na,]
```

```
##              Column NA_Counts    NA_prop
## gearbox   gearbox        784 0.01492509
## fuelType fuelType       2061 0.03923547
## Damage     Damage       6697 0.12749148
```

Since the null value proportion of the columns gearbox and fuelType are very low, it is safe to remove those rows, which represent less than 5% of the data

Finally, for the null values in the damage column, I are going to use an imputation method, since 12% is a more significant proportion. For this, I will use the MICE algorithm, and, since are only working with a binary column, the appropriate model is logistic regression.

Feature Engineering

There are 3 variables in the clean data than I cannot use given their date format. However, with some transformations, I can make use of the information that they include within their relationships.

If a car advertisement has been posted for a long time, this might have resulted in a potential price decrease, given the lack of demand. In order to obtain how long an add has been posted, I can calculate the time difference between the variables dateCrawled and lastSeen.

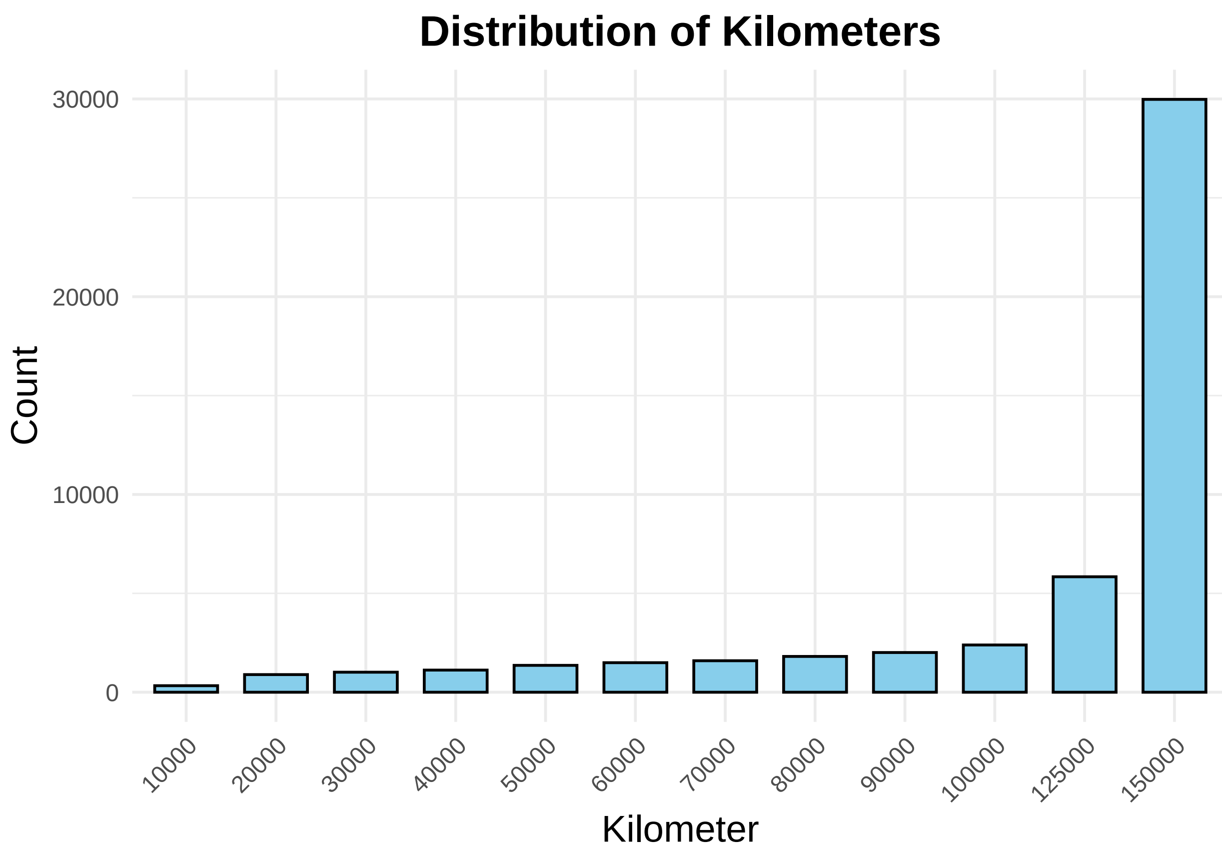
```r
autos$posted <- difftime(autos$lastSeen ,autos$dateCrawled,
                         units = 'days')
autos$posted <- round(as.numeric(autos$posted), 2)

autos <- autos[,-c(1,12,13)]
```

After a quick overview at the variable kilometer, I realized that it seems to be distributed into numeric categories rather than continuously. Let's take a deeper look at it, how many categories are there?

```r
length(unique(autos$kilometer))
```

```
## [1] 12
```

Now, transform it into a factor to visually analyze these categories and their distribution



In the plot above, see how the number of used cars and the kilometers have a positive correlation. In addition, notice the extraordinary proportion of cars marked as 150 000 kilometers. Given this bizarre distribution, assume that this category belongs to those cars with at least 150 thousand kilometers.

Ideally, I would have the exact number of kilometers for every observation, treating it as a continuous variable, however, this is not the case. Therefore, the approach that I will follow will be to convert it into a factor, where every of the categories above will have their own numeric index following an ascending order. This will leave us with a total of 12 categories.

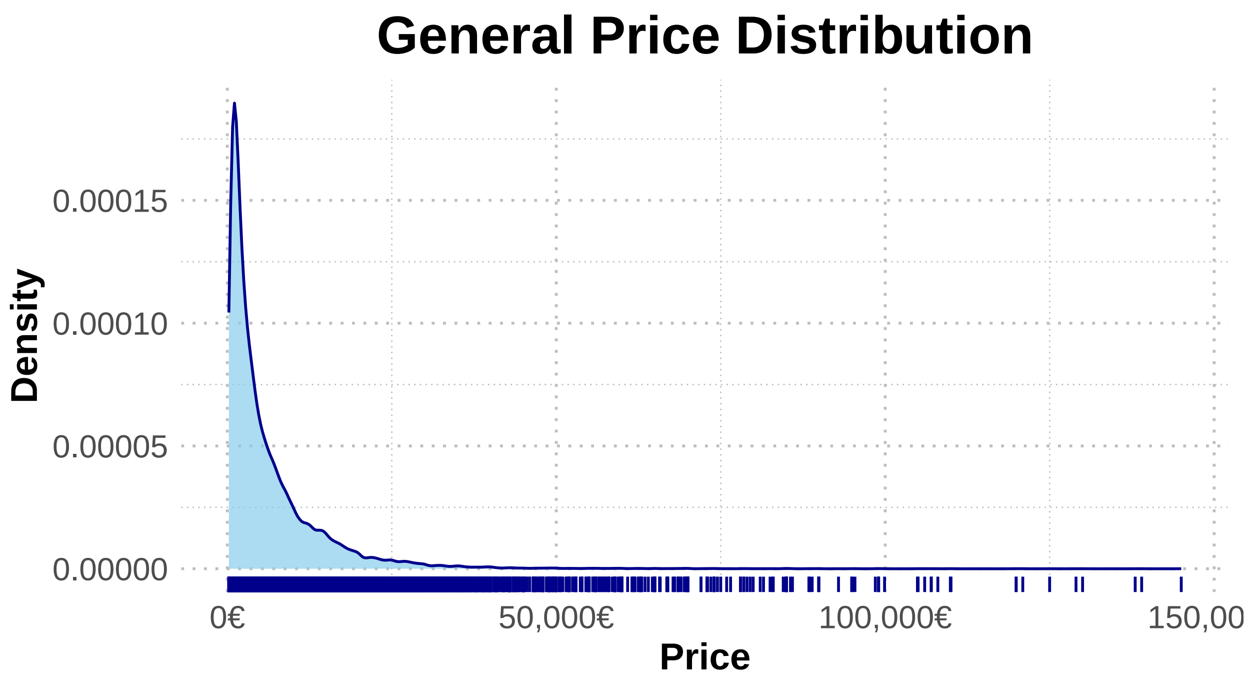
```
autos$kilometer <- as.factor(autos$kilometer)
autos$kilometer <- as.integer(autos$kilometer)
```

Another interesting variable is the year of registration. For a better understanding, this variable will be transformed into the car's age. This can be done since all the dates in the data correspond to the year 2016.

Note that I could express the car age in months, however, the price of used cars does not generally suffer short term changes to a degree where I would be losing insights by expressing age in years. Therefore, and for a better interpretability, I will discard the use of months.

```
autos$years <- 2016 - autos $yearOfRegistration
autos$years <- as.integer(autos$years)

autos$monthOfRegistration <- NULL
autos$yearOfRegistration <- NULL
```

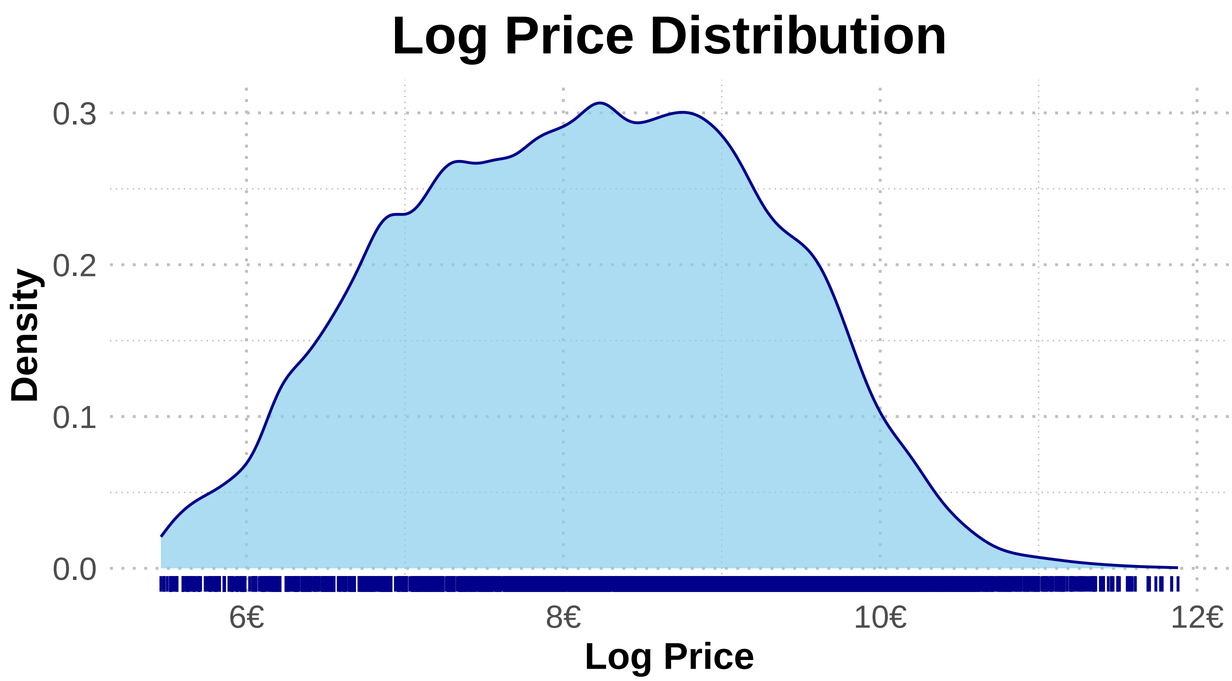
It is also important to analyze the distribution of the dependent variable. In order to do so, I will visualize it:



The dataset covers a large range of prices, however, the plot shows how cars over 25,000€ represent a minuscule proportion of the data. A great way to deal with this type of distribution is to apply a logarithmic transformation

```
autos$log_price <- round(log(autos$price),2)
```

Let's take a look at the distribution of the new variable:



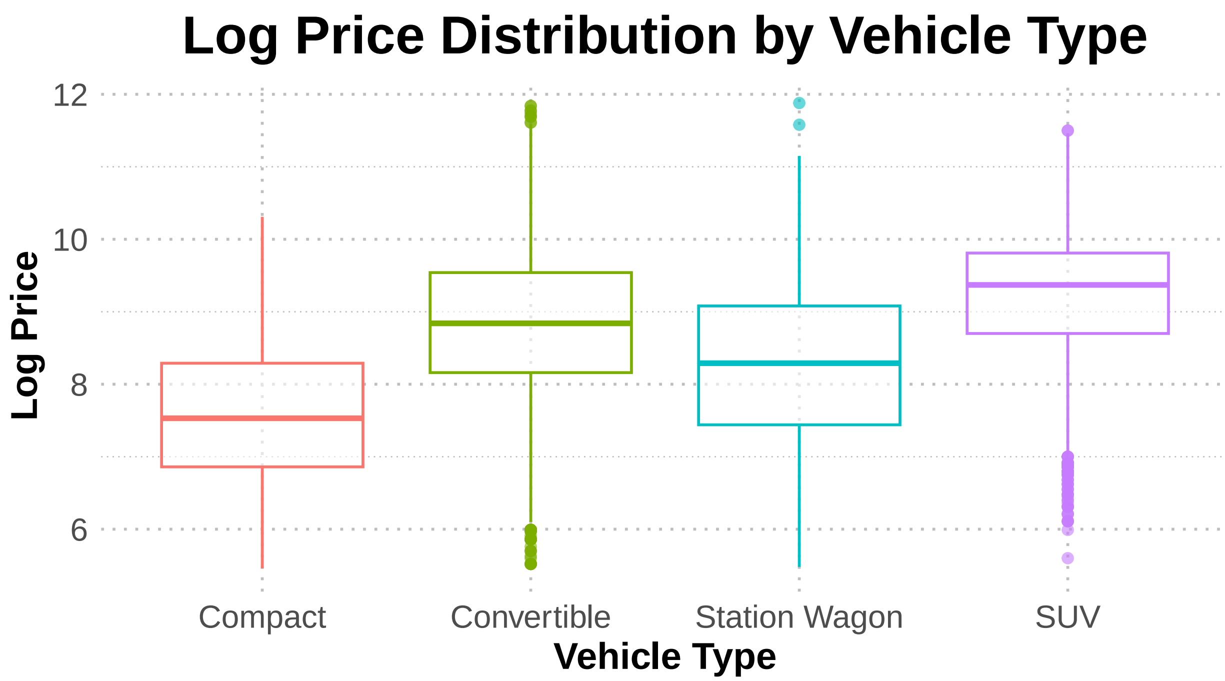
The log transformation has skewed the previous distribution, stabilizing variance and shrinking the range of values. This has resulted in a much more normally distributed variable, which will make a more effective analysis. It is essential to consider that this transformation may imply the loss of some interpretability, and will require an inverse transformation for the predictions.

Outlier Removal

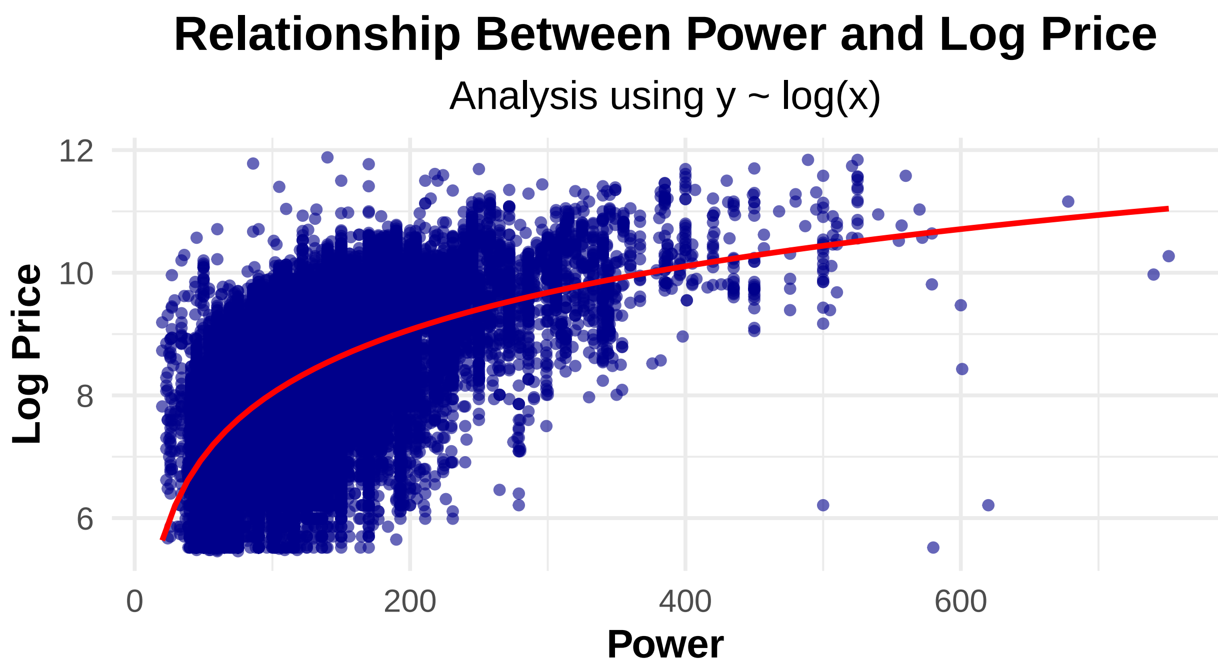
The model will focus on predicting the price of ordinary used cars, since the price of special cars can depend on other factors such as location, availability, transport, modifications or maintenance, none of which are considered in the data.

Because of this, I must find those observations that are significantly different from the others, specially when it comes to price.

Here are some plots that can help us understand the price distribution of used cars depending on different variables.



The plot shows how different vehicle types show various distributions, where SUVs tend to be the most expensive and Compact cars the least.
By making use of boxplots, I can easily spot outliers. This is done using the IQR method, which I will now consider to remove such observations.



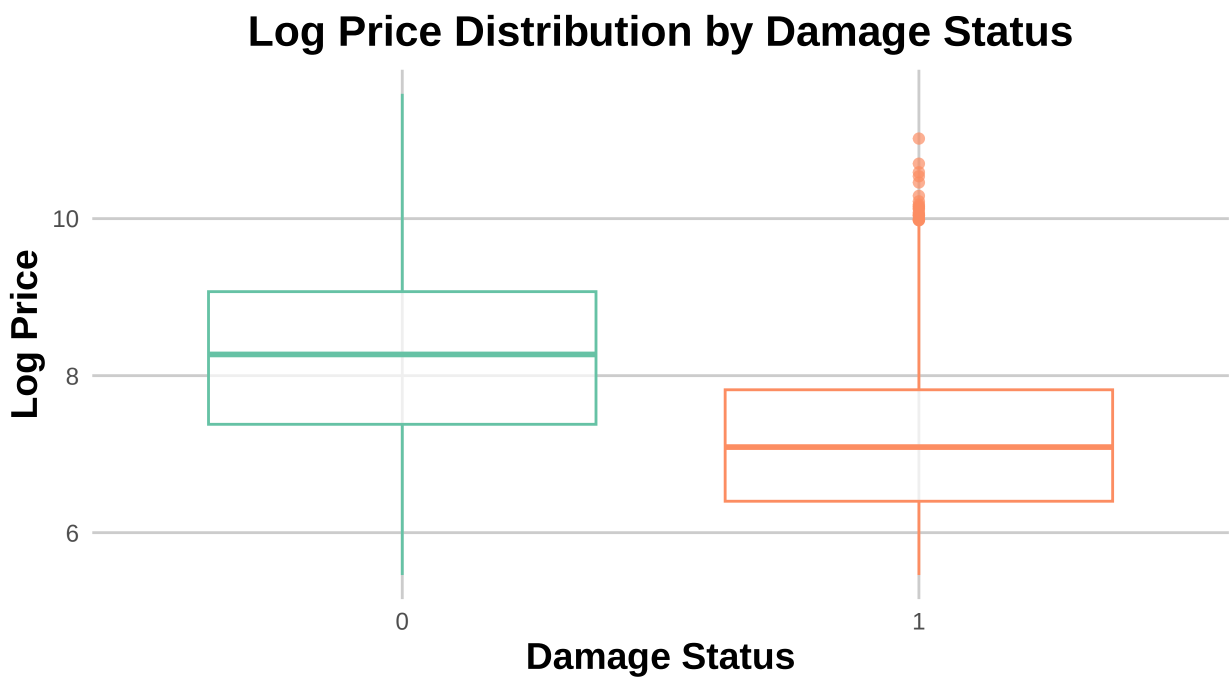
The plot shows how cars with 0−200 HP can have very different prices, this is due to the influence of other variables such as damage, kilometers or age. Meanwhile cars with 200−400 HP suffer a rather linear minimum log price increase, since they tend to be more valuable despite other inconveniences. Finally, notice that, because of the log transformation, the log price is upper bounded at around 12 for any HP range.

Now, I will remove the outliers using a vehicle type division.

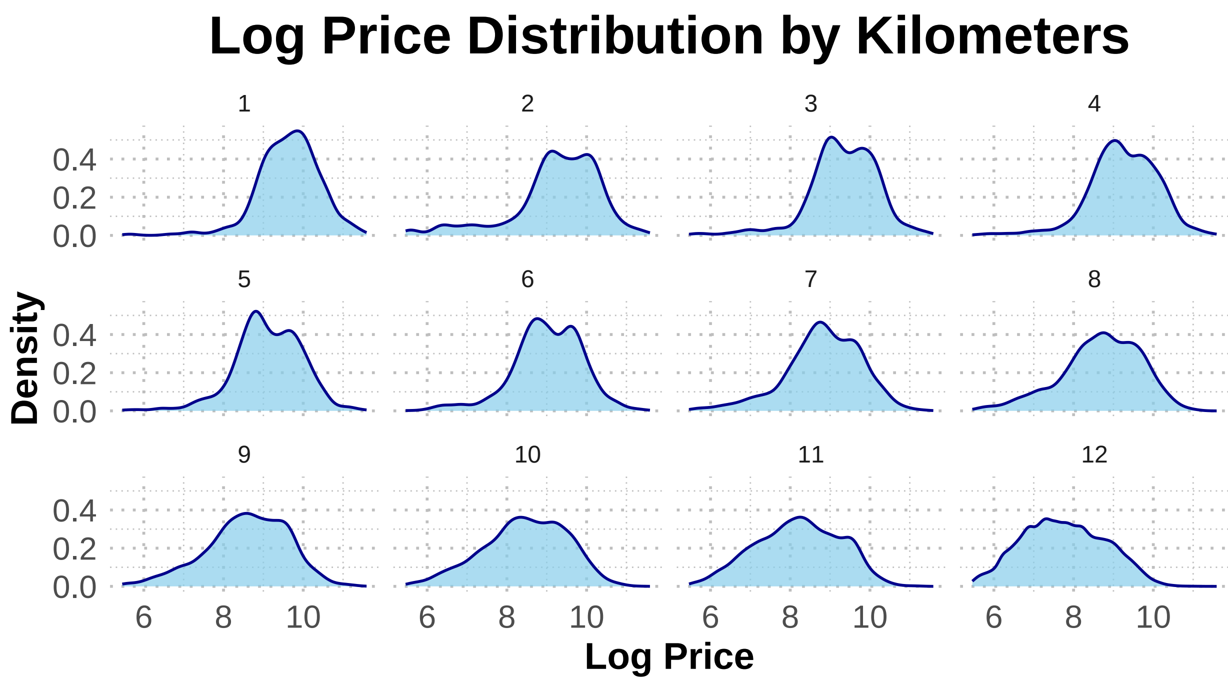
```
outlier_indices <- integer(0)

for (type in levels(autos$vehicleType)) {
  rows <- which(autos$vehicleType == type)
  IQR_value <- IQR(autos$log_price[rows])
  upr <- quantile(autos$log_price[rows], 0.75) + IQR_value * 1.5
  lwr <- quantile(autos$log_price[rows], 0.25) - IQR_value * 1.5
  outliers <- which(autos$vehicleType == type &
                      (autos$log_price <= lwr | autos$log_price >= upr))
  outlier_indices <- c(outlier_indices, outliers)
}
autos <- autos[-outlier_indices,]
```

Visualization

Before beginning the model building process, let's visualize a few more variables so that I can get more insights about their categories and how their semantic meaning is shown by the data.



While both categories cover observations along the entire price range, I can clearly notice that damaged cars tend to have a lower price by looking at both medians. This perfectly aligns to the semantic meaning of both categories



Kilometer classes 1−7 follow very similar distributions, where most of the values are located at log_price 8−10 and the peak is close to 9. However, as the kilometers increase(classes 8−12), the distribution becomes much smoother, with most values within the 6−10 log_price range, and a shorter peak much closer to 8.

Model Building

After thoroughly analyzing and cleaning the data, it is now time to create themodel. In order to find the best model, different algorithms will be tested and compared.

Statistical Models

Stepwise Regression

Firstly, I will make use of a comparison between a null model and a model using all of the numeric predictors. Using stepwise regression will allow us to understand the influence of these predictors and show us a statistical analysis of the best model.

```
full <- lm(log_price ~ powerPS + kilometer + posted + Damage + years, autos)
null <- lm(log_price ~ 1, autos)
stepwise_model <- stepAIC(null,
                          scope = list(lower = null, upper = full),
                          direction = "both",trace = F)
summary(stepwise_model)
```

```
##
## Call:
## lm(formula = log_price ~ years + powerPS + kilometer + Damage +
```

```
##     posted, data = autos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5318 -0.3335  0.0258  0.3428  6.9668
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.946e+00  1.317e-02  679.11   <2e-16 ***
## years       -7.020e-02  5.422e-04 -129.46   <2e-16 ***
## powerPS      9.318e-03  4.821e-05  193.30   <2e-16 ***
## kilometer   -1.038e-01  1.153e-03  -90.04   <2e-16 ***
## Damage1     -6.257e-01  9.431e-03  -66.35   <2e-16 ***
## posted       9.321e-03  3.366e-04   27.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6246 on 49696 degrees of freedom
## Multiple R-squared:  0.6983, Adjusted R-squared:  0.6983
## F-statistic: 2.301e+04 on 5 and 49696 DF,  p-value: < 2.2e-16
```

Let's check the multicollinearity of the model, as well as the AIC for some evaluation.

```r
vif(stepwise_model)
```

```
##     years   powerPS kilometer    Damage    posted
##  1.342925  1.080555  1.269248  1.025614  1.019765
```

```r
AIC(stepwise_model)
```

```
## [1] 94277.7
```

By looking at the summary of the final model, I quickly notice that, despite the low p-value of all of the numeric predictors, the relatively low R-squared value shows that there is still a significant proportion of the data variability that cannot be represented by a linear relationship.

In addition, the low VIF discards multicollinearity, and the large AIC confirms that, as I expected, a linear model is not suitable.

ML Models

Firstly, I have separated my data into training and test sets, and I have also defined the train control procedure. In this case, I will be using Cross Validation with 5 folds.

Below, I have included the training process code for both of the algorithms I used: K-Nearest Neighbors and Random Forest.

Due to limited computational resources, I have trained the models in a different environment, and then imported the hyper-parameter grids with the results back into this report.

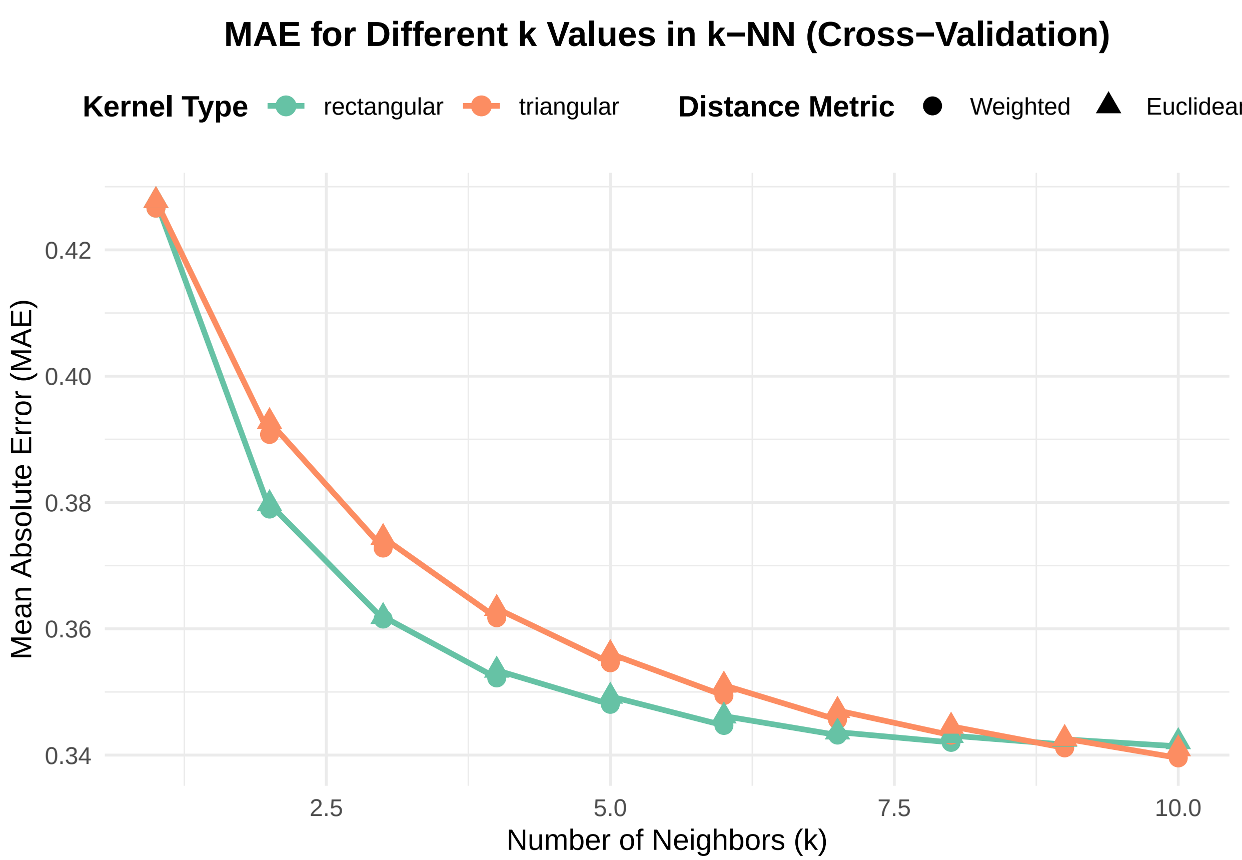
```r
set.seed(42)

# Delete the price variable, since I will work with the log price
data <- autos[,-1]

# Training/test sets
indexes <- sample(1:nrow(data), size = round(0.8 * nrow(data)), replace = F)
train_data = data[indexes,]
test_data = data[-indexes,]

train_control_knn <- trainControl(method = "cv",
                                  number = 5,                # 5-fold cross-validation
                                  savePredictions = "final", # Save predictions for later analysis
                                  verboseIter = TRUE)        # Print progress of training

train_control_rf <- trainControl(method = "cv", number = 5, verboseIter = TRUE)
```

K-Nearest Neighbors

Import the results.

```r
knn_results = read.csv("knn_results.csv", header = T, dec = ".", sep = ",")
```

Plot test accuracy (MAE) vs k.



By looking at the plot, it is clear that using a rectangular kernel leads to better results. Regarding the distance metric, there are no significant differences between weighted and euclidean distances. Therefore, whichever can be chosen. Using the elbow method, the optimal number of neighbors would be 4, although, with some extra computational flexibility, one could choose 5 in order to claim a 0.35 MAE upper bound.

The best model uses the following parameters:

- k = 4
- kernel = Rectangular
- distance = Euclidean

Let's take a look at this model's results to then compare them to the next algorithm, this will tell us which will be the final model.

```
best_index <- which(knn_results$kmax == 4 & knn_results$distance == 2.0
                   & knn_results$kernel == "rectangular")
print(knn_results[best_index,4:9])
```

```
##         RMSE  Rsquared       MAE      RMSESD  RsquaredSD      MAESD
## 15 0.4769522 0.8245309 0.3534298 0.003734674 0.003430516 0.00265539
```

The Rsquared of the model confirms its significance with respect to the data, and since the optimal k has been chosen, the MAE is balanced with the computational cost.

Random Forest

```
# I will use the following hyperparameter combination
grid_rf <- expand.grid(
  mtry = c(4, 5, 6),
  ntree = c(100, 300, 500),
  nodesize = c(10, 15, 20)
)

# Add the MAE to the grid
set.seed(42)
grid_rf$MAE <- rep(0, nrow(grid_rf))
MAE_CV <- rep(0,5)
```

Training process

Import results to environment

```
grid_rf <- read.csv("rf_results.csv", header = T, sep = ",", dec = ".")
```

Select the best model and train it

```
best_index <- which.min(grid_rf$MAE)
```

Import the best model object, which has been trained at a more powerful environment.

```r
load("best_model.RData")
```

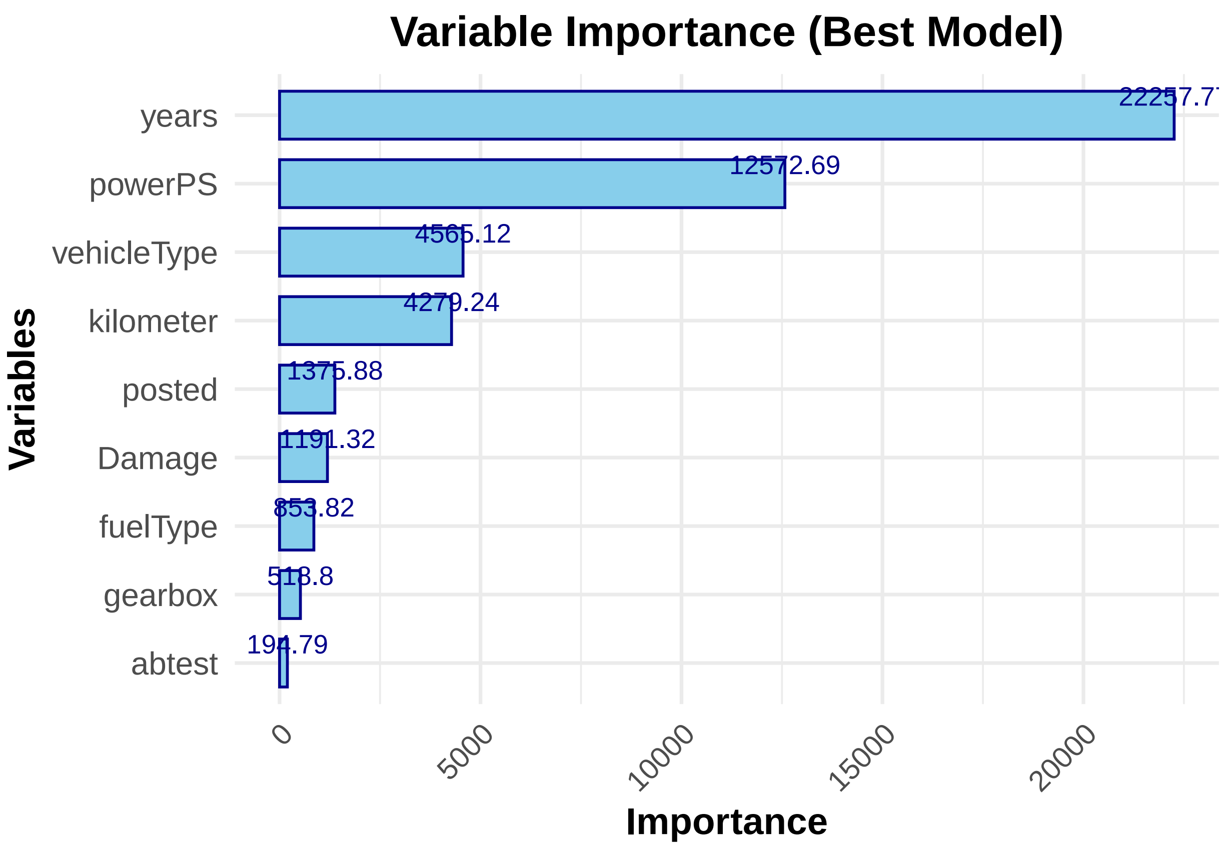
What is the best model's MAE?

```r
grid_rf$MAE[best_index]
```

```
## [1] 0.2916448
```

As expected, Random Forest performed much better than KNN for this task, with a better adaptation to the different variable types. Now, it is time to check the variable importance to get some essential insights.

```r
var_import <- best_model$importance
var_import_df <- data.frame(Variable = rownames(var_import), Importance = var_import[, 1])
```



Similarly to what I had estimated in previous plot analysis, the age of the car has the most significant impact in its pricing. Next is the horsepower followed by the vehicle type and kilometer category.

These are not surprising results, since the aforementioned variables are undoubtedly the first that come to mind when one thinks about buying or selling a used car.

Now, the last thing to do is to save the model, which can be used to predict the price of future used cars.

```r
save(best_model, file = "best_model.RData")
```

Once the model is saved, it is ready to be used, however, remember to transform the predicted log price back into the normal price scale, simply by using the exponent operation.

Conclusion

This model can later be used in potential applications such as a web API to simulate a used cars portal price estimation system. The price estimation will be used to determine whether going through further physical inspections of the car are worth the cost. This would save car selling companies time and money while also reducing the web portal user traffic.