SVM Regression

Linus Fackler, Fernando Colman

This data set contains information about car sales to predict the price of a car given a set of data. This information contains the price, year, model, odometer reading, and other values. The data set can be found here: https://www.kaggle.com/datasets/deepcontractor/car-price-prediction-challenge

Load the data & package

```
library(e1071)
df <- read.csv("car_price_prediction.csv", header = TRUE)
str(df)
```

```
## 'data.frame':    19237 obs. of  18 variables:
##  $ ID              : int  45654403 44731507 45774419 45769185 45809263
45802912 45656768 45816158 45641395 45756839 ...
##  $ Price           : int  13328 16621 8467 3607 11726 39493 1803 549 1098
26657 ...
##  $ Levy            : chr  "1399" "1018" "-" "862" ...
##  $ Manufacturer    : chr  "LEXUS" "CHEVROLET" "HONDA" "FORD" ...
##  $ Model           : chr  "RX 450" "Equinox" "FIT" "Escape" ...
##  $ Prod..year      : int  2010 2011 2006 2011 2014 2016 2010 2013 2014
2007 ...
##  $ Category        : chr  "Jeep" "Jeep" "Hatchback" "Jeep" ...
##  $ Leather.interior: chr  "Yes" "No" "No" "Yes" ...
##  $ Fuel.type       : chr  "Hybrid" "Petrol" "Petrol" "Hybrid" ...
##  $ Engine.volume   : chr  "3.5" "3" "1.3" "2.5" ...
##  $ Mileage         : chr  "186005 km" "192000 km" "200000 km" "168966 km"
...
##  $ Cylinders       : num  6 6 4 4 4 4 4 4 4 6 ...
##  $ Gear.box.type   : chr  "Automatic" "Tiptronic" "Variator" "Automatic"
...
##  $ Drive.wheels    : chr  "4x4" "4x4" "Front" "4x4" ...
##  $ Doors           : chr  "04-May" "04-May" "04-May" "04-May" ...
##  $ Wheel           : chr  "Left wheel" "Left wheel" "Right-hand drive"
"Left wheel" ...
##  $ Color           : chr  "Silver" "Black" "Black" "White" ...
##  $ Airbags         : int  12 8 2 0 4 4 12 12 12 12 ...
```

Data cleaning

First, we have to clean the data. Some columns are unnecessary, which is why we are going to throw them out. Some contain numbers followed by characters, which is why we have to convert them first to just numbers.

We throw out columns “ID”, “Levy”, and “Engine volume”, since they don’t help us much in predicting the car price.

```
df <- df[, c(2,6,7,8,9,11,12,13,14,15,16,17,18)]
str(df)

## 'data.frame':    19237 obs. of  13 variables:
##  $ Price           : int  13328 16621 8467 3607 11726 39493 1803 549 1098
26657 ...
##  $ Prod..year      : int  2010 2011 2006 2011 2014 2016 2010 2013 2014
2007 ...
##  $ Category        : chr  "Jeep" "Jeep" "Hatchback" "Jeep" ...
##  $ Leather.interior: chr  "Yes" "No" "No" "Yes" ...
##  $ Fuel.type       : chr  "Hybrid" "Petrol" "Petrol" "Hybrid" ...
##  $ Mileage         : chr  "186005 km" "192000 km" "200000 km" "168966 km"
...
##  $ Cylinders       : num  6 6 4 4 4 4 4 4 4 6 ...
##  $ Gear.box.type   : chr  "Automatic" "Tiptronic" "Variator" "Automatic"
...
##  $ Drive.wheels    : chr  "4x4" "4x4" "Front" "4x4" ...
##  $ Doors           : chr  "04-May" "04-May" "04-May" "04-May" ...
##  $ Wheel           : chr  "Left wheel" "Left wheel" "Right-hand drive"
"Left wheel" ...
##  $ Color           : chr  "Silver" "Black" "Black" "White" ...
##  $ Airbags         : int  12 8 2 0 4 4 12 12 12 12 ...
```

We check how many NA rows there are.

```
sapply(df, function(x) sum(is.na(x)==TRUE))

##            Price        Prod..year         Category Leather.interior
##                0                 0                0                0
##        Fuel.type           Mileage        Cylinders    Gear.box.type
##                0                 0                0                0
##     Drive.wheels             Doors            Wheel            Color
##                0                 0                0                0
##          Airbags
##                0
```

None, that’s great.

For simplicity reasons, we are going to rename some of the columns, as their names are too unnecessarily long.

```
names(df)[2] <- "Year"
names(df)[4] <- "Leather"
names(df)[5] <- "Fuel"
names(df)[8] <- "Gearbox"
names(df)[9] <- "Drivetrain"
str(df)

## 'data.frame':    19237 obs. of  13 variables:
##  $ Price     : int  13328 16621 8467 3607 11726 39493 1803 549 1098 26657
...
```

```
##  $ Year      : int  2010 2011 2006 2011 2014 2016 2010 2013 2014 2007 ...
##  $ Category  : chr  "Jeep" "Jeep" "Hatchback" "Jeep" ...
##  $ Leather   : chr  "Yes" "No" "No" "Yes" ...
##  $ Fuel      : chr  "Hybrid" "Petrol" "Petrol" "Hybrid" ...
##  $ Mileage   : chr  "186005 km" "192000 km" "200000 km" "168966 km" ...
##  $ Cylinders : num  6 6 4 4 4 4 4 4 4 6 ...
##  $ Gearbox   : chr  "Automatic" "Tiptronic" "Variator" "Automatic" ...
##  $ Drivetrain: chr  "4x4" "4x4" "Front" "4x4" ...
##  $ Doors     : chr  "04-May" "04-May" "04-May" "04-May" ...
##  $ Wheel     : chr  "Left wheel" "Left wheel" "Right-hand drive" "Left
wheel" ...
##  $ Color     : chr  "Silver" "Black" "Black" "White" ...
##  $ Airbags   : int  12 8 2 0 4 4 12 12 12 12 ...
```

We are going to check, how many unique values the column “Leather” seats has.

```
unique(df$Leather)

## [1] "Yes" "No"
```

There is no need for further changing this column.

We will need to cut off the “km” after the mileage, to make it into an integer value.

```
df$Mileage[1:10]

##  [1] "186005 km" "192000 km" "200000 km" "168966 km" "91901 km"  "160931
km"
##  [7] "258909 km" "216118 km" "398069 km" "128500 km"

df$Mileage <- gsub(" .*","",df$Mileage)
df$Mileage[1:10]

##  [1] "186005" "192000" "200000" "168966" "91901"  "160931" "258909"
"216118"
##  [9] "398069" "128500"
```

Now, we transform it to integer values.

```
df <- transform(df, Mileage = as.integer(Mileage))
str(df)

## 'data.frame':    19237 obs. of  13 variables:
##  $ Price     : int  13328 16621 8467 3607 11726 39493 1803 549 1098 26657
...
##  $ Year      : int  2010 2011 2006 2011 2014 2016 2010 2013 2014 2007 ...
##  $ Category  : chr  "Jeep" "Jeep" "Hatchback" "Jeep" ...
##  $ Leather   : chr  "Yes" "No" "No" "Yes" ...
##  $ Fuel      : chr  "Hybrid" "Petrol" "Petrol" "Hybrid" ...
##  $ Mileage   : int  186005 192000 200000 168966 91901 160931 258909 216118
398069 128500 ...
##  $ Cylinders : num  6 6 4 4 4 4 4 4 4 6 ...
```

```
## $ Gearbox : chr "Automatic" "Tiptronic" "Variator" "Automatic" ...
## $ Drivetrain: chr "4x4" "4x4" "Front" "4x4" ...
## $ Doors : chr "04-May" "04-May" "04-May" "04-May" ...
## $ Wheel : chr "Left wheel" "Left wheel" "Right-hand drive" "Left
wheel" ...
## $ Color : chr "Silver" "Black" "Black" "White" ...
## $ Airbags : int 12 8 2 0 4 4 12 12 12 12 ...
```

Let's check the maximum value of mileage.

```
max(df$Mileage)

## [1] 2147483647

length(df$Mileage[df$Mileage > 500000])

## [1] 258
```

We see that 2147483647 is a very unrealistic number of kilometers. There are 258 observations with a mileage of over 500,000 kilometers, which is why we will throw them out.

```
df <- df[df$Mileage < 500000,]
```

We can check the same for the price.

```
max(df$Price)

## [1] 26307500

length(df$Price[df$Price > 200000])

## [1] 13
```

These unrealistically high values will negatively impact our data, which is why we will throw everything over 200,000 out.

```
df <- df[df$Price < 200000,]
```

We will also change Cylinders to integer values.

```
df <- transform(df, Cylinders = as.integer(Cylinders))
str(df)

## 'data.frame': 18963 obs. of 13 variables:
## $ Price : int 13328 16621 8467 3607 11726 39493 1803 549 1098 26657
...
## $ Year : int 2010 2011 2006 2011 2014 2016 2010 2013 2014 2007 ...
## $ Category : chr "Jeep" "Jeep" "Hatchback" "Jeep" ...
## $ Leather : chr "Yes" "No" "No" "Yes" ...
## $ Fuel : chr "Hybrid" "Petrol" "Petrol" "Hybrid" ...
## $ Mileage : int 186005 192000 200000 168966 91901 160931 258909 216118
398069 128500 ...
```

```
##  $ Cylinders : int  6 6 4 4 4 4 4 4 4 6 ...
##  $ Gearbox   : chr  "Automatic" "Tiptronic" "Variator" "Automatic" ...
##  $ Drivetrain: chr  "4x4" "4x4" "Front" "4x4" ...
##  $ Doors     : chr  "04-May" "04-May" "04-May" "04-May" ...
##  $ Wheel     : chr  "Left wheel" "Left wheel" "Right-hand drive" "Left
wheel" ...
##  $ Color     : chr  "Silver" "Black" "Black" "White" ...
##  $ Airbags   : int  12 8 2 0 4 4 12 12 12 12 ...
```

For some reason, the doors have “May” and “Mar” attached to them. We will get rid of them and change it to integer values.

```
unique(df$Doors)

## [1] "04-May" "02-Mar" ">5"

length(df$Doors[df$Doors == ">5"])

## [1] 125

df$Model[df$Doors == ">5"][1:20]

## NULL
```

After looking closer into rows where doors are “>5”, we see that this is most likely just an error, so we will throw these rows out. It is just 128 out of over 19,000 rows, so it won’t affect our results.

```
df <- df[!(df$Doors == ">5"),]
```

We are now going to take out the -Mar and -May, and convert it to integer values

```
df$Doors <- gsub("-.*","",df$Doors)
df$Doors <- gsub("0","",df$Doors)
df$Doors[1:10]

##  [1] "4" "4" "4" "4" "4" "4" "4" "4" "4" "4"

df <- transform(df, Doors = as.integer(Doors))
str(df)

## 'data.frame':    18838 obs. of  13 variables:
##  $ Price     : int  13328 16621 8467 3607 11726 39493 1803 549 1098 26657
...
##  $ Year      : int  2010 2011 2006 2011 2014 2016 2010 2013 2014 2007 ...
##  $ Category  : chr  "Jeep" "Jeep" "Hatchback" "Jeep" ...
##  $ Leather   : chr  "Yes" "No" "No" "Yes" ...
##  $ Fuel      : chr  "Hybrid" "Petrol" "Petrol" "Hybrid" ...
##  $ Mileage   : int  186005 192000 200000 168966 91901 160931 258909 216118
398069 128500 ...
##  $ Cylinders : int  6 6 4 4 4 4 4 4 4 6 ...
##  $ Gearbox   : chr  "Automatic" "Tiptronic" "Variator" "Automatic" ...
##  $ Drivetrain: chr  "4x4" "4x4" "Front" "4x4" ...
```

```
##  $ Doors     : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ Wheel     : chr  "Left wheel" "Left wheel" "Right-hand drive" "Left
wheel" ...
##  $ Color     : chr  "Silver" "Black" "Black" "White" ...
##  $ Airbags   : int  12 8 2 0 4 4 12 12 12 12 ...
```

Factorizing all chr data

```
df$Category <- factor(df$Category)
df$Leather <- factor(df$Leather)
df$Fuel <- factor(df$Fuel)
df$Gearbox <- factor(df$Gearbox)
df$Drivetrain <- factor(df$Drivetrain)
df$Wheel <- factor(df$Wheel)
df$Color <- factor(df$Color)

str(df)

## 'data.frame':    18838 obs. of  13 variables:
##  $ Price     : int  13328 16621 8467 3607 11726 39493 1803 549 1098 26657
...
##  $ Year      : int  2010 2011 2006 2011 2014 2016 2010 2013 2014 2007 ...
##  $ Category  : Factor w/ 11 levels "Cabriolet","Coupe",..: 5 5 4 5 4 5 4
10 10 5 ...
##  $ Leather   : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 2 2 2 2 ...
##  $ Fuel      : Factor w/ 7 levels "CNG","Diesel",..: 3 6 6 3 6 2 3 6 3 6
...
##  $ Mileage   : int  186005 192000 200000 168966 91901 160931 258909 216118
398069 128500 ...
##  $ Cylinders : int  6 6 4 4 4 4 4 4 4 6 ...
##  $ Gearbox   : Factor w/ 4 levels "Automatic","Manual",..: 1 3 4 1 1 1 1 1
1 1 ...
##  $ Drivetrain: Factor w/ 3 levels "4x4","Front",..: 1 1 2 1 2 2 2 2 2 1
...
##  $ Doors     : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ Wheel     : Factor w/ 2 levels "Left wheel","Right-hand drive": 1 1 2 1
1 1 1 1 1 1 ...
##  $ Color     : Factor w/ 16 levels "Beige","Black",..: 13 2 2 15 13 15 15
8 2 13 ...
##  $ Airbags   : int  12 8 2 0 4 4 12 12 12 12 ...
```

Train and Test sets

Divide into train and test sets

```
set.seed(1234)
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df), nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

Data Exploration of Training Data

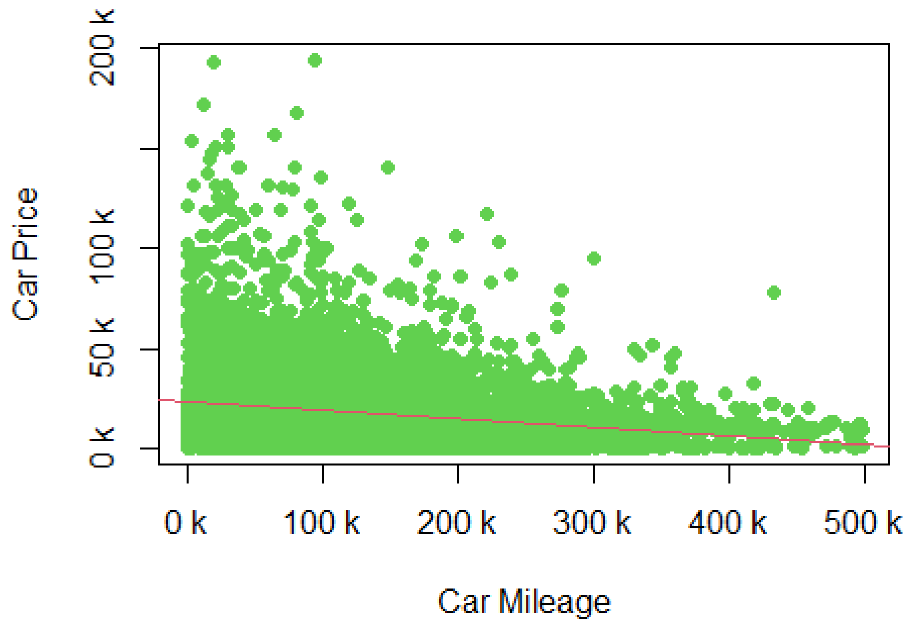
First, we will explore the training data statistically and graphically

```
print(paste("Number of Rows: ", nrow(train)))

## [1] "Number of Rows:  11302"

print(paste("Average price: ", mean(train$Price)))

## [1] "Average price:  17312.4615997169"

print(paste("Median price: ", median(train$Price)))

## [1] "Median price:  13328"

print(paste("Average mileage: ", mean(train$Mileage)))

## [1] "Average mileage:  135654.299681472"

print(paste("Median mileage: ", median(train$Mileage)))

## [1] "Median mileage:  125000"

print(paste("Avergae number of airbags: ", mean(train$Airbags)))

## [1] "Avergae number of airbags:  6.58308264024067"

print(paste("Median number of airbags: ", median(train$Airbags)))

## [1] "Median number of airbags:  6"
```

This seems like pretty realist data for car sales.

Now, we'll plot the car price and mileage, to see if there is some obvious correlation.

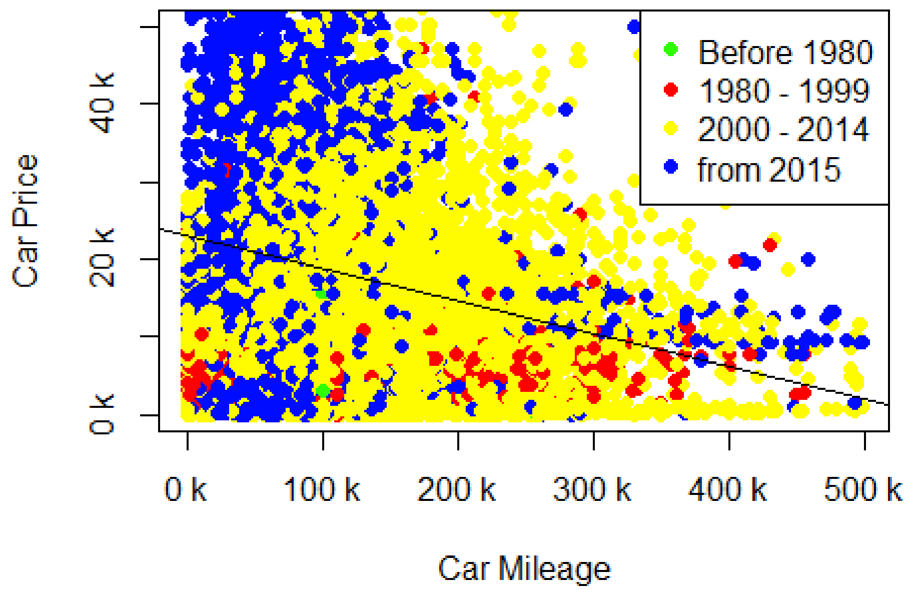
```
plot(train$Price ~ train$Mileage, xlab = "Car Mileage", ylab = "Car Price",
yaxt = "n", xaxt="n", col = 3, pch = 19)
xTicks = axTicks(1)
yTicks = axTicks(2)
axis(1, at=xTicks, labels = paste(formatC(xTicks / 1000, format = 'd'), 'k',
sep = ' '))
axis(2, at=yTicks, labels = paste(formatC(yTicks / 1000, format = 'd'), 'k',
sep = ' '))
abline(lm(train$Price ~ train$Mileage), col = 2)
```



As expected, the price goes down if the mileage goes up. Let's take a closer look at that.

We will group the cars into their production years.

```
colors <- c("#2EFF00", #green:  before 1980
            "#FF0000", #red:    between 1980 - 1999
            "#FFFB00", #yellow  between 2000 - 2014
            "#000CFF" #blue     from 2015
           )

yrs <- train$Year
group <- ifelse(yrs < 1980, 1, ifelse(yrs < 2000, 2, ifelse(yrs < 2015, 3,
4)))


plot(train$Price ~ train$Mileage, xlab = "Car Mileage", ylab = "Car Price",
yaxt = "n", xaxt="n",  col=colors[group], pch = 19, ylim=c(0,50000))
xTicks = axTicks(1)
yTicks = axTicks(2)
axis(1, at=xTicks, labels = paste(formatC(xTicks / 1000, format = 'd'), 'k',
sep = ' '))
axis(2, at=yTicks, labels = paste(formatC(yTicks / 1000, format = 'd'), 'k',
sep = ' '))
abline(lm(train$Price ~ train$Mileage), col = 1)

legend("topright", legend=c("Before 1980", "1980 - 1999", "2000 - 2014",
"from 2015"), pch = 19, col=colors)
```



This shows us that cars with a production year after 2014 will tend to be more expensive and have less kilometers. Cars that are over 20 years old are in the lower third of the prices. Cars older than 42 are not seen at all in this graph.

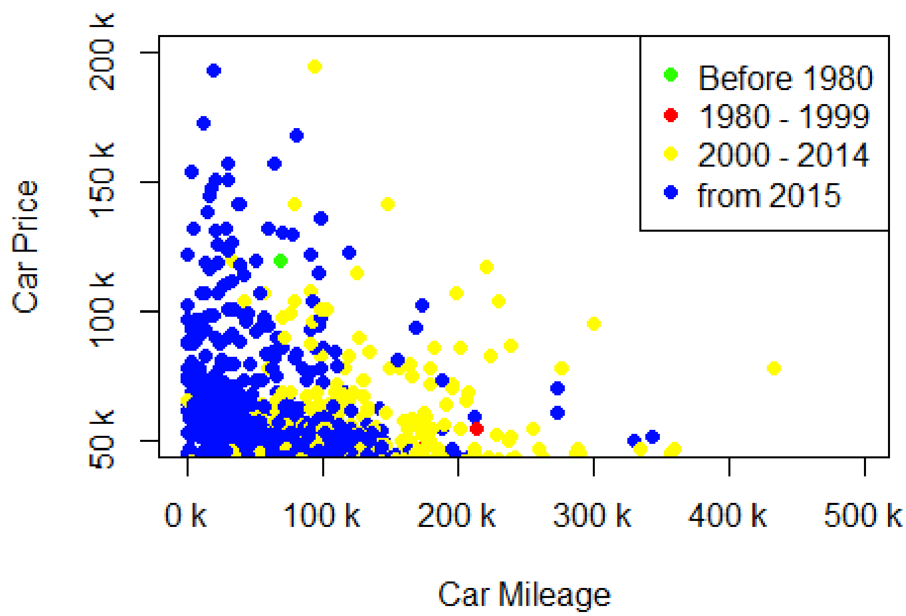
```
length(train$Year[train$Year < 1980])

## [1] 13
```

No surprise, there is only 20 cars in our training data set that is built before 1980.

Now, let's look at cars with a value of 50 - 200k.

```
colors <- c("#2EFF00", #green:  before 1980
            "#FF0000", #red:    between 1980 - 1999
            "#FFFB00", #yellow  between 2000 - 2014
            "#000CFF" #blue     from 2015
           )

yrs <- train$Year
group <- ifelse(yrs < 1980, 1, ifelse(yrs < 2000, 2, ifelse(yrs < 2015, 3,
4)))


plot(train$Price ~ train$Mileage, xlab = "Car Mileage", ylab = "Car Price",
yaxt = "n", xaxt="n",  col=colors[group], pch = 19, ylim=c(50000,200000))
xTicks = axTicks(1)
yTicks = axTicks(2)
```

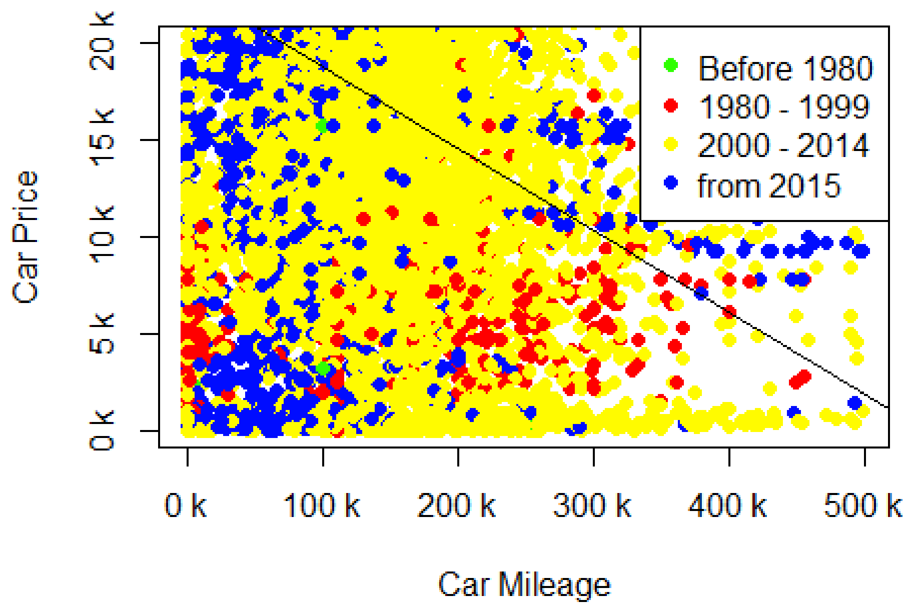
```
axis(1, at=xTicks, labels = paste(formatC(xTicks / 1000, format = 'd'), 'k',
sep = ' '))
axis(2, at=yTicks, labels = paste(formatC(yTicks / 1000, format = 'd'), 'k',
sep = ' '))

legend("topright", legend=c("Before 1980", "1980 - 1999", "2000 - 2014",
"from 2015"), pch = 19, col=colors)
```



This section is largely dominated by cars built after 2014.

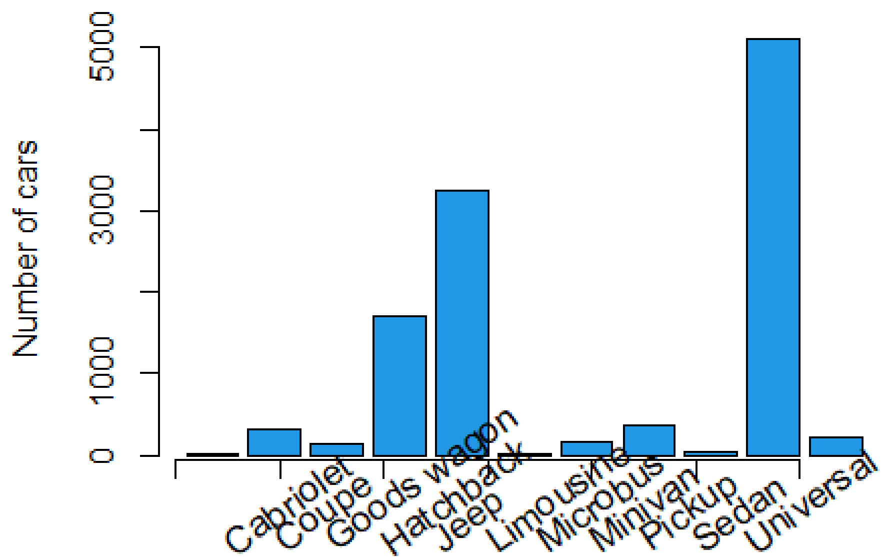
```
colors <- c("#2EFF00", #green:  before 1980
            "#FF0000", #red:    between 1980 - 1999
            "#FFFB00", #yellow  between 2000 - 2014
            "#000CFF" #blue     from 2015
           )

yrs <- train$Year
group <- ifelse(yrs < 1980, 1, ifelse(yrs < 2000, 2, ifelse(yrs < 2015, 3,
4)))


plot(train$Price ~ train$Mileage, xlab = "Car Mileage", ylab = "Car Price",
yaxt = "n", xaxt="n",  col=colors[group], pch = 19, ylim=c(0,20000))
xTicks = axTicks(1)
yTicks = axTicks(2)
axis(1, at=xTicks, labels = paste(formatC(xTicks / 1000, format = 'd'), 'k',
sep = ' '))
```

```
axis(2, at=yTicks, labels = paste(formatC(yTicks / 1000, format = 'd'), 'k',
sep = ' '))
abline(lm(train$Price ~ train$Mileage), col = 1)
legend("topright", legend=c("Before 1980", "1980 - 1999", "2000 - 2014",
"from 2015"), pch = 19, col=colors)
```



This plot shows cars with a price under 20,000. Again, the price goes down the more kilometers it has.

```
counts <- table(train$Category)
barplot(counts, ylab = "Number of cars", xlab = "", xaxt = "n", col = 4)

axis(1, labels=FALSE)
text(x = 0:(length(counts) - 1),
     y = -1500,
     labels = paste("      ", names(counts)),
     xpd = NA,
     srt = 35,
     cex = 1.1,
     adj = 0)
```



Linear Regression

```
lm1 <- lm(Price~., data=train)
summary(lm1)

##
## Call:
## lm(formula = Price ~ ., data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41074  -8267  -1305   5918 180695
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -2.314e+06  7.194e+04 -32.167  < 2e-16 ***
## Year                   1.160e+03  3.589e+01  32.317  < 2e-16 ***
## CategoryCoupe         -3.453e+03  3.517e+03  -0.982 0.326198
## CategoryGoods wagon   -1.193e+04  3.725e+03  -3.202 0.001369 **
## CategoryHatchback     -9.096e+03  3.493e+03  -2.604 0.009231 **
## CategoryJeep          -1.178e+03  3.487e+03  -0.338 0.735438
## CategoryLimousine      2.268e+04  6.180e+03   3.670 0.000243 ***
## CategoryMicrobus      -6.638e+03  3.679e+03  -1.804 0.071211 .
## CategoryMinivan       -3.640e+03  3.563e+03  -1.022 0.306981
## CategoryPickup         3.429e+03  4.524e+03   0.758 0.448468
## CategorySedan         -9.624e+03  3.474e+03  -2.771 0.005605 **
## CategoryUniversal     -1.592e+03  3.627e+03  -0.439 0.660646
```

```
## LeatherYes            -4.626e+02  4.034e+02  -1.147 0.251469    
## FuelDiesel             4.544e+03  9.963e+02   4.561 5.15e-06 ***
## FuelHybrid            -4.258e+03  1.023e+03  -4.162 3.18e-05 ***
## FuelHydrogen          -8.295e+03  1.491e+04  -0.556 0.578082    
## FuelLPG               -4.164e+02  1.207e+03  -0.345 0.730231    
## FuelPetrol            -1.171e+03  9.509e+02  -1.231 0.218300    
## FuelPlug-in Hybrid     8.055e+03  2.332e+03   3.455 0.000553 ***
## Mileage               -2.749e-02  1.713e-03 -16.052  < 2e-16 ***
## Cylinders              1.558e+03  1.506e+02  10.347  < 2e-16 ***
## GearboxManual          5.627e+03  6.932e+02   8.117 5.27e-16 ***
## GearboxTiptronic       1.208e+04  4.222e+02  28.613  < 2e-16 ***
## GearboxVariator        7.193e+03  7.905e+02   9.099  < 2e-16 ***
## DrivetrainFront        2.347e+03  4.727e+02   4.964 6.99e-07 ***
## DrivetrainRear         3.698e+03  6.036e+02   6.126 9.31e-10 ***
## Doors                  5.248e+02  4.693e+02   1.118 0.263505    
## WheelRight-hand drive -2.918e+03  6.178e+02  -4.723 2.36e-06 ***
## ColorBlack             6.361e+00  1.736e+03   0.004 0.997076    
## ColorBlue             -1.321e+03  1.788e+03  -0.739 0.459977    
## ColorBrown             1.733e+03  2.215e+03   0.782 0.434057    
## ColorCarnelian red    -5.286e+02  2.264e+03  -0.233 0.815425    
## ColorGolden            1.653e+03  2.320e+03   0.712 0.476187    
## ColorGreen             5.683e+02  2.047e+03   0.278 0.781320    
## ColorGrey              8.759e+02  1.760e+03   0.498 0.618758    
## ColorOrange            1.793e+03  2.365e+03   0.758 0.448350    
## ColorPink              1.476e+03  5.041e+03   0.293 0.769722    
## ColorPurple           -2.964e+03  3.746e+03  -0.791 0.428836    
## ColorRed              -2.085e+03  1.876e+03  -1.112 0.266258    
## ColorSilver           -1.195e+03  1.740e+03  -0.687 0.492186    
## ColorSky blue          2.171e+03  2.435e+03   0.891 0.372704    
## ColorWhite             2.125e+02  1.738e+03   0.122 0.902674    
## ColorYellow            1.246e+03  2.465e+03   0.506 0.613153    
## Airbags               -5.237e+02  3.690e+01 -14.192  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14870 on 11258 degrees of freedom
## Multiple R-squared:  0.2985, Adjusted R-squared:  0.2958
## F-statistic: 111.4 on 43 and 11258 DF,  p-value: < 2.2e-16
```

```r
pred <- predict(lm1, newdata = test)
cor_lm1 <- cor(pred, test$Price)
mse_lm1 <- mean((pred - test$Price) ^2)

print(paste("cor=", cor_lm1))
```

```
## [1] "cor= 0.525781076156056"
```

```r
print(paste("mse=", mse_lm1))
```

```
## [1] "mse= 193424474.691449"
```

Linear Kernel

We will have to use smaller data samples, otherwise my computer won't be able to compute the following models. SVM with 11000 observations takes about 8 minutes. Tuning couldn't even finish, it reached the maximum number of iterations.

```
trainsmall <- head(train, 2000)
testsmall <- head(test, 500)
valdsmall <- head(vald, 500)
svm1 <- svm(Price~., data=trainsmall, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = Price ~ ., data = trainsmall, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.02272727
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1681
```

```
pred <- predict(svm1, newdata=testsmall)
cor_svm1 <- cor(pred, testsmall$Price)
mse_svm1 <- mean((pred - testsmall$Price) ^2)
print(paste("cor=", cor_svm1))
```

```
## [1] "cor= 0.523067369892093"
```

```
print(paste("mse=", mse_svm1))
```

```
## [1] "mse= 168603894.790165"
```

Tune

```
tune_svm1 <- tune(svm, Price~. , data=valdsmall, kernel="linear",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
```

```
##   0.1
##
## - best performance: 243940906
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 291937015  116926804
## 2 1e-02 255949493   98456449
## 3 1e-01 243940906   87934321
## 4 1e+00 250391135   84300344
## 5 5e+00 251851422   84202646
## 6 1e+01 251873175   84248854
## 7 1e+02 252067005   84415121
```

Evaluate on best linear svm

```
pred <- predict(tune_svm1$best.model, newdata = testsmall)
cor_svm1_tune <- cor(pred, testsmall$Price)
mse_svm1_tune <- mean((pred - testsmall$Price) ^2)
print(paste("cor=", cor_svm1_tune))

## [1] "cor= 0.514899488912912"

print(paste("mse=", mse_svm1_tune))

## [1] "mse= 176416898.639076"
```

Polynomial Kernel

```
svm2 <- svm(Price~., data=trainsmall, kernel="polynomial", cost=10, scale =
TRUE)
summary(svm2)

##
## Call:
## svm(formula = Price ~ ., data = trainsmall, kernel = "polynomial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  0.02272727
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1577

pred <- predict(svm2, newdata = testsmall)
cor_svm2 <- cor(pred, testsmall$Price)
```

```
mse_svm2 <- mean((pred - testsmall$Price) ^2)
print(paste("cor=", cor_svm2))

## [1] "cor= 0.630556779356833"

print(paste("mse=", mse_svm2))

## [1] "mse= 144990455.730843"
```

Tune

```
tune_svm2 <- tune(svm, Price~. , data=valdsmall, kernel="polynomial",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune_svm2)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 213861004
##
## - Detailed performance results:
##     cost     error dispersion
## 1 1e-03 320540535  136272664
## 2 1e-02 320311942  136284675
## 3 1e-01 317341143  135655312
## 4 1e+00 302912094  132070005
## 5 5e+00 275978709  123209232
## 6 1e+01 261143097  119429034
## 7 1e+02 213861004   90454855
```

Evaluate on best linear svm

```
pred <- predict(tune_svm2$best.model, newdata = testsmall)
cor_svm2_tune <- cor(pred, testsmall$Price)
mse_svm2_tune <- mean((pred - testsmall$Price) ^2)
print(paste("cor=", cor_svm2_tune))

## [1] "cor= 0.474380238754904"

print(paste("mse=", mse_svm2_tune))

## [1] "mse= 237326314.430851"
```

The pre-tuned polynomial svm was better, as this used a 100 cost.

Radial Kernel

```
svm3 <- svm(Price~., data=trainsmall, kernel="radial", cost=10, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = Price ~ ., data = trainsmall, kernel = "radial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.02272727
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1545
```

```
pred <- predict(svm3, newdata = testsmall)
cor_svm3 <- cor(pred, testsmall$Price)
mse_svm3 <- mean((pred - testsmall$Price) ^2)
print(paste("cor=", cor_svm3))
```

```
## [1] "cor= 0.695253196627123"
```

```
print(paste("mse=", mse_svm3))
```

```
## [1] "mse= 121875946.04193"
```

Tune hyperperameters

```
set.seed(1234)
tune.out <- tune(svm, Price~., data=valdsmall, kernel="radial",
ranges=list(cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 231031179
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 283015990  136824406
## 2  1e+00   0.5 231051545  108956205
## 3  1e+01   0.5 231031179   95218483
## 4  1e+02   0.5 260025782   87163389
## 5  1e+03   0.5 326759443  107199292
```

```
## 6   1e-01   1.0 298952988  139496002
## 7   1e+00   1.0 264247251  126884496
## 8   1e+01   1.0 261324505  107294508
## 9   1e+02   1.0 281453741  115982248
## 10  1e+03   1.0 291186053  119750273
## 11  1e-01   2.0 308238712  139883449
## 12  1e+00   2.0 287127710  135662818
## 13  1e+01   2.0 289350276  120019813
## 14  1e+02   2.0 297547565  124749814
## 15  1e+03   2.0 340410786  167685428
## 16  1e-01   3.0 310659591  139726767
## 17  1e+00   3.0 293328952  137420818
## 18  1e+01   3.0 294832705  122836759
## 19  1e+02   3.0 300934235  125455587
## 20  1e+03   3.0 321178042  140717663
## 21  1e-01   4.0 311989519  139798836
## 22  1e+00   4.0 295660969  137445722
## 23  1e+01   4.0 296308605  123324026
## 24  1e+02   4.0 306084624  127537743
## 25  1e+03   4.0 311877424  131280533
```

Cost = 10 and gamma = 0.5 shows clearly the lowest error and dispersion.

```
svm4 <- svm(Price~., data=trainsmall, kernel="radial", cost=10, gamma=0.5,
scale=TRUE)
summary(svm4)

##
## Call:
## svm(formula = Price ~ ., data = trainsmall, kernel = "radial", cost = 10,
##     gamma = 0.5, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1556

pred <- predict(svm4, newdata = testsmall)
cor_svm4 <- cor(pred, testsmall$Price)
mse_svm4 <- mean((pred - testsmall$Price) ^2)
print(paste("cor=", cor_svm4))

## [1] "cor= 0.728147563806647"

print(paste("mse=", mse_svm4))
```

```
## [1] "mse= 110893534.092218"
```

Analysis

The Radial Kernel with tuned hyperparameters will give us the best result of 0.72 correlation. The second best was the polynomial kernel with 0.63, and third was the linear kernel with 0.52. Linear Regression was as good as the linear kernel SVM.

Looking at the data provided, it was obvious that the radial kernel would outperform the other ones. The data is very cluttered and is not at all linearly separable. This is why the linear kernel didn’t work well. The polynomial was definitely better, but still couldn’t perfectly handle our messy data.

Therefore, the radial was best in this case. With a very big difference to the linear regression.