Project Report - Code Part

Xidan Kou & Yunfei Wang

11/10/2021

Part I : Exploratory Data Anaysis

We choose the Auto Pricing dataset

Step 1:

Delete fault and useless information.

Incorrect data/predictors are: “nrOfPictures” contains all 0 “seller” only has the value of “private” “offerType” has only value of “Angebot”

Useless predictors by logic: dateCrawled, name, dateCreated, and lastseen

For the sake for printing, we commented some code which prints too many information.

```
load("Autos.Rdata")
train = train[,c(-1,-2,-3,-4,-17,-18,-20)]
test = test[,c(-1,-2,-3,-4,-17,-18,-20)]
```

Step 2:

take a look at what variables we have, and their type

```
# summary(train)
```

Step 3:

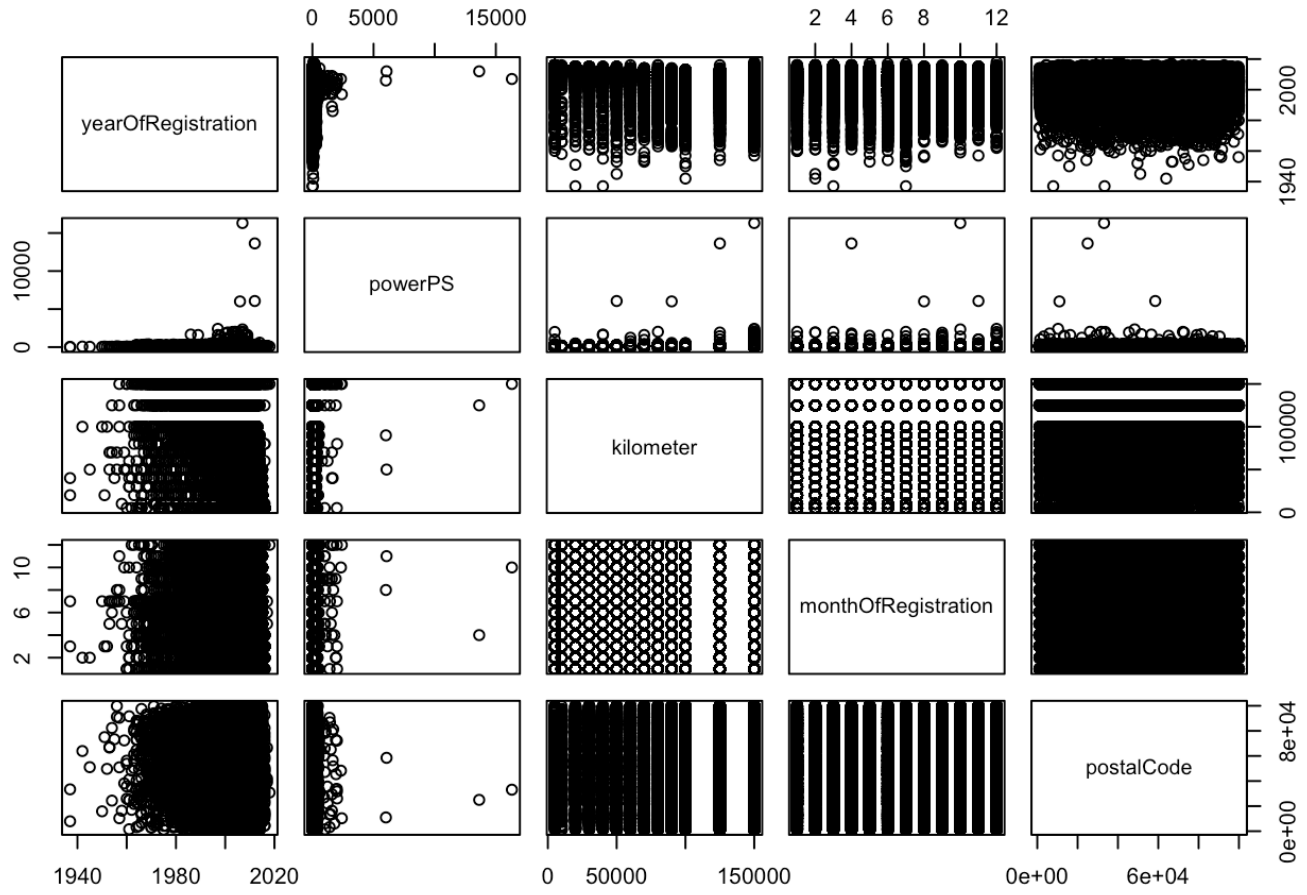
We further discovered several incorrect data. For example, the the range of “yearOfRegistration ranges from 1000 to 9999” So, we specify the range of each variable.

```
IQR = (7200-1150)
min = 1150
max = 7200 + 1.5*IQR
train=train[with(train,price>=min&price<=max),]
test=test[with(test,price>=min&price<=max),]

train=train[with(train, yearOfRegistration >= 1910 & yearOfRegistration<=2021),]
test=test[with(test, yearOfRegistration >= 1910 & yearOfRegistration<=2021),]

train=train[with(train, monthOfRegistration >= 1),]
test=test[with(test, monthOfRegistration >=1),]

train = train[!train$vehicleType == "",]
train = train[!train$abtest == "",]

train = train[!train$gearbox == "",]
train = train[!train$model == "",]

train = train[!train$kilometer == "",]
train = train[!train$fuelType == "",]

train = train[!train$brand == "",]
train = train[!train$postalCode == "",]


train = train[!train$notRepairedDamage== "",]
train = train[!train$fuelType == "",]


test = test[!test$vehicleType == "",]
test = test[!test$abtest == "",]

test = test[!test$gearbox == "",]
test = test[!test$model == "",]

test = test[!test$kilometer == "",]
test = test[!test$fuelType == "",]

test = test[!test$brand == "",]
test = test[!test$postalCode == "",]


test = test[!test$notRepairedDamage== "",]
test = test[!test$fuelType == "",]
```

Step 4

We plot the pairs function to see if there are any high correlation exists between predictors. Here is plot only for numeric predictors.

```
pairs(train[,c(4,6,8,9,13)])
```

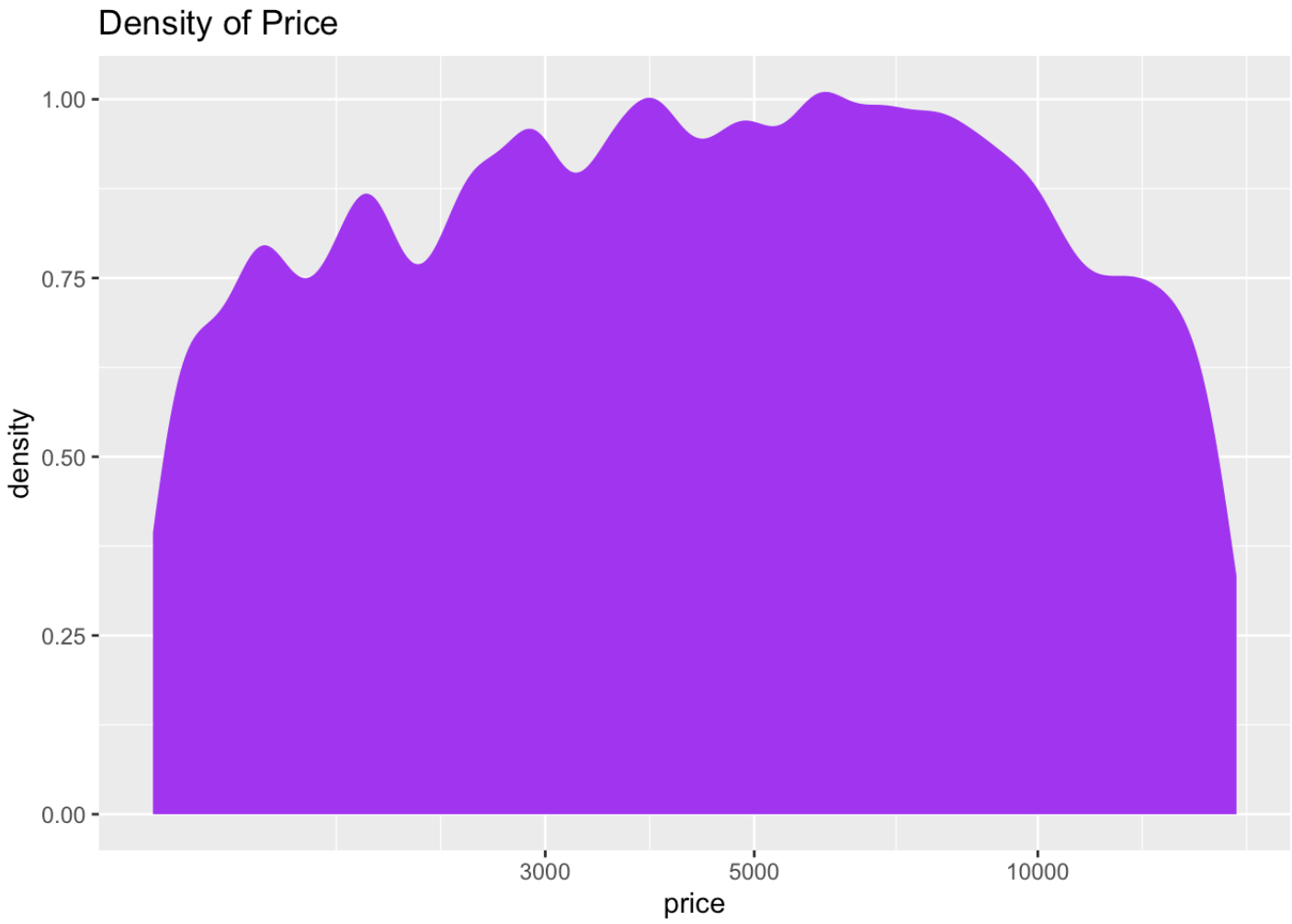


We didn't discover any highly correlated predictors. So, no actions needed at this step.

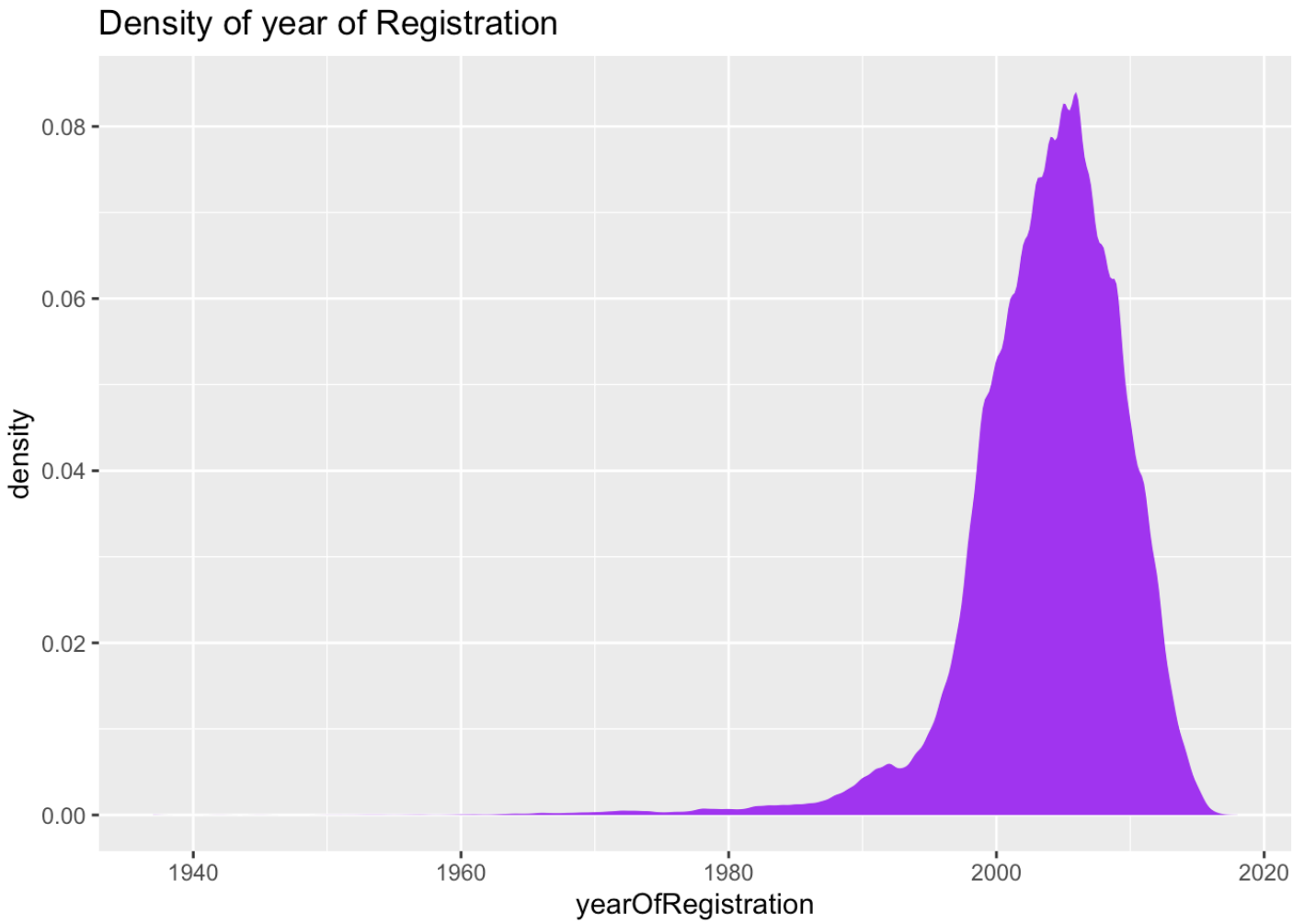
Step 5

We take a look at the distribution of each predictors.

```
library(ggplot2)
ggplot(train,aes(price)) + stat_density(fill = "purple")  + scale_x_log10()+ labs(
title = "Density of Price")
```



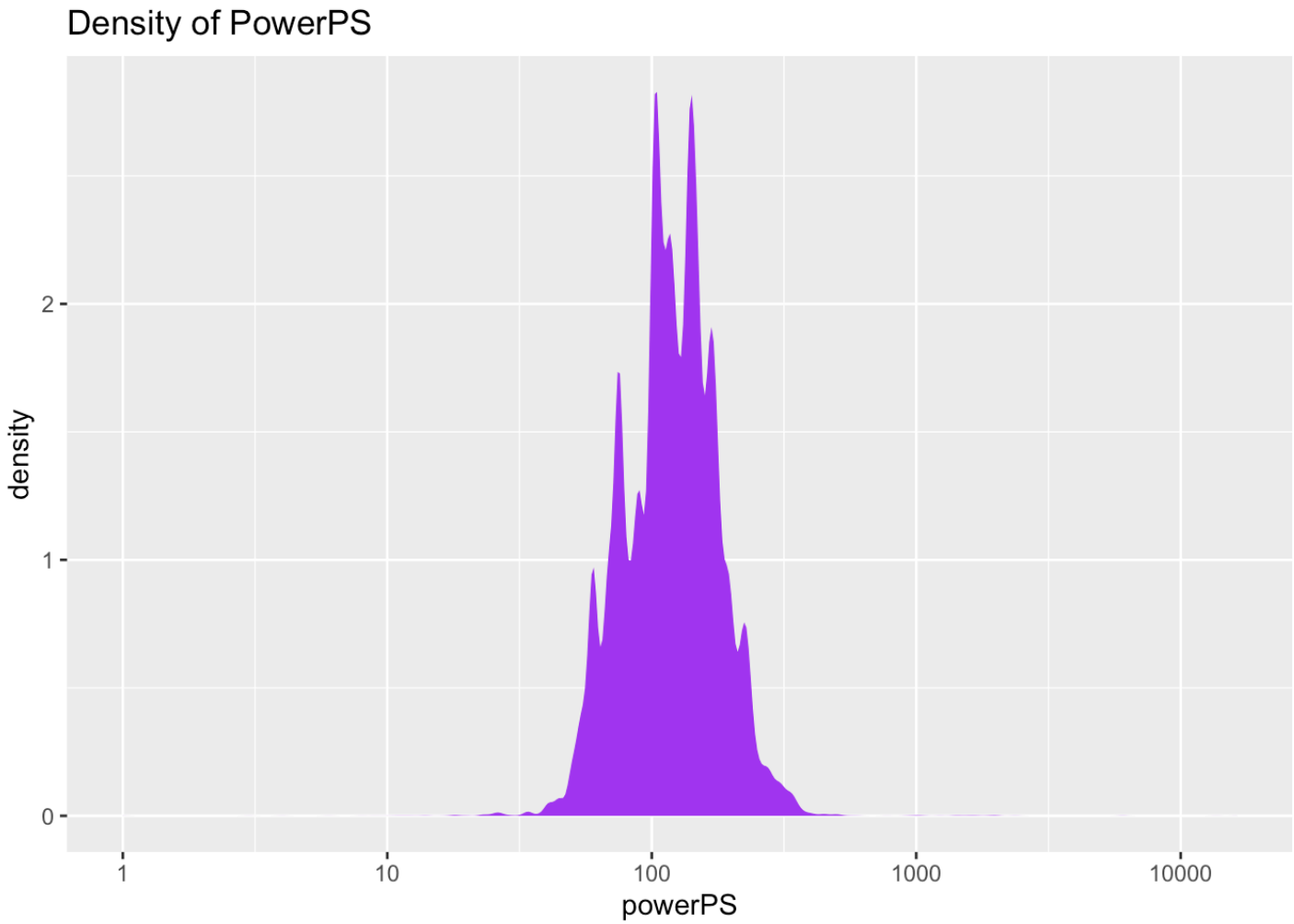

```
ggplot(train,aes(yearOfRegistration)) + stat_density(fill = "purple") + labs(title
= "Density of year of Registration")
```



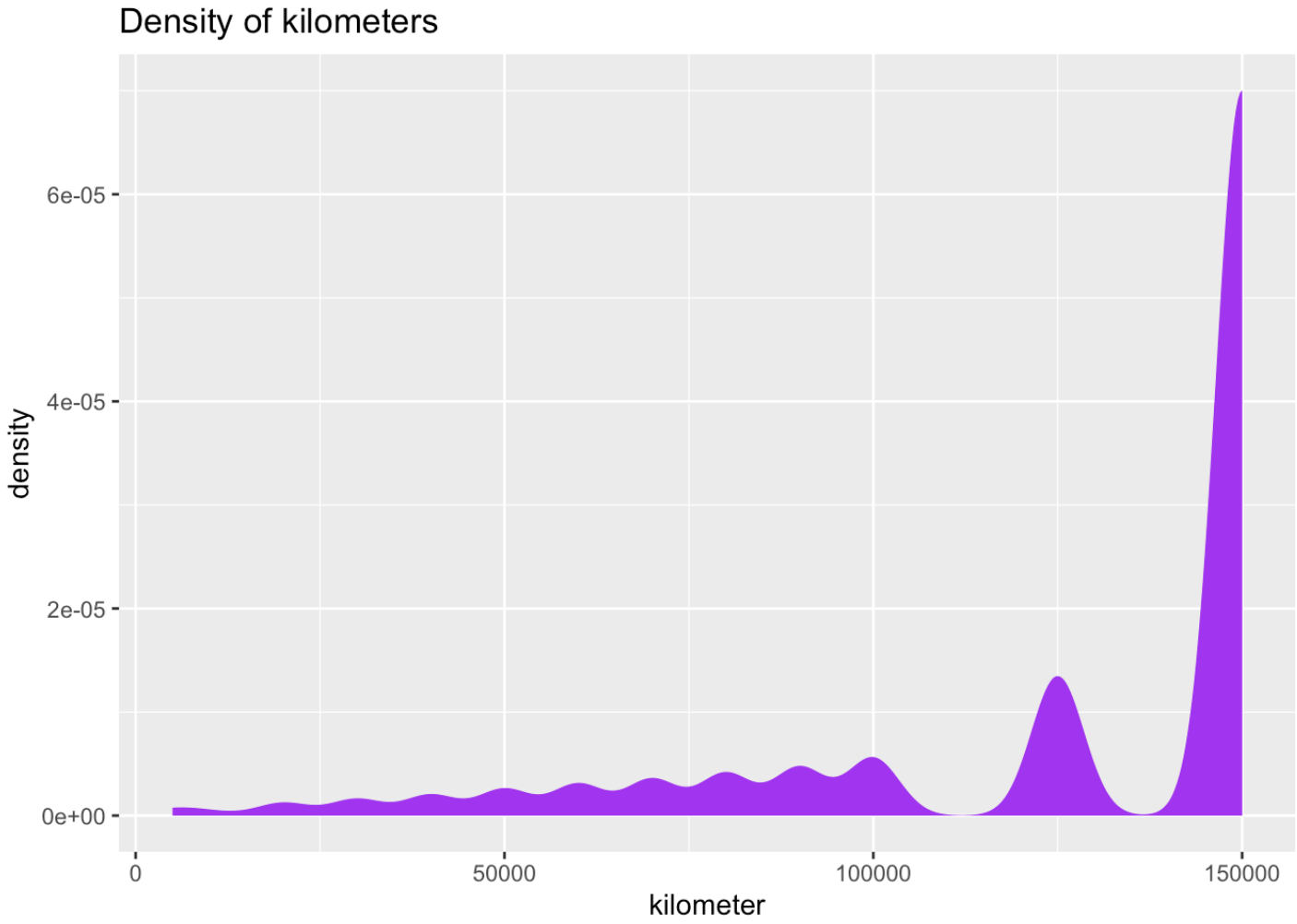

```
ggplot(train,aes(powerPS)) + stat_density(fill = "purple")  + scale_x_log10()+ lab
s(title = "Density of PowerPS")
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

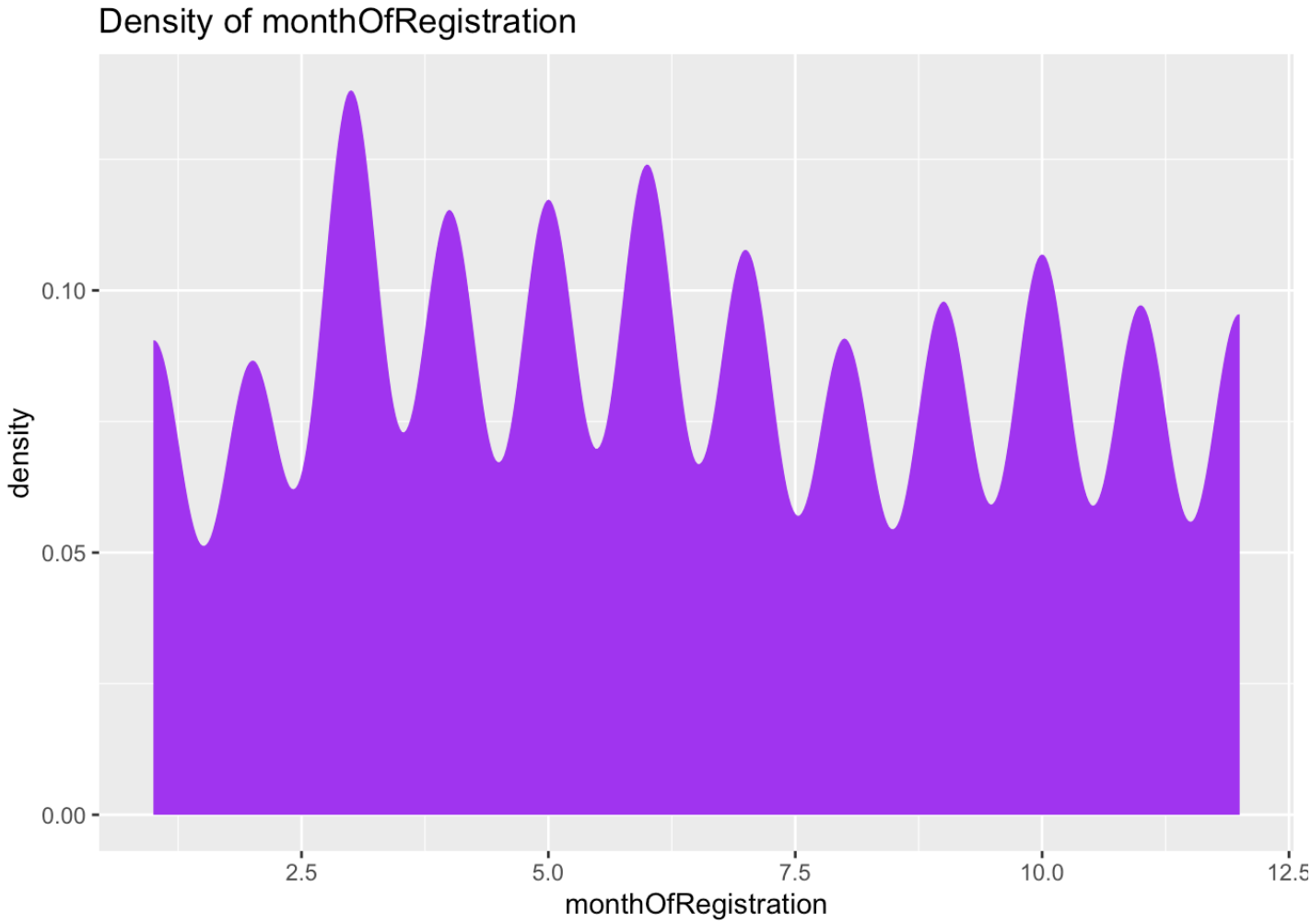
```
## Warning: Removed 1951 rows containing non-finite values (stat_density).
```




```
ggplot(train,aes(kilometer)) + stat_density(fill = "purple") + labs(title = "Densi
ty of kilometers")
```




```
ggplot(train,aes(monthOfRegistration)) + stat_density(fill = "purple") + labs(titl
e = "Density of monthOfRegistration ")
```



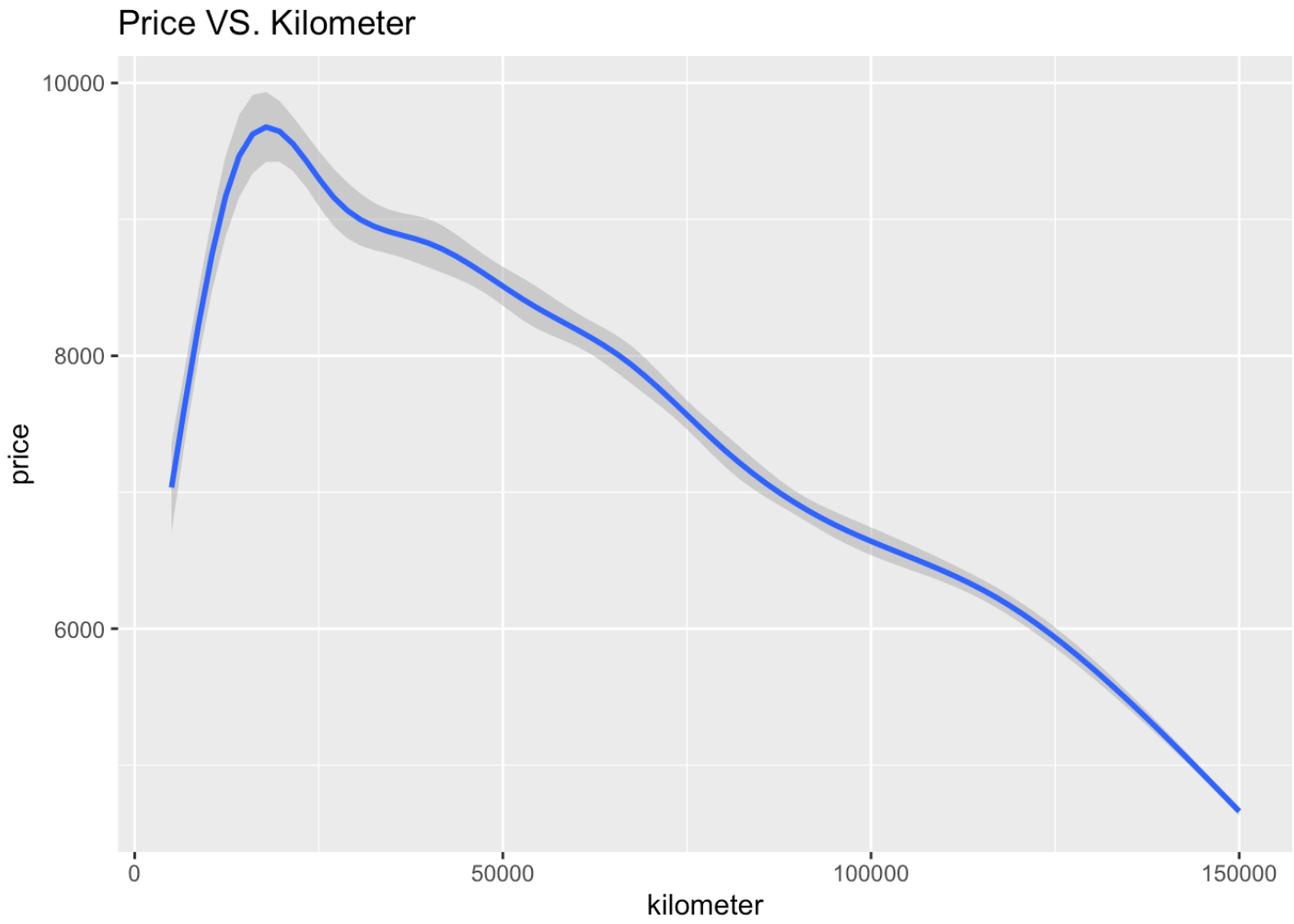

```
# table(train$abtest)
# table(train$vehicleType)
# table(train$gearbox)
# table(train$model)
# table(train$monthOfRegistration)
# table(train$fuelType)
# table(train$brand)
# table(train$notRepairedDamage)
```

Step 6

We want to detect early signs of what variables are likely to be important in predicting the response So we plot the graph of price VS. predictors for each predictors.

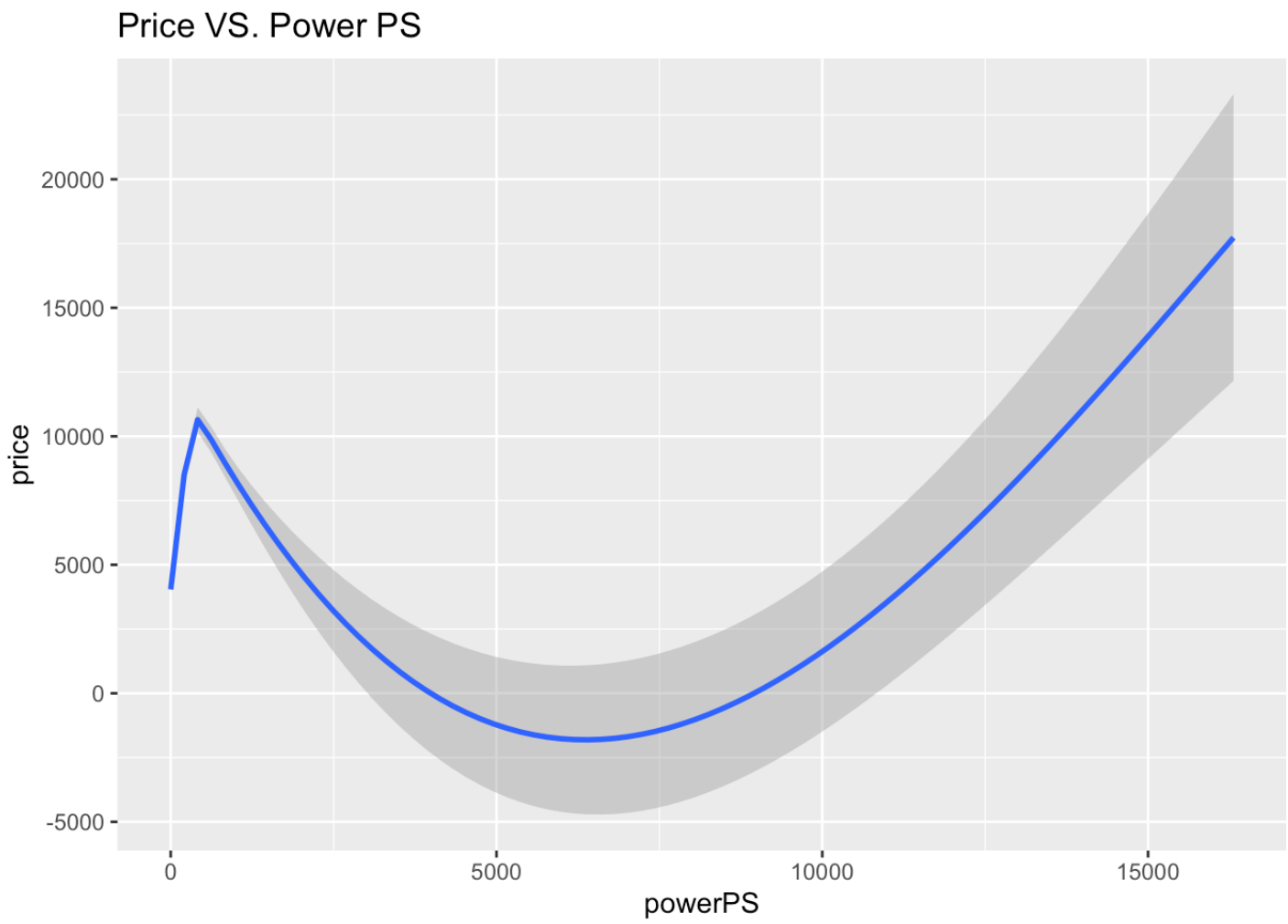
```
ggplot(train, aes(x = kilometer, y = price)) + geom_smooth() + labs(title = "Price VS. Kilometer", xlab = "kilometer", ylab = "Price")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



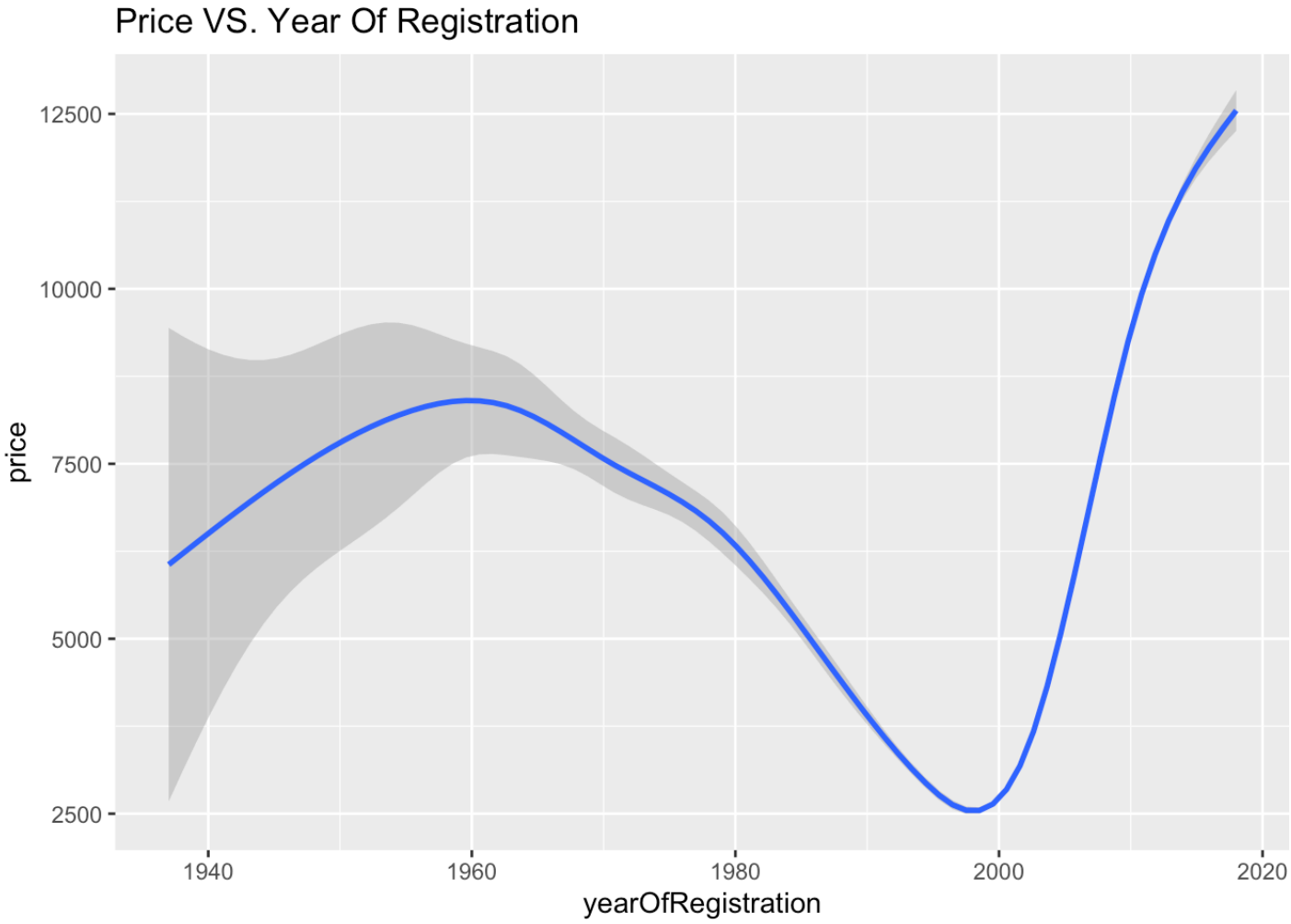

```
ggplot(train, aes(x = powerPS, y = price)) + geom_smooth() + labs(title = "Price V
S. Power PS", xlab = "PowerPS", ylab = "Price")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



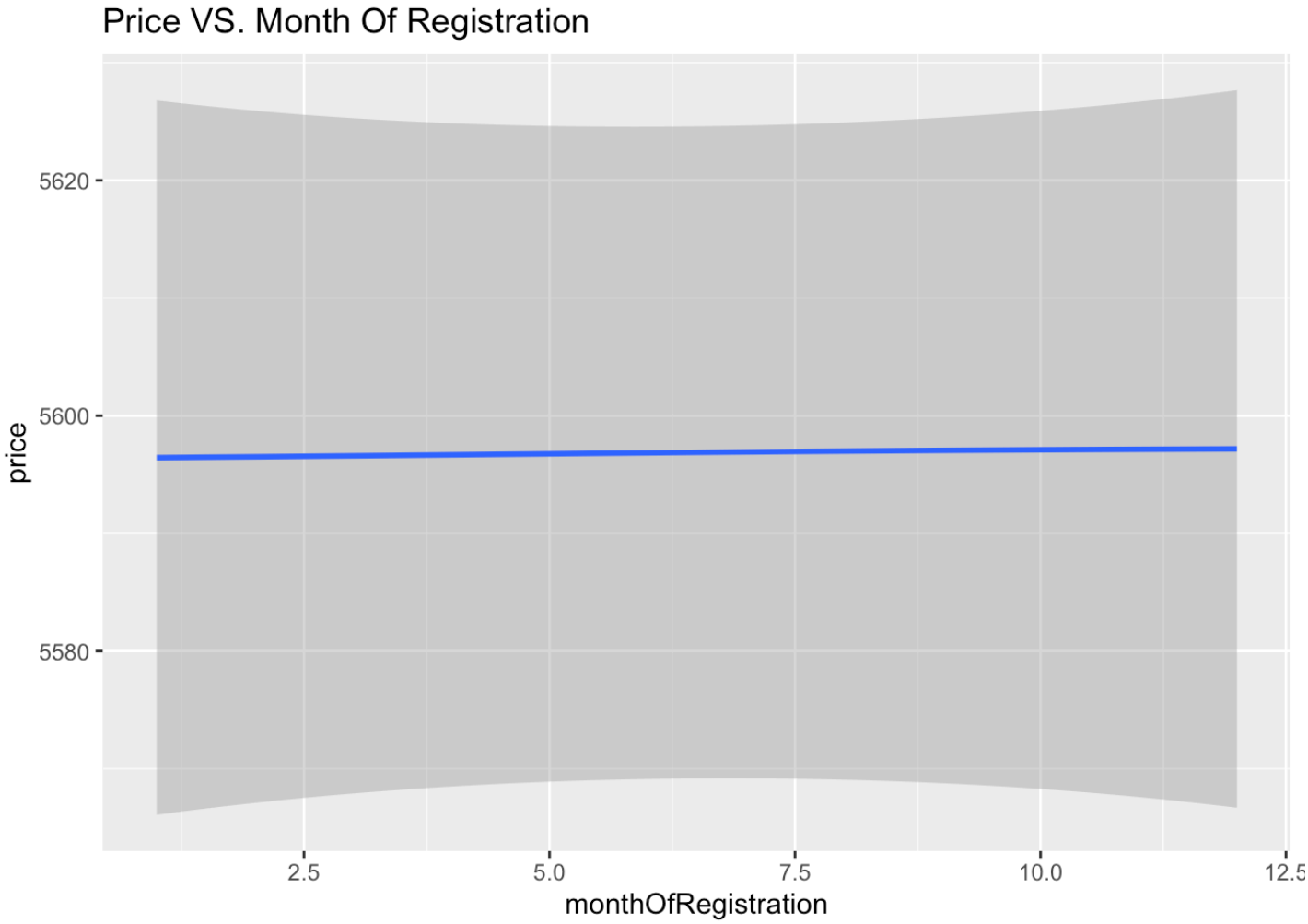

```
ggplot(train, aes(x = yearOfRegistration, y = price)) + geom_smooth() + labs(title
= "Price VS. Year Of Registration", xlab = "Year Of Registration", ylab = "Price")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



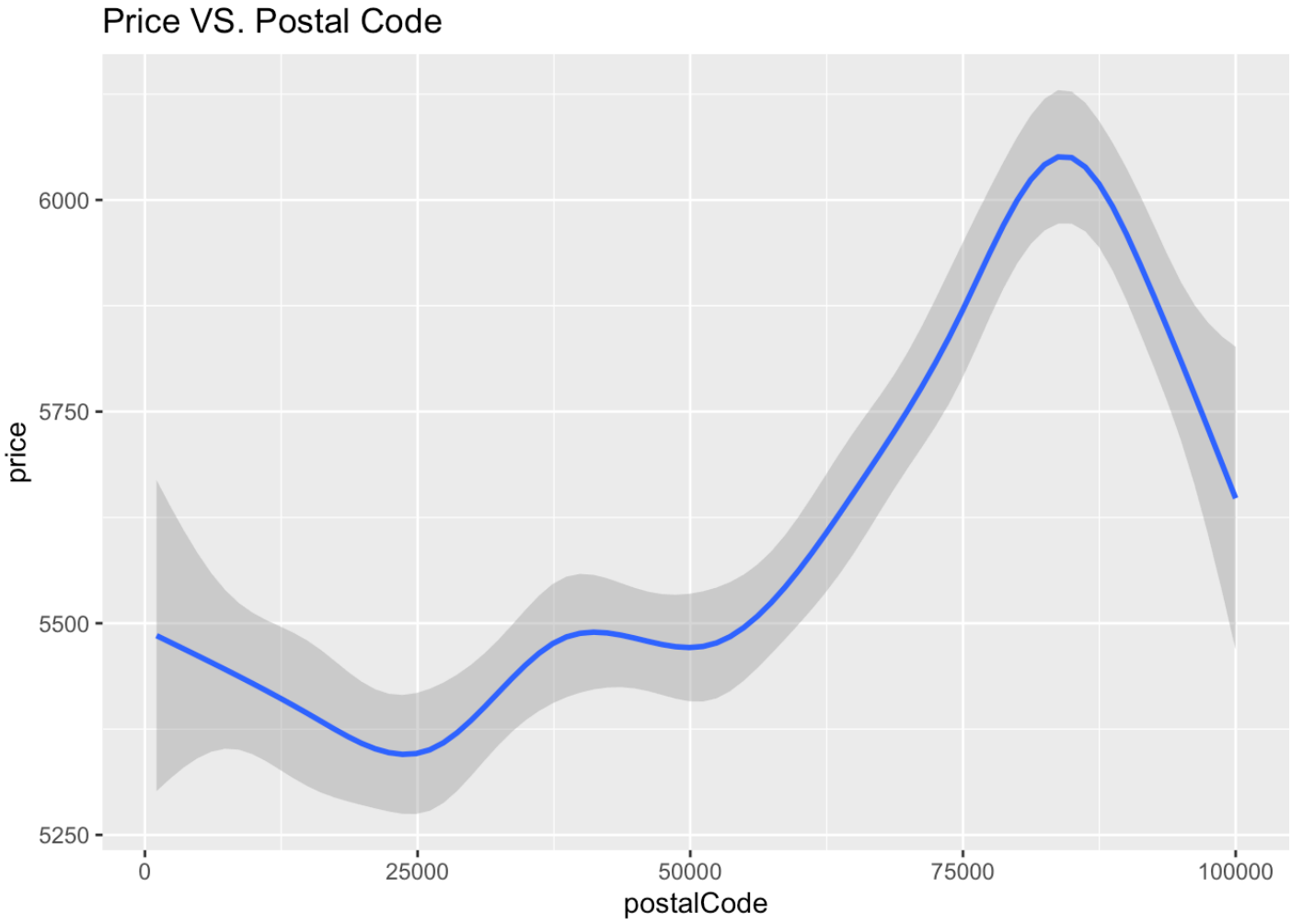

```
ggplot(train, aes(x = monthOfRegistration, y = price)) + geom_smooth() + labs(titl
e = "Price VS. Month Of Registration", xlab = "monthOfRegistration", ylab = "Price
")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

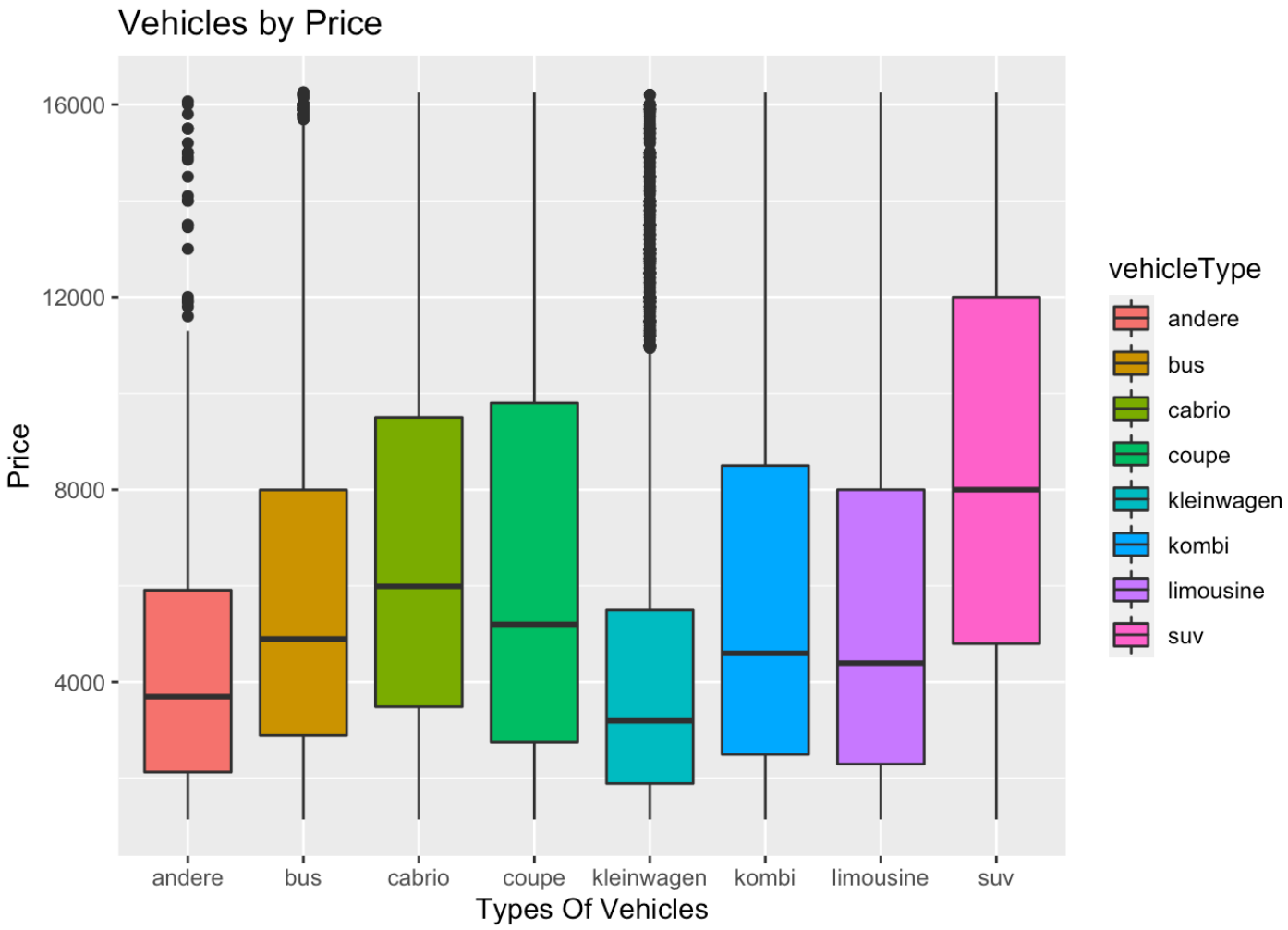



```
ggplot(train, aes(x = postalCode , y = price)) + geom_smooth() + labs(title = "Pri
ce VS. Postal Code", xlab = "Postal Code", ylab = "Price")
```

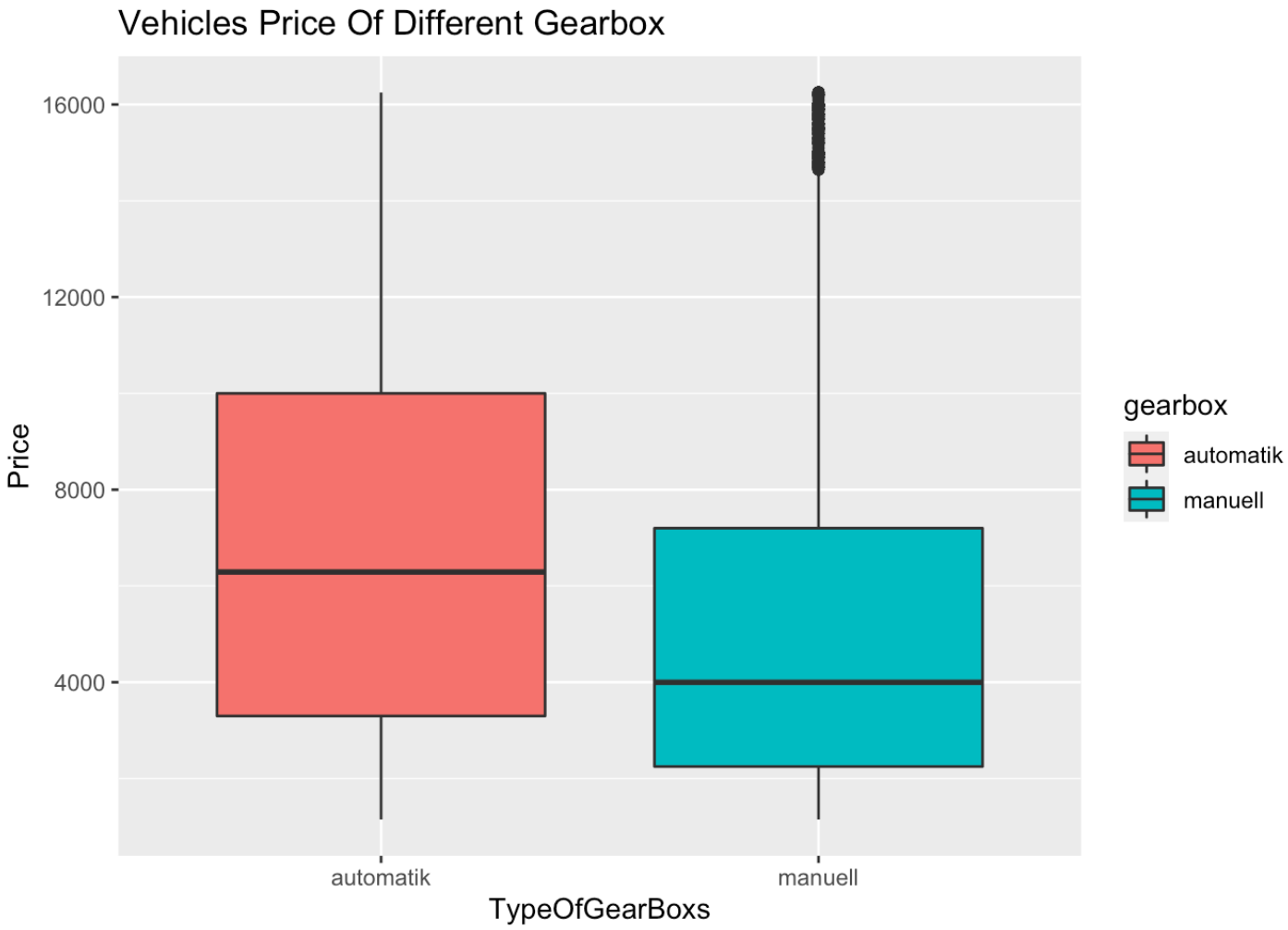
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



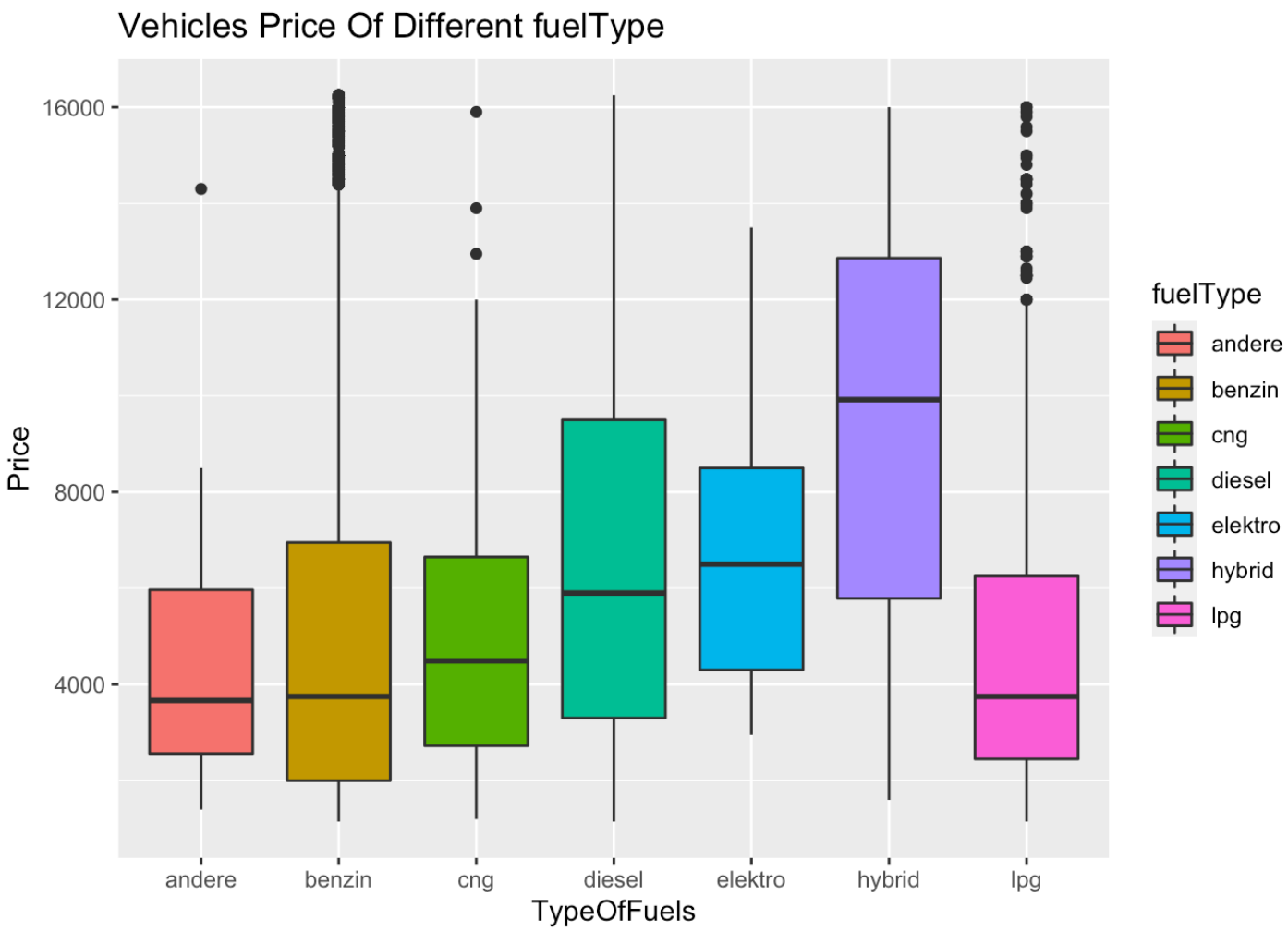

```
#plot of the price distribution of differentt vehicle type
ggplot(train,aes(y = price,x=vehicleType,fill=vehicleType))+
  geom_boxplot()+labs(title="Vehicles by Price")+xlab("Types Of Vehicles")+ylab("P
rice")
```



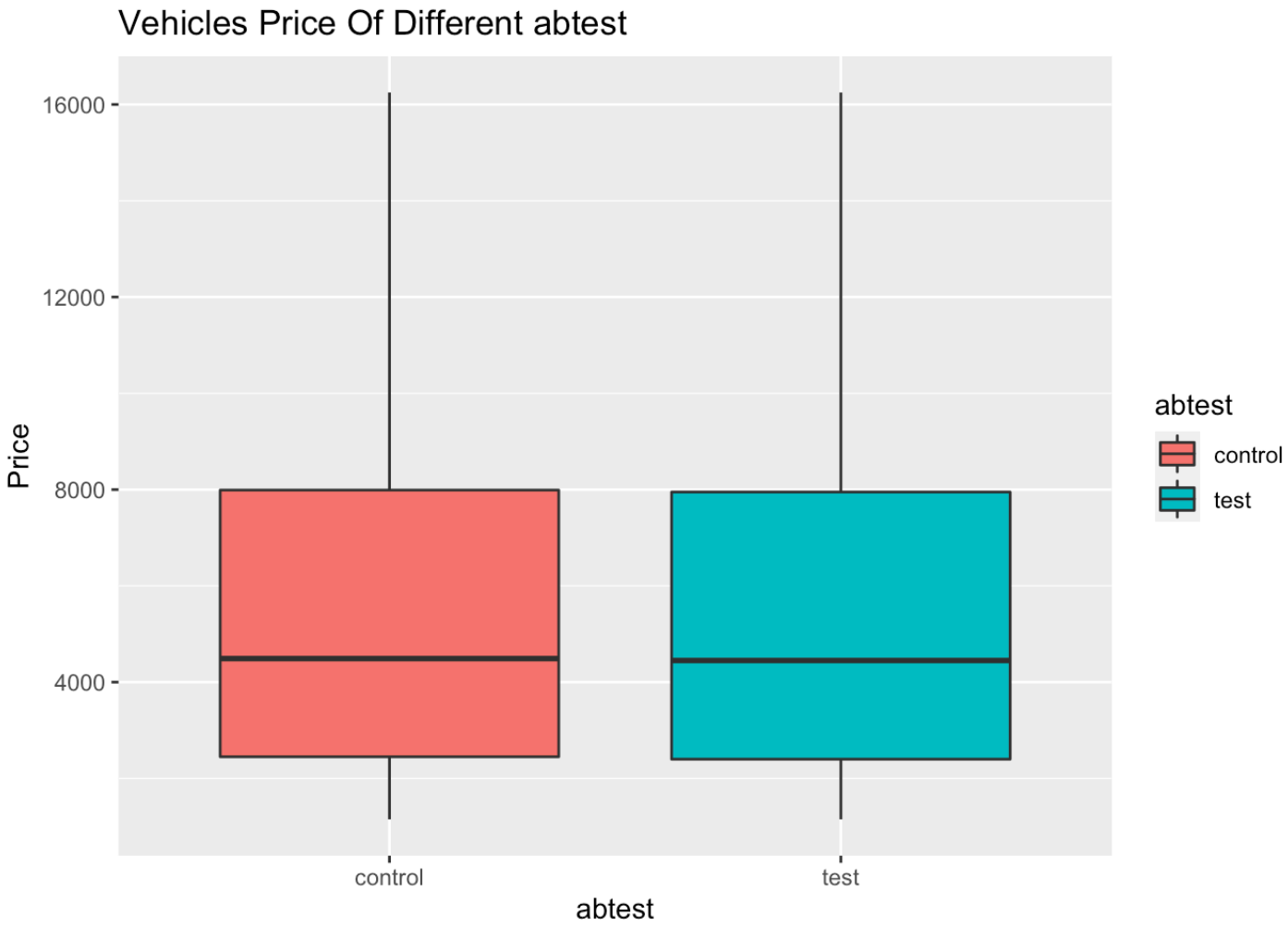

```
# We can see that the mean price of SUV is much higher than other vehicle types an
d the mean price of kleinwagen is the lowest over all the vehicle types, the mean
price of other vehicles are similar.

#plot of the price distribution of different gearbox type
ggplot(train,aes(y = price,x=gearbox,fill=gearbox))+
  geom_boxplot()+labs(title="Vehicles Price Of Different Gearbox")+xlab("TypeOfGea
rBoxs")+ylab("Price")
```



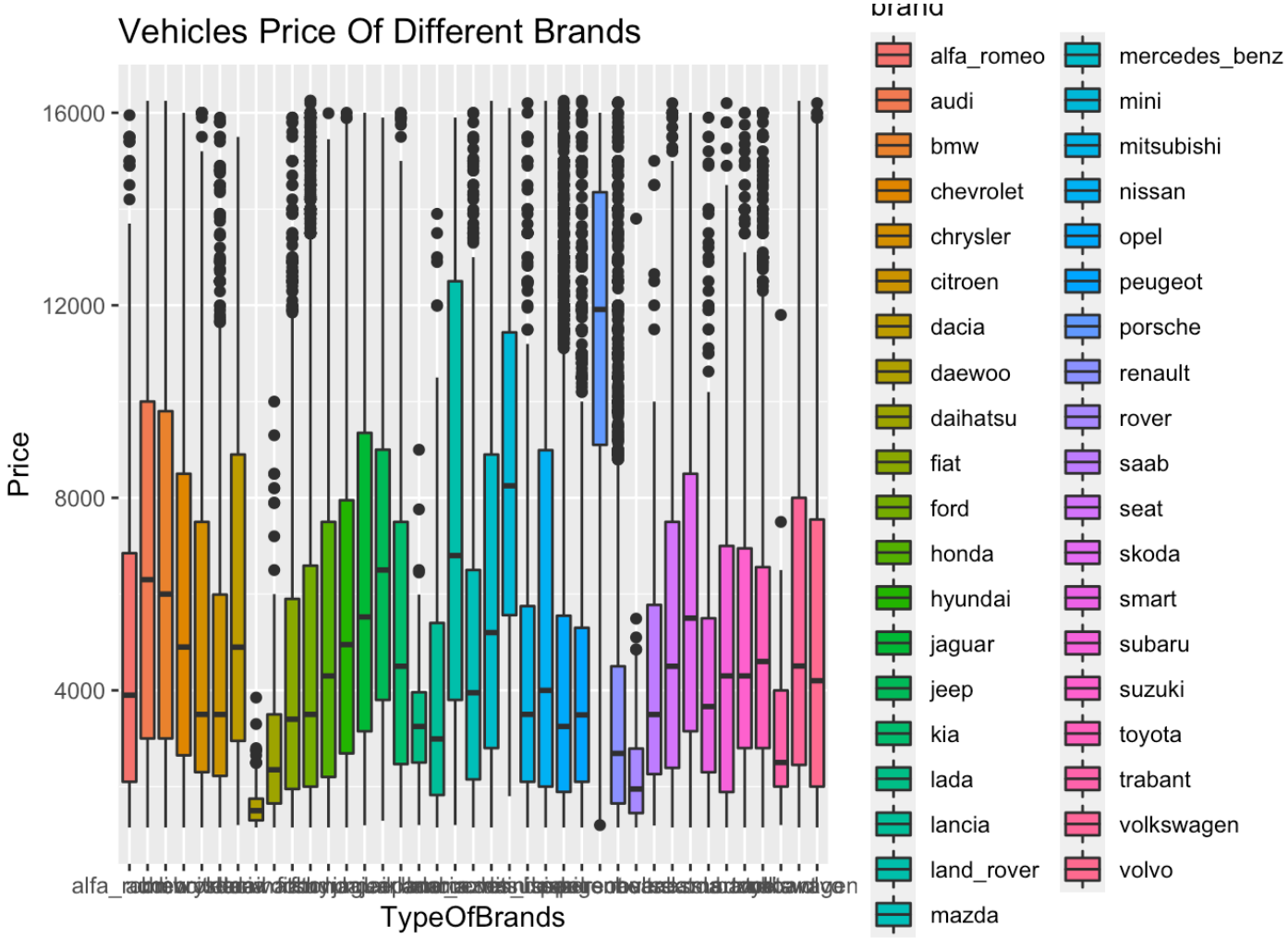

```
#We can see that the mean price of the vehicles have automatik gearbox is higher t
han the vehicles have manuell gear box.

#plot of the price distribution of different fuel type
ggplot(train,aes(y = price,x=fuelType,fill=fuelType))+
  geom_boxplot()+labs(title="Vehicles Price Of Different fuelType")+xlab("TypeOfFu
els")+ylab("Price")
```



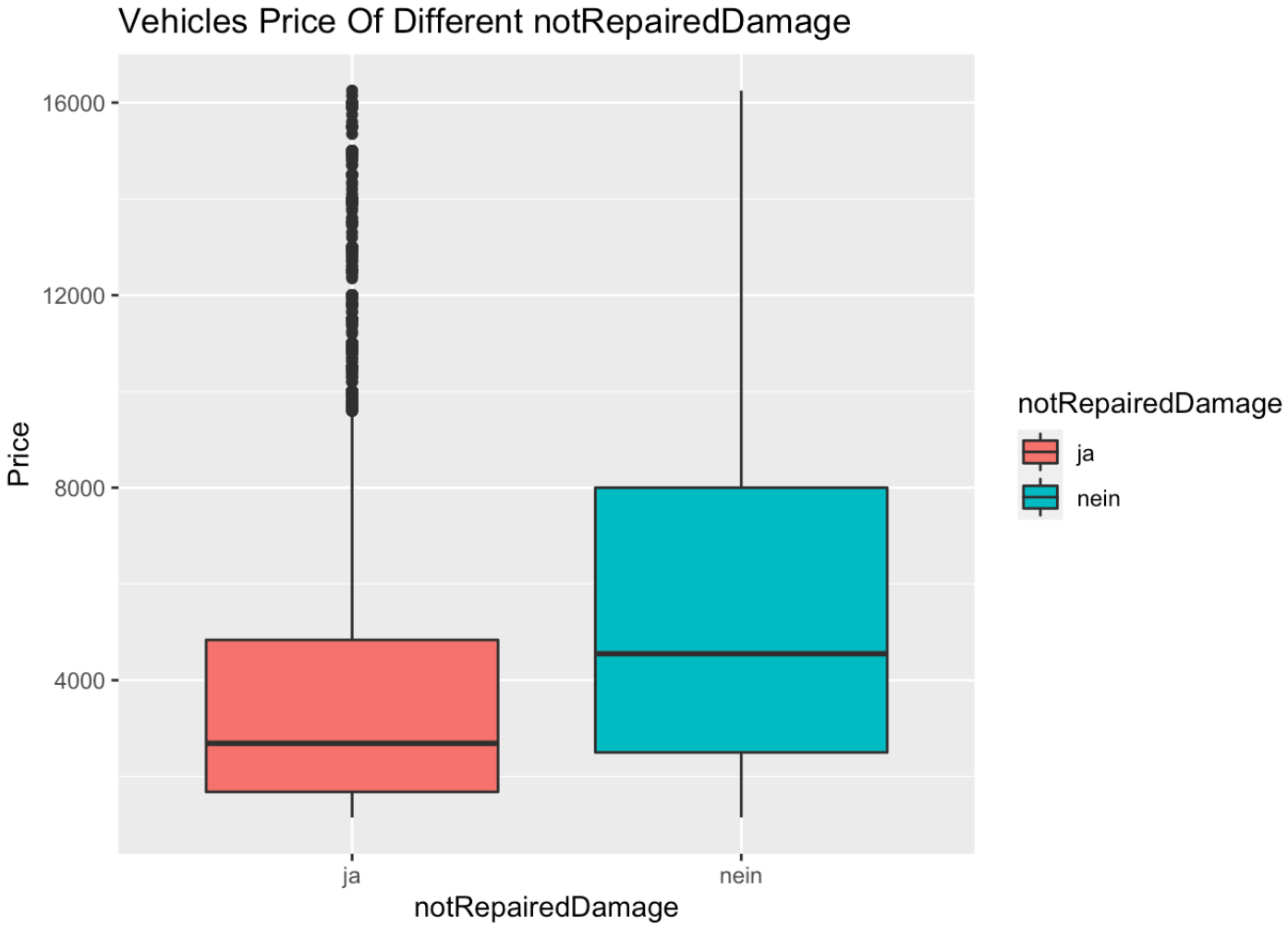

```
#We can see that the price of hybrid fuel type vehicles is much higher than other fuel type vehicles, others mean price seems similar.

ggplot(train,aes(y = price,x=abtest,fill=abtest))+
  geom_boxplot()+labs(title="Vehicles Price Of Different abtest")+xlab("abtest")+ylab("Price")
```




```
#We can see that the price of hybrid fuel type vehicles is much higher than other
fuel type vehicles, others mean price seems similar.

#plot of the price distribution of different brand type
ggplot(train,aes(y = price,x=brand,fill=brand))+
  geom_boxplot()+labs(title="Vehicles Price Of Different Brands")+xlab("TypeOfBran
ds")+ylab("Price")
```


Vehicles Price Of Different Brands
Price
16000
12000
8000
4000
TypeOfBrands
brand
alfa_romeo
audi
bmw
chevrolet
chrysler
citroen
dacia
daewoo
daihatsu
fiat
ford
honda
hyundai
jaguar
jeep
kia
lada
lancia
land_rover
mazda
mercedes_benz
mini
mitsubishi
nissan
opel
peugeot
porsche
renault
rover
saab
seat
skoda
smart
subaru
suzuki
toyota
trabant
volkswagen
volvo

```
#There are so many different brand here and the plot is hard to see, so I am going
to use numeric result.
#the mean price of different brand

# with(train,by(price,brand,mean))

#We can see that the three highest mean price brand are porsche,  land_rover and s
onstige_autos. The three lowest mean price band are daewoo, rover and daihatsu.

#the mean price of different model
# model = with(train,by(price,model,mean))
#We can see that the highest mean price model is 'glk' and the lowest mean price m
odel is 'lanos'.

#The mean price of different kilometer of vehicles

# with(train,by(price,kilometer,mean))

#We can see that the highest mean price is the vehicles with 10000 kilometers and
the lowest mean price is the vehicles with 150000 kilometers.


ggplot(train,aes(y = price,x=notRepairedDamage,fill=notRepairedDamage))+
  geom_boxplot()+labs(title="Vehicles Price Of Different notRepairedDamage")+xlab(
"notRepairedDamage")+ylab("Price")
```



Since the plot of monthOfRegistration VS. Price showed a strait line, there are no relationships between them. So, we would also consider to delete this variable. Postal Code would nonsense to predict price, so we delete this predictor as well.

```
train = train[,c(-2,-9,-13)]
test = test[,c(-2,-9,-13)]
```

Besides, here is what we discovered:

“Kilometer” are likely be the important predictor of the price, which beyound the certain value of kilometer, price starts to decrease dramatically.

The graph of “Price VS. powerPs” showed the upward U shape, where there is the trend that before and after certain value of Power PS, the price will decrease and increase.

“Price” kind of fluctuated with the change of registration year, it shows that the price reach its lowest when the car is registered at year 2000. For the part after year 2000, We can tell that the price would be higher for recent or newer cars,and lower for old cars.

Step 7

Since the data contained both categorical variables and continuous variables, We can try Classification methods for Logistic Regression, and also Smoothing Splines Regression.

Part II: Model Analysis

Model 1 : Ridge Regression

Reasons for using Ridge Regression Ridge Regression solves the problem of over fitting , as just regular squared error regression fails to recognize the less important features and uses all of them, leading to over fitting. Ridge regression adds a slight bias, to fit the model according to the true values of the data

```
#Ridge Regression
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-2
```

```
#define x and y
x_var <-  data.matrix(train[,c(-1)])
y_var <- train$price
#use cross-validation to choose the tuning parameter λ
ridge = cv.glmnet(x_var,y_var, alpha=0)
lambda.ridge = ridge$lambda.min
lambda.ridge
```

```
## [1] 298.8788
```

```
#fit the ridge model on the training set
ridge.mod=glmnet(x_var,y_var,alpha=0,lambda=lambda.ridge)
newX.test <- data.matrix(test[,c(-1)])
newX.train <- data.matrix(train[,c(-1)])
#fit the ridge model on the training set
ridge.mod=glmnet(x_var,y_var,alpha=0,lambda=lambda.ridge)
#the test MSE associated with this value of λ
#get prediction for a test set
ridge.pred.test=predict(ridge.mod,s=lambda.ridge,newx=newX.test)
#get prediction for a train set
ridge.pred.train=predict(ridge.mod,s=lambda.ridge,newx=newX.train)
#MSE for training set
MSE.ridge.train=mean((ridge.pred.train-train$price)^2)
MSE.ridge.train
```

```
## [1] 9144232
```

```
coef(ridge.mod)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)        -4.502995e+05
## vehicleType        -2.207469e+01
## yearOfRegistration  2.282249e+02
## gearbox            -1.385556e+03
## powerPS             6.675788e+00
## model               5.533975e-01
## kilometer          -2.876902e-02
## fuelType            4.656549e+02
## brand              -2.319420e+01
## notRepairedDamage   1.578579e+03
```

```
#MSE for test set
MSE.ridge.test=mean((ridge.pred.test-test$price)^2)
MSE.ridge.test
```

```
## [1] 9059573
```

Model 2 : LASSO

Reasons for using LASSO Ii produces simpler and more interpretable models that incorporate only a reduced set of the predictors.

```
#Lasso Regression
#define x and y
#use cross-validation to choose the tuning parameter λ
lasso = cv.glmnet(x_var,y_var, alpha=1)
lambda.lasso = lasso$lambda.min
lambda.lasso
```

```
## [1] 16.32559
```

```
#fit the LASSO on the training set
lasso.mod=glmnet(x_var,y_var,alpha=1,lambda=lambda.lasso, thresh =1e-12)
#get prediction for a test set
lasso.pred.test=predict(lasso.mod,s=lambda.lasso,newx=newX.test)
#get prediction for a train set
lasso.pred.train=predict(lasso.mod,s=lambda.lasso,newx=newX.train)
#get the MSE on test set
MSE.lasso.test=mean((lasso.pred.test-test$price)^2)
MSE.lasso.test
```

```
## [1] 9014649
```

```
#get the MSE on train set
MSE.lasso.train=mean((lasso.pred.train-train$price)^2)
MSE.lasso.train
```

```
## [1] 9126523
```

```
coef(lasso.mod)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)        -4.724682e+05
## vehicleType        -1.762425e+01
## yearOfRegistration  2.393371e+02
## gearbox            -1.439491e+03
## powerPS             7.004123e+00
## model               2.739827e-01
## kilometer          -3.025708e-02
## fuelType            4.854645e+02
## brand              -2.269282e+01
## notRepairedDamage   1.619806e+03
```

Data Transformation - train data

```
# Transform Vehicle type to three types
train$VT.suv = ifelse(train$vehicleType == 'suv'  ,1,0)
train$VT.andere.klein = ifelse(train$vehicleType == 'andere'|train$vehicleType ==
'kleinwagen' ,1,0)
train$VT.others = ifelse(train$VT.suv == 0 & train$VT.andere.klein == 0,1,0)

# Transform model to three types based on the range
# 1. model with min < price < 5439
# 2. model with 5440 < price < 9619
```

```
# 3. model with 9619 < price < max
# model = with(train,by(price,model,mean))
# sort(model)

train$model.low = ifelse(train$model == 'lanos'|train$model == 'lybra'|train$model
== 'samara'|train$model == '145'|train$model == 'seicento'|train$model == 'cordoba
'|train$model == 'move'|train$model == 'arosa'|train$model == 'carisma'|train$mode
l == 'r19'|train$model == 'kalos'|train$model == 'nubira'|train$model == 'lupo'|tr
ain$model == 'omega'|train$model == 'primera'|train$model == 'cuore'|train$model =
= 'galant'|train$model == 'stilo'|train$model == 'almera'|train$model == 'v40'|tra
in$model == 'matiz'|train$model == 'escort'|train$model == 'getz'|train$model == '
156'|train$model == 'bora'|train$model == '80'|train$model == '90'|train$model ==
'agila'|train$model == 'carnival'|train$model == 'sirion'|train$model == 'twingo'|
train$model == '147'|train$model == 'vectra'|train$model == 'ka'|train$model == 'c
2'|train$model == 'forfour'|train$model == 'punto'|train$model == 'justy'|train$mo
del == 'corolla'|train$model == 'kangoo'|train$model == 'scenic'|train$model == 'm
icra'|train$model == 'clio'|train$model == '1_reihe'|train$model == '100'|train$mo
del == 'voyager'|train$model == 'fox'|train$model == 'kalina'|train$model == 'ptcr
uiser'|train$model == '850'|train$model == 'ypsilon'|train$model == '601'|train$mo
del == 'fusion'|train$model == 'panda'|train$model == 'combo'|train$model == 'modu
s'|train$model == 'laguna'|train$model == 'toledo'|train$model == 'espace'|train$m
odel == '9000'|train$model == 'corsa'|train$model == 'legacy'|train$model == 'niva
'|train$model == 'picanto'|train$model == 'doblo'|train$model == 'v_klasse'|train$
model == 'colt'|train$model == 'musa'|train$model == 'logan'|train$model == 'a_kla
sse'|train$model == 'c1'|train$model == 'kappa'|train$model == 'croma'|train$model
== 'tigra'|train$model == '2_reihe'|train$model == 'calibra'|train$model == 'c3'|t
rain$model == 'mondeo'|train$model == 'a2'|train$model == 'signum'|train$model ==
'fortwo'|train$model == 'aygo'|train$model == 'zafira'|train$model == '3_reihe'|tr
ain$model == 'galaxy'|train$model == 'polo'|train$model == 'megane'|train$model ==
'kadett'|train$model == 'yaris'|train$model == '4_reihe'|train$model == 'bravo'|tr
ain$model == 'lancer'|train$model == 'meriva'|train$model == 'astra'|train$model =
= 'focus'|train$model == 'fabia'|train$model == 'alhambra'|train$model == 'sandero
'|train$model == 'berlingo'|train$model == '6_reihe'|train$model == 'ducato'|train
$model == 's_type'|train$model == '900'|train$model == 'materia'|train$model == 'f
iesta'|train$model == 'c5'|train$model == 'i3'|train$model == 'aveo'|train$model =
= 'v70'|train$model == 'note'|train$model == 'transit'|train$model == 'ibiza'|trai
n$model == 'forester'|train$model == 'jazz'|train$model == 'civic'|train$model ==
'cherokee'|train$model == 'roadster'|train$model == 'spark'|train$model == 'mx_rei
he'|train$model == 'sharan',1,0)

train$model.high = ifelse(train$model == 'captiva'|train$model == 'discovery'|trai
n$model == 'touareg'|train$model == 'duster'|train$model == 'mustang'|train$model
== '300c'|train$model == 'navara'|train$model == 'amarok'|train$model == 'eos'|tra
in$model == 'g_klasse'|train$model == 'clubman'|train$model == 'xc_reihe'|train$mo
del == 'crossfire'|train$model == 'insignia'|train$model == 'x_reihe'|train$model
== 'scirocco'|train$model == 'lodgy'|train$model == 'v60'|train$model == 'viano'|t
rain$model == 'cx_reihe'|train$model == 'm_reihe'|train$model == 'exeo'|train$mode
l == '6er'|train$model == 'juke'|train$model == 'sl'|train$model == 'qashqai'|trai
n$model == 'cayenne'|train$model == 'boxster'|train$model == 'defender'|train$mode
l == 'yeti'|train$model == 'b_max'|train$model == 'tiguan'|train$model == 'kuga'|t
```

```
rain$model == '911'|train$model == 'a1'|train$model == 'cc'|train$model == 'a5'|tr
ain$model == 'range_rover_sport'|train$model == 'q5'|train$model == 'q7',1,0)

train$model.medium = ifelse(train$model.low == 0 & train$model.high == 0,1,0)

# Transform Fuel type to three types
train$FT.hybrid = ifelse(train$fuelType == 'hybrid',1,0)
train$FT.diesel.elektro = ifelse(train$fuelType == 'diesel' & train$fuelType == 'e
lektro' ,1,0)
train$FT.others = ifelse(train$FT.hybrid == 0 & train$FT.diesel.elektro == 0 ,1,0)

# Transform brand type to three types

train$brand.daewoo.rover = ifelse(train$brand == 'daewoo'|train$brand == 'rover' ,
1,0)
train$brand.porsche = ifelse(train$brand == 'porsche',1,0)
train$brand.others = ifelse(train$brand.daewoo.rover == 0 &train$brand.porsche ==
0 ,1,0)

# save new train data
train_trans = train[,c(-2,-6,-8,-9)]
train_trans = train_trans[,c(-9,-10,-14,-15,-16)]

train_trans$brand.others = as.factor(train_trans$brand.others)
train_trans$brand.porsche = as.factor(train_trans$brand.porsche)
train_trans$FT.hybrid = as.factor(train_trans$FT.hybrid)
train_trans$model.medium = as.factor(train_trans$model.medium)
train_trans$model.high = as.factor(train_trans$model.high)
train_trans$VT.andere.klein = as.factor(train_trans$VT.andere.klein)
train_trans$VT.suv = as.factor(train_trans$VT.suv)
train_trans$notRepairedDamage = as.factor(train_trans$notRepairedDamage)
train_trans$gearbox = as.factor(train_trans$gearbox)
```

Data Transformation - test data

```
# Transform Vehicle type to three types
test$VT.suv = ifelse(test$vehicleType == 'suv'  ,1,0)
test$VT.andere.klein = ifelse(test$vehicleType == 'andere'|test$vehicleType == 'kl
einwagen' ,1,0)
test$VT.others = ifelse(test$VT.suv == 0 & test$VT.andere.klein == 0,1,0)


# Transform model to three types based on the range
# 1. model with min < price < 5439
# 2. model with 5440 < price < 9619
# 3. model with 9619 < price < max
# model = with(train,by(price,model,mean))
# sort(model)
```

```
test$model.low = ifelse(test$model == 'lanos'|test$model == 'lybra'|test$model ==
'samara'|test$model == '145'|test$model == 'seicento'|test$model == 'cordoba'|test
$model == 'move'|test$model == 'arosa'|test$model == 'carisma'|test$model == 'r19'
|test$model == 'kalos'|test$model == 'nubira'|test$model == 'lupo'|test$model == '
omega'|test$model == 'primera'|test$model == 'cuore'|test$model == 'galant'|test$m
odel == 'stilo'|test$model == 'almera'|test$model == 'v40'|test$model == 'matiz'|t
est$model == 'escort'|test$model == 'getz'|test$model == '156'|test$model == 'bora
'|test$model == '80'|test$model == '90'|test$model == 'agila'|test$model == 'carni
val'|test$model == 'sirion'|test$model == 'twingo'|test$model == '147'|test$model
== 'vectra'|test$model == 'ka'|test$model == 'c2'|test$model == 'forfour'|test$mod
el == 'punto'|test$model == 'justy'|test$model == 'corolla'|test$model == 'kangoo'
|test$model == 'scenic'|test$model == 'micra'|test$model == 'clio'|test$model == '
1_reihe'|test$model == '100'|test$model == 'voyager'|test$model == 'fox'|test$mode
l == 'kalina'|test$model == 'ptcruiser'|test$model == '850'|test$model == 'ypsilon
'|test$model == '601'|test$model == 'fusion'|test$model == 'panda'|test$model == '
combo'|test$model == 'modus'|test$model == 'laguna'|test$model == 'toledo'|test$mo
del == 'espace'|test$model == '9000'|test$model == 'corsa'|test$model == 'legacy'|
test$model == 'niva'|test$model == 'picanto'|test$model == 'doblo'|test$model == '
v_klasse'|test$model == 'colt'|test$model == 'musa'|test$model == 'logan'|test$mod
el == 'a_klasse'||test$model == 'c1'|test$model == 'kappa'|test$model == 'croma'|t
est$model == 'tigra'|test$model == '2_reihe'|test$model == 'calibra'|test$model ==
'c3'|test$model == 'mondeo'|test$model == 'a2'|test$model == 'signum'|test$model =
= 'fortwo'|test$model == 'aygo'|test$model == 'zafira'|test$model == '3_reihe'|tes
t$model == 'galaxy'|test$model == 'polo'||test$model == 'megane'|test$model == 'ka
dett'|test$model == 'yaris'|test$model == '4_reihe'|test$model == 'bravo'|test$mod
el == 'lancer'|test$model == 'meriva'|test$model == 'astra'|test$model == 'focus'|
test$model == 'fabia'||test$model == 'alhambra'|test$model == 'sandero'|test$model
== 'berlingo'|test$model == '6_reihe'|test$model == 'ducato'|test$model == 's_type
'|test$model == '900'|test$model == 'materia'|test$model == 'fiesta'|test$model ==
'c5'|test$model == 'i3'|test$model == 'aveo'|test$model == 'v70'|test$model == 'no
te'|test$model == 'transit'|test$model == 'ibiza'|test$model == 'forester'|test$mo
del == 'jazz'|test$model == 'civic'|test$model == 'cherokee'|test$model == 'roadst
er'|test$model == 'spark'|test$model == 'mx_reihe'|test$model == 'sharan',1,0)

test$model.high = ifelse(test$model == 'captiva'|test$model == 'discovery'|test$mo
del == 'touareg'|test$model == 'duster'|test$model == 'mustang'|test$model == '300
c'|test$model == 'navara'|test$model == 'amarok'|test$model == 'eos'|test$model ==
'g_klasse'|test$model == 'clubman'|test$model == 'xc_reihe'|test$model == 'crossfi
re'|test$model == 'insignia'|test$model == 'x_reihe'|test$model == 'scirocco'|test
$model == 'lodgy'|test$model == 'v60'|test$model == 'viano'|test$model == 'cx_reih
e'|test$model == 'm_reihe'|test$model == 'exeo'|test$model == '6er'|test$model ==
'juke'|test$model == 'sl'|test$model == 'qashqai'|test$model == 'cayenne'|test$mod
el == 'boxster'|test$model == 'defender'|test$model == 'yeti'|test$model == 'b_max
'|test$model == 'tiguan'|test$model == 'kuga'|test$model == '911'|test$model == 'a
1'|test$model == 'cc'|test$model == 'a5'|test$model == 'range_rover_sport'|test$mo
del == 'q5'|test$model == 'q7',1,0)

test$model.medium = ifelse(test$model.low == 0 & test$model.high == 0,1,0)
```

```
# Transform Fuel type to three types
test$FT.hybrid = ifelse(test$fuelType == 'hybrid',1,0)
test$FT.diesel.elektro = ifelse(test$fuelType == 'diesel' & test$fuelType == 'elek
tro' ,1,0)
test$FT.others = ifelse(test$FT.hybrid == 0 & test$FT.diesel.elektro == 0 ,1,0)

# Transform brand type to three types

test$brand.daewoo.rover = ifelse(test$brand == 'daewoo'|test$brand == 'rover' ,1,0
)
test$brand.porsche = ifelse(test$brand == 'porsche',1,0)
test$brand.others = ifelse(test$brand.daewoo.rover == 0 &test$brand.porsche == 0 ,
1,0)

# save new train data
test_trans = test[,c(-2,-6,-8,-9)]
test_trans = test_trans[,c(-9,-10,-14,-15,-16)]

test_trans$brand.others = as.factor(test_trans$brand.others)
test_trans$brand.porsche = as.factor(test_trans$brand.porsche)
test_trans$FT.hybrid = as.factor(test_trans$FT.hybrid)
test_trans$model.medium = as.factor(test_trans$model.medium)
test_trans$model.high = as.factor(test_trans$model.high)
test_trans$VT.andere.klein = as.factor(test_trans$VT.andere.klein)
test_trans$VT.suv = as.factor(test_trans$VT.suv)
test_trans$notRepairedDamage = as.factor(test_trans$notRepairedDamage)
test_trans$gearbox = as.factor(test_trans$gearbox)
```

Model 3 : GAMs

Reasons for using GAM GAMs allow us to fit a non-linear fj to each Xj. This means that we do not need to manually try out many different transformations on each variable individually. The non-linear fits can potentially make more accurate predictions for the response Y

```
library(splines)
#install.packages("foreach")
library(gam)
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20
```

```
cut_kilo = 17500
cut_powerPs = c(625,6250)
cut_year_register = c(1960,1997)

gam = lm(price ~ ns(kilometer, knots = cut_kilo) +ns(powerPS, knots = cut_powerPs)
+ns(yearOfRegistration, knots = cut_year_register) + gearbox+notRepairedDamage + V
T.suv + VT.andere.klein + model.medium + model.high + FT.hybrid +brand.porsche + b
rand.others, data = train_trans)

AIC(gam)
```

```
## [1] 1377498
```

```
predict.gam.train = predict(gam,train_trans)
mean((predict.gam.train-train_trans$price)^2)
```

```
## [1] 4677049
```

```
predict.gam = predict(gam,test_trans)
MSE.gam = mean((predict.gam-test_trans$price)^2)
```

Model4 : Regression Tree

```
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

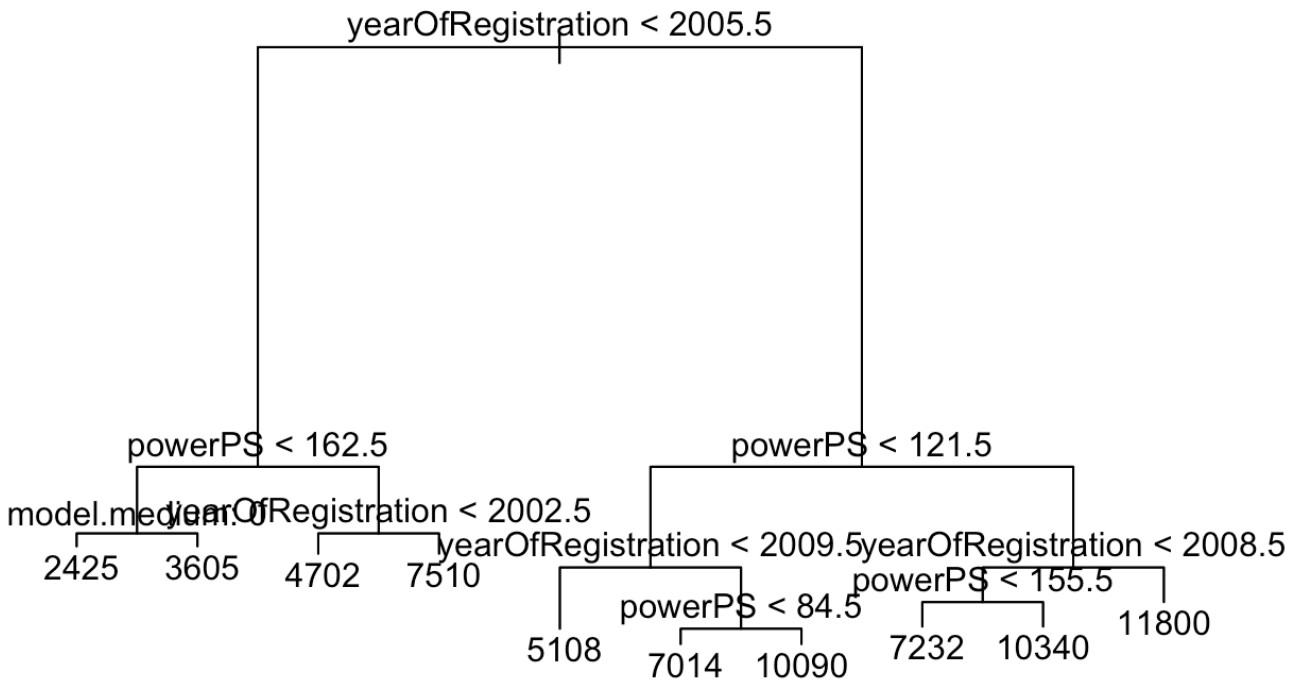
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)

# Regression Tree
tree.train = tree(formula = price ~. , data = train_trans)
summary(tree.train)
```

```
##
## Regression tree:
## tree(formula = price ~ ., data = train_trans)
## Variables actually used in tree construction:
## [1] "yearOfRegistration" "powerPS"            "model.medium"
## Number of terminal nodes:  10
## Residual mean deviance:  5854000 = 4.431e+11 / 75690
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10600.0  -1458.0   -409.6      0.0   1225.0  13580.0
```

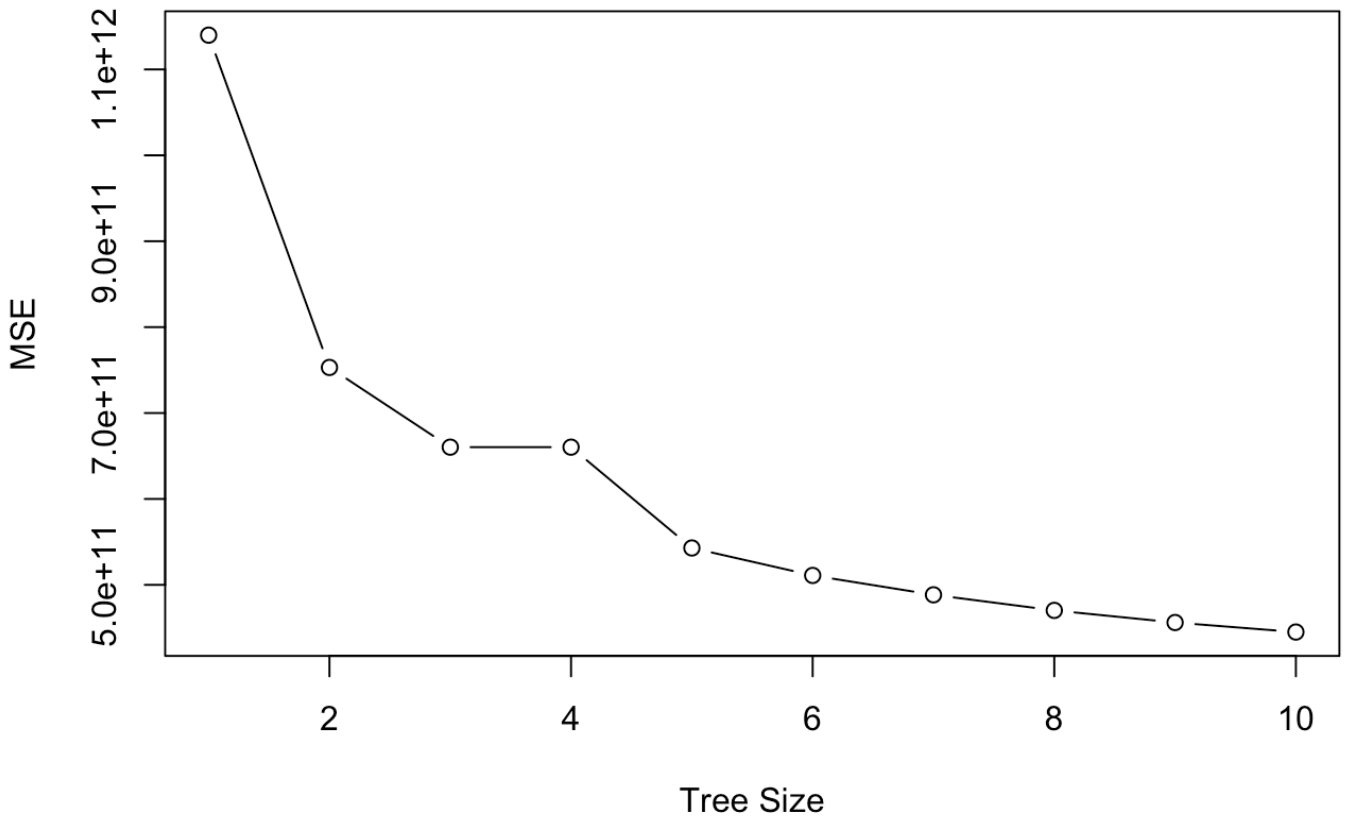
```
plot(tree.train)
text(tree.train,pretty=0)
```



```
yhat=predict(tree.train,newdata=test_trans)
mean((yhat-test_trans$price)^2)
```

```
## [1] 6164074
```

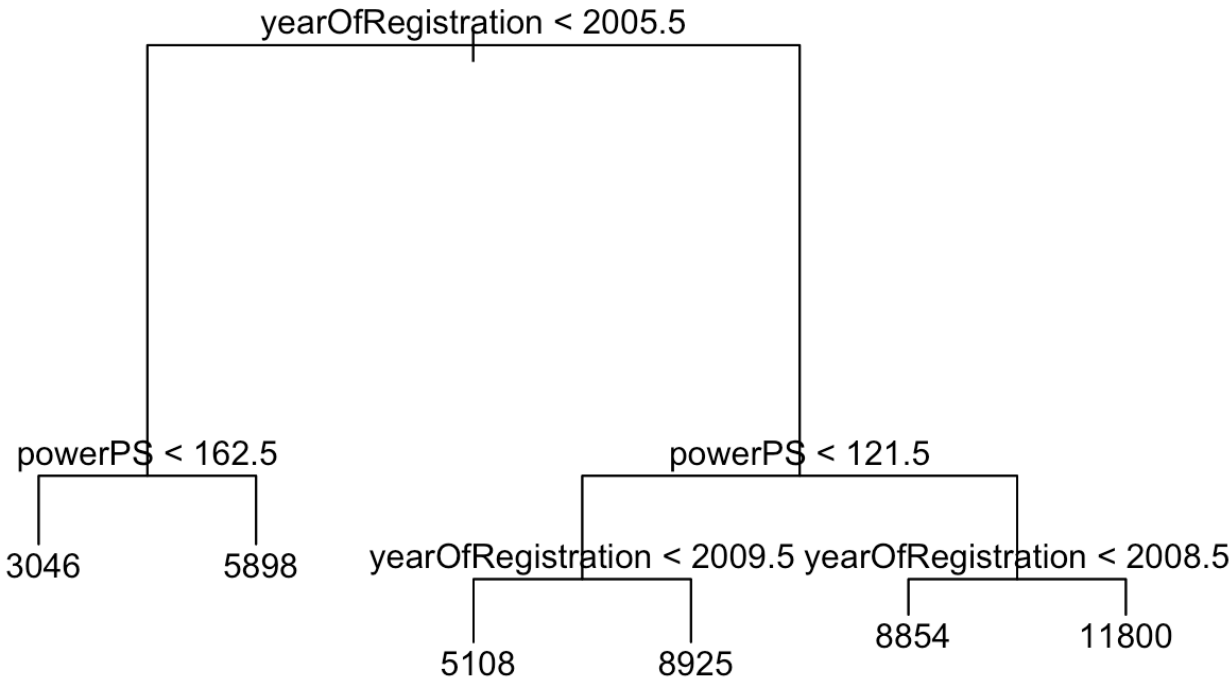
```
# CV
cv.train = cv.tree(tree.train)
plot(cv.train$size ,cv.train$dev ,type='b', xlab = "Tree Size", ylab = "MSE")
```




```
# prune the tree
prune.tree.train=prune.tree(tree.train ,best=6)
yhat=predict(prune.tree.train,newdata=test_trans)
mean((yhat-test_trans$price)^2)
```

```
## [1] 6716194
```

```
plot(prune.tree.train)
text(prune.tree.train,pretty=0)
```


yearOfRegistration < 2005.5
powerPS < 162.5
powerPS < 121.5
3046
5898
yearOfRegistration < 2009.5
yearOfRegistration < 2008.5
5108
8925
8854
11800