Data Science Development

Rafael Castillo

19/03/2021

Introduction

Many factors can determine a laptop’s price, for example, the hardware or brand. A linear regression model can be trained to predict a laptop’s price by obtaining a dataset containing laptop’s details. This model could be used to know if a laptop is overpriced or at what price a laptop can be sold. This analysis could also help to understand what are the variables which are most correlated and significant towards the computer price.

The dataset for this analysis contains specifications about laptops and their price. the dataset has been obtained from Kaggle under the following link: https://www.kaggle.com/ionaskel/laptop-prices?select=laptops.csv (https://www.kaggle.com/ionaskel/laptop-prices?select=laptops.csv)

Dataset Dimensions (Row x Column) : 1303 X 13

- Dataset Code Book:
  - Index: Number of the row in the dataset. | int
  - Company: The name of the company that manufactured the laptop. | Factor
  - Product: Name of the device. | Factor
  - TypeName: Laptop category, e.g.(netbook, notebook, ultrabook). | Factor
  - Inches: Screen size in inches. | num
  - ScreenResolution: Screen Resolution. | Factor
  - Cpu: CPU Characteristics. | Factor
  - Ram: RAM Capacity. | Factor
  - Memory: Storage Capacity. | Factor
  - Gpu: GPU Characteristics. | Factor
  - OpSys: Operating System installed on the laptop. | Factor
  - Weight: Laptop’s Weight. | Factor
  - Price_euros: Laptop’s Price in euros. | num | Target Variable

Importing dataset and libraries

```
setwd("D:/OneDrive/Desktop/MASTERS/Data Science Development")
laptops = read.csv("laptops.csv", stringsAsFactors = T)
library(ggplot2)
library(plyr)
library(dplyr)
library(gsubfn)
library(stringi)
library(ggpubr)
library(corrplot)
library(MASS)
library(leaps)
#library(devtools)
library(caret)
library(e1071)
library(kernlab)
library(tidyverse)
#install_github("vqv/ggbiplot")
```

Exploratory Data Analysis and Pre-processing

```
# Displays a preview of the information contained within the dataset
head(laptops, 4)
```

```
##   index Company     Product  TypeName Inches                   ScreenResolution
## 1     1   Apple MacBook Pro Ultrabook   13.3 IPS Panel Retina Display 2560x1600
## 2     2   Apple Macbook Air Ultrabook   13.3                           1440x900
## 3     3      HP      250 G6  Notebook   15.6                  Full HD 1920x1080
## 4     4   Apple MacBook Pro Ultrabook   15.4 IPS Panel Retina Display 2880x1800
##                          Cpu  Ram               Memory
## 1       Intel Core i5 2.3GHz  8GB            128GB SSD
## 2       Intel Core i5 1.8GHz  8GB 128GB Flash Storage
## 3 Intel Core i5 7200U 2.5GHz  8GB            256GB SSD
## 4       Intel Core i7 2.7GHz 16GB            512GB SSD
##                            Gpu OpSys Weight Price_euros
## 1 Intel Iris Plus Graphics 640 macOS 1.37kg     1339.69
## 2       Intel HD Graphics 6000 macOS 1.34kg      898.94
## 3        Intel HD Graphics 620 No OS 1.86kg      575.00
## 4           AMD Radeon Pro 455 macOS 1.83kg     2537.45
```

```
# Displaying the internal structure of an R object
str(laptops)
```

```
## 'data.frame':    1303 obs. of  13 variables:
##  $ index           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Company         : Factor w/ 19 levels "Acer","Apple",..: 2 2 8 2 2 1 2 2 3 1 ...
##  $ Product         : Factor w/ 618 levels "110-15ACL (A6-7310/4GB/500GB/W10)",..: 302 300 51 302 302 59 302 30
0 615 431 ...
##  $ TypeName        : Factor w/ 6 levels "2 in 1 Convertible",..: 5 5 4 5 5 4 5 5 5 5 ...
##  $ Inches          : num  13.3 13.3 15.6 15.4 13.3 15.6 15.4 13.3 14 14 ...
##  $ ScreenResolution: Factor w/ 40 levels "1366x768","1440x900",..: 24 2 9 26 24 1 26 2 9 16 ...
##  $ Cpu             : Factor w/ 118 levels "AMD A10-Series 9600P 2.4GHz",..: 55 53 64 75 57 15 74 53 96 73 ...
##  $ Ram             : Factor w/ 9 levels "12GB","16GB",..: 9 9 9 2 9 6 2 9 2 9 ...
##  $ Memory          : Factor w/ 39 levels "1.0TB HDD","1.0TB Hybrid",..: 5 3 17 30 17 27 16 16 30 17 ...
##  $ Gpu             : Factor w/ 110 levels "AMD FirePro W4190M",..: 59 52 54 10 60 18 61 52 98 62 ...
##  $ OpSys           : Factor w/ 9 levels "Android","Chrome OS",..: 5 5 6 5 5 7 4 5 7 7 ...
##  $ Weight          : Factor w/ 179 levels "0.69kg","0.81kg",..: 39 36 75 72 39 106 91 36 42 63 ...
##  $ Price_euros     : num  1340 899 575 2537 1804 ...
```

```
# Displaying a statistical analysis of all the variables in the dataframe
summary(laptops)
```

```
##      index           Company                  Product
##  Min.   :   1.0   Dell   :297   XPS 13            :  30
##  1st Qu.: 331.5   Lenovo :297   Inspiron 3567     :  29
##  Median : 659.0   HP     :274   250 G6            :  21
##  Mean   : 660.2   Asus   :158   Legion Y520-15IKBN:  19
##  3rd Qu.: 990.5   Acer   :103   Vostro 3568       :  19
##  Max.   :1320.0   MSI    : 54   Inspiron 5570     :  18
##                   (Other):120   (Other)           :1167
##                TypeName       Inches
##  2 in 1 Convertible:121   Min.   :10.10
##  Gaming            :205   1st Qu.:14.00
##  Netbook           : 25   Median :15.60
##  Notebook          :727   Mean   :15.02
##  Ultrabook         :196   3rd Qu.:15.60
##  Workstation       : 29   Max.   :18.40
##
##                                 ScreenResolution
##  Full HD 1920x1080                        :507
##  1366x768                                 :281
##  IPS Panel Full HD 1920x1080              :230
##  IPS Panel Full HD / Touchscreen 1920x1080: 53
##  Full HD / Touchscreen 1920x1080          : 47
##  1600x900                                 : 23
##  (Other)                                  :162
##                          Cpu           Ram                       Memory
##  Intel Core i5 7200U 2.5GHz :190   8GB    :619   256GB SSD           :412
##  Intel Core i7 7700HQ 2.8GHz:146   4GB    :375   1TB HDD             :223
##  Intel Core i7 7500U 2.7GHz :134   16GB   :200   500GB HDD           :132
##  Intel Core i7 8550U 1.8GHz : 73   6GB    : 41   512GB SSD           :118
##  Intel Core i5 8250U 1.6GHz : 72   12GB   : 25   128GB SSD +  1TB HDD: 94
##  Intel Core i5 6200U 2.3GHz : 68   2GB    : 22   128GB SSD           : 76
##  (Other)                    :620   (Other): 21   (Other)             :248
##                      Gpu             OpSys          Weight     Price_euros
##  Intel HD Graphics 620  :281   Windows 10:1072   2.2kg  :121   Min.   : 174
##  Intel HD Graphics 520  :185   No OS     :  66   2.1kg  : 58   1st Qu.: 599
##  Intel UHD Graphics 620 : 68   Linux     :  62   2.4kg  : 44   Median : 977
##  Nvidia GeForce GTX 1050: 66   Windows 7 :  45   2.3kg  : 41   Mean   :1124
##  Nvidia GeForce GTX 1060: 48   Chrome OS :  27   2.5kg  : 38   3rd Qu.:1488
##  Nvidia GeForce 940MX   : 43   macOS     :  13   2kg    : 35   Max.   :6099
##  (Other)                :612   (Other)   :  18   (Other):966
```

Based on the previous analysis, we can see our dataset contains 13 variables and 1303 rows, of which we can see most of them are factors and a few numeric. Many of the factors contain a high number of levels, this will need to be handled. We can also see that there are no missing values. Based on these analyses, the following pre-processing is proposed.

Dataset Pre-processing Analysis

This pre-processing aims to simplify certain aspects of the dataset and allow for more qualitative variables that can be used to train the linear model. I will be working on each column in order from the first one to the last one.

Index

The index variable contains the index of each row, this column is of no significance to the project and will be removed.

```
# Dropping the index column of dataset
laptops["index"] = NULL
```

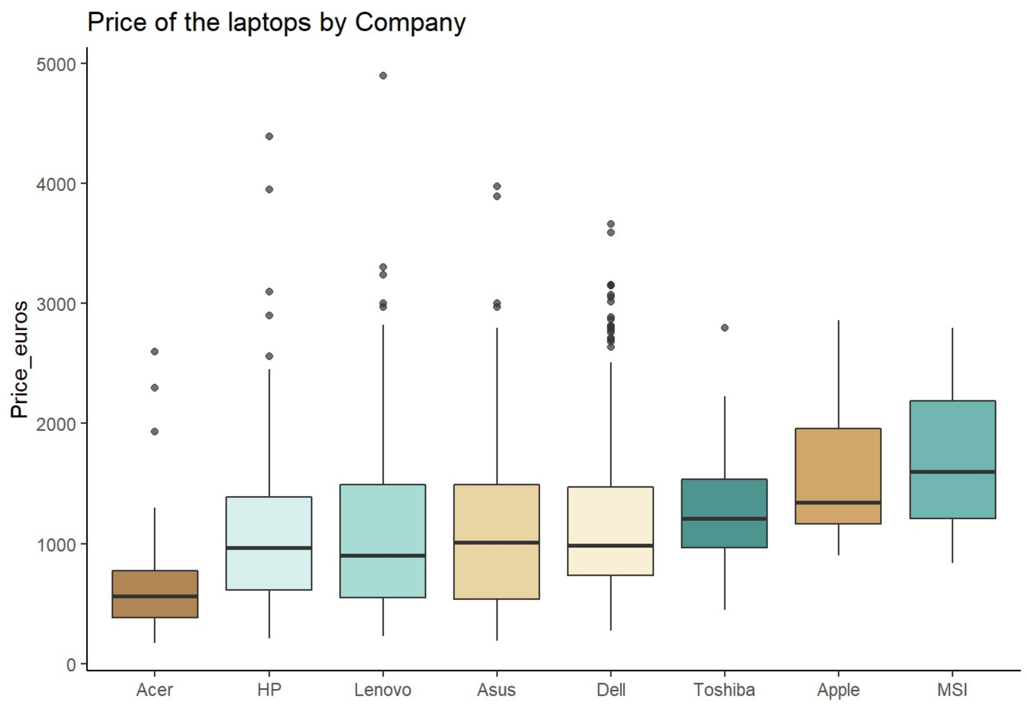
Company

If there is any brand with a small number of observations I might decide to remove it.

```
# The following code creates and sorts the information of Company to the its brand distribution
sort(table(laptops$Company))
```

```
##
##    Huawei     Chuwi   Fujitsu    Google        LG      Vero    Xiaomi Microsoft
##         2         3         3         3         3         4         4         6
##  Mediacom     Razer   Samsung     Apple   Toshiba       MSI      Acer      Asus
##         7         7         9        21        48        54       103       158
##        HP      Dell    Lenovo
##       274       297       297
```

```
# Removes the rows and levels of the companies with limited amount of observations
laptops = droplevels(laptops[!laptops$Company == "Huawei" & !laptops$Company == "Chuwi" & !laptops$Company == "Fu
jitsu" & !laptops$Company == "Google" & !laptops$Company == "LG" & !laptops$Company == "Vero" & !laptops$Company
== "Xiaomi" & !laptops$Company == "Microsoft" & !laptops$Company == "Mediacom" & !laptops$Company == "Samsung" &
!laptops$Company == "Razer", ])

# Bar chart to explore the price of the laptops by brand
ggplot(laptops, aes(x=reorder(Company, Price_euros), y=Price_euros, fill=Company)) + ggtitle('Price of the laptop
s by Company')+ xlab("") + geom_boxplot(alpha=0.7) + theme_classic() + theme(legend.position="none") + scale_fill
_brewer(palette="BrBG")
```



I have decided to delete all the brands with less than 20 instances as these labels will not impact the model as much.

I printed a box plot which shows the quartiles of the prices for each company. As expected, we can see that the gaming company MSI has laptops that require the most high-end specification hence the highest prices. Apple seems to be the second most expensive computer even though they don’t contain specific gaming hardware, but the brand has the tendency to be overpriced. The cheapest laptop in the group is the Acer; Taiwanese brands tend to be cheaper.

Optionally I would have renamed all the companies with fewer observations to “Others” in order to make use of the other quantitative variables on these rows as I think these could hold significant information.

Product

This variable seems to contain 618 levels which are too many to draw any differences or conclusions. Since the Product name comes as a combination of the brand name and the computer’s specification, I will be removing this column.

```
# Dropping the Product column of dataset
laptops["Product"] = NULL
```

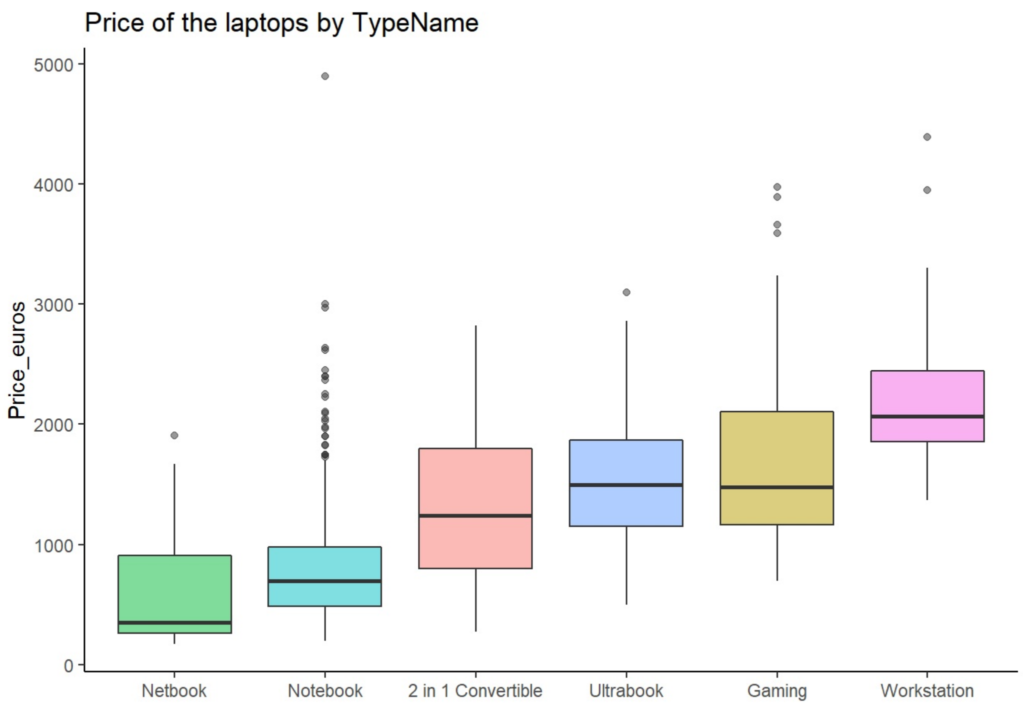
TypeName

If there is any TypeName with a small number of observations, I might decide to remove it.

```
# The following code creates and sorts the information of TypeName to the its distribution
sort(table(laptops$TypeName))
```

```
## 
##            Netbook        Workstation 2 in 1 Convertible          Ultrabook 
##                 24                 29                118                173 
##             Gaming           Notebook 
##                200                708
```

```
# Bar chart to explore the price of the laptops by TypeName
ggplot(laptops, aes(x=reorder(TypeName, Price_euros), y=Price_euros, fill=TypeName)) + ggtitle('Price of the lapt
ops by TypeName')+ xlab("") + geom_boxplot(alpha=0.5) + theme_classic() + theme(legend.position="none")
```



Looking at the distribution of laptops' price by their types, workstations which are generally designed to perform intense processes such as rendering, 3D animation, CAD, data analysis, and video editing, are the most expensive.

We can also see gaming laptops come in second place due to their high-end hardware, similar to workstations, but as the graph showed, it possesses a competitive median value with Ultrabook, making them similar in that regard.

Netbook, Notebook, Ultrabook follow a naming hierarchy; Netbooks are slimmer, lighter, and offers a more simplified set of tools, followed by Notebooks which tend to have a better set of tools, as shown in our graph, these can range a lot in price considering all the outliers it contains. The most expensive of the hierarchy being the Ultrabook which possesses more powerful hardware.

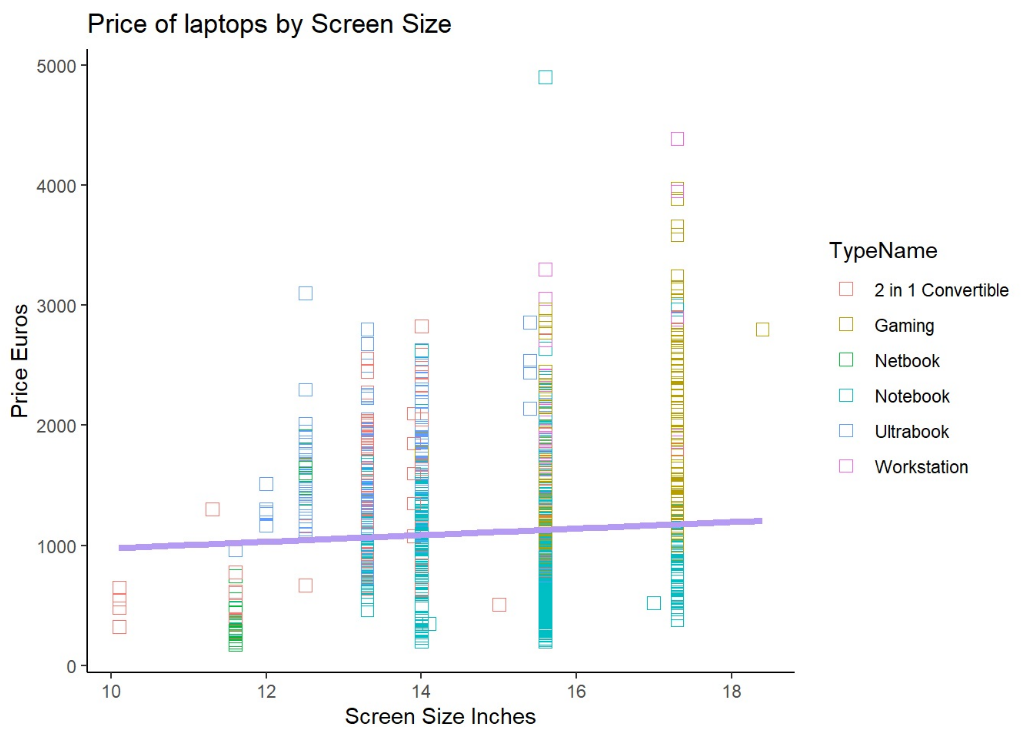
Lastly, the 2 in 1 convertible can also be used as tablets, these aren't cheap due to their flexibility, as seen on our graph, but they aren't as expensive since the amount of space for powerful hardware is limited.

I can conclude that the graph makes sense in how it has classified price ranges with types and consider there to be a correlation and there is not enough evidence to reject any of the levels.

Inches

The Inches of the screen size of the laptop can be used as a quantitative variable for training the linear model.

```
# qqplot showcasing the distribution of remote laptop price by Screen Size filled with TypeName
ggplot(laptops, aes(x=Inches, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=22)+
ggtitle('Price of laptops by Screen Size') + xlab('Screen Size Inches') + ylab('Price Euros')+
geom_smooth(method=lm, se=FALSE, col='#B69BF5', size=1.5) + theme_classic()
```



We can see that there isn't a straight linear relation in this model because of all the outliers found. This is because laptops with the same screen size can have cheap or costly hardware; in other words, the price depends on other variables. Nonetheless, the standard linear model in the graph does show a slight price increase when screen size increases.

As seen in the graph, convertibles which are also tablets, tend to have smaller screen size as well as netbooks; this is because these devices are meant to be small and hence not enough space to fit enough hardware to make them costly. We can also see notebooks and ultrabooks in the 13 to 16-inch range; these devices have more room for expensive hardware hence increasing in price. As we move on, we can see most gaming devices have bigger screen size; this can be due to gaming hardware requiring more room and a larger device needs a bigger screen. I can conclude that there is some evidence of a correlation between screen size and laptops price.

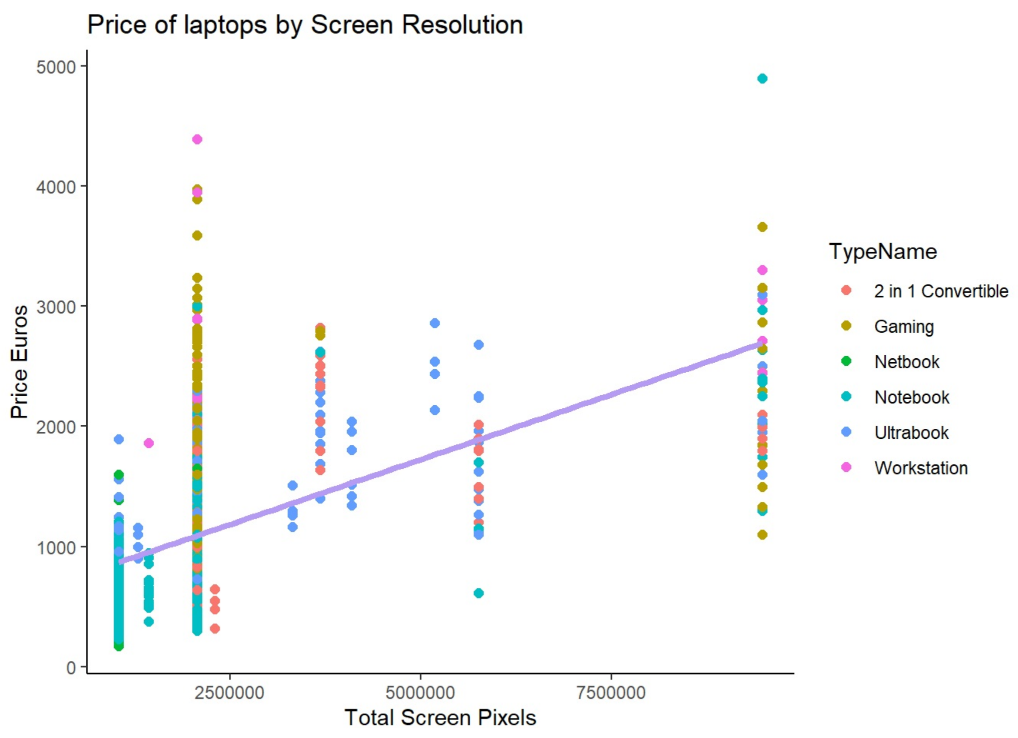
ScreenResolution

This column will need to be separated into multiple columns, capturing the resolution as a quantitative variable. Some laptops can have multiple screen types, e.g.(IPS, HD), one column for each, whether true or false, as quantitative.

```
# The following code creates and sorts the information of displayType to the its distribution
ScreenResoT = sort(table(laptops$ScreenResolution))
head(ScreenResoT,6)
```

```
##
##       IPS Panel Full HD 1366x768      IPS Panel Full HD 2560x1440
##                                1                                1
##  Touchscreen / Full HD 1920x1080 Touchscreen / Quad HD+ 3200x1800
##                                1                                1
##                        1920x1080     IPS Panel Quad HD+ 3200x1800
##                                2                                2
```

```
# Creates a copy
displayType = laptops$ScreenResolution

#Removes all non-digit characters from each screen resolution instance.
laptops$ScreenResolution = as.numeric(gsub("\\D", "", laptops$ScreenResolution))

#Removes all observations with no displayType
# screenResoBefore = nrow(laptops)
# laptops = na.omit(laptops)
# paste(c(screenResoBefore - nrow(laptops), "rows have been removed due to no display information"),  collapse =
" ")


#Divided the y and x pixels and multiplies them
yResolution = as.numeric(gsub("^\\d{4}", "", laptops$ScreenResolution))
xResolution = as.numeric(stri_extract_first_regex(laptops$ScreenResolution, "^\\d{4}"))
laptops$ScreenResolution = xResolution * yResolution

# qqplot showcasing the distribution of remote laptop price by Screen Resolution filled with TypeName
ggplot(laptops, aes(x=ScreenResolution, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=20)+
ggtitle('Price of laptops by Screen Resolution') + xlab('Total Screen Pixels') + ylab('Price Euros')+
geom_smooth(method=lm, se=FALSE, col='#B69BF5', size=1.5) + theme_classic()
```



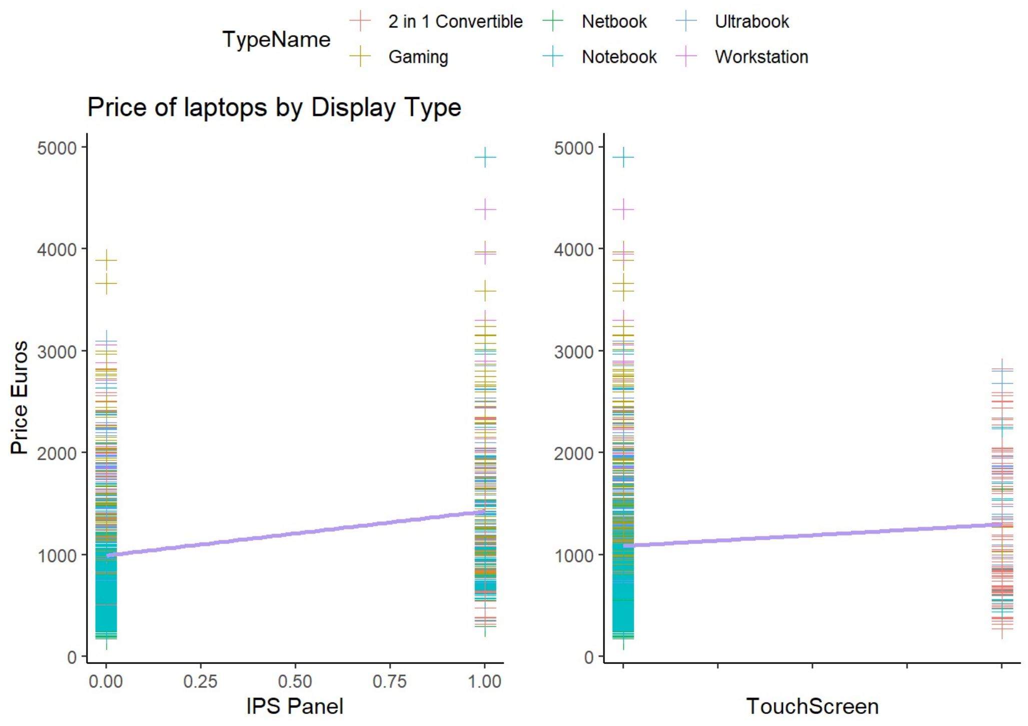
Based on the first few lines I have decided to remove the screen resolution number, multiply them into one number containing the total amount of pixels and substitute this new column for the existing ScreenResolution column.

The results of screen resolution are interesting; most observations are found at 250,000,000, i.e.(1920x1080), which is the most common resolution for screens. We can also see netbooks and notebooks dominate at smaller resolutions, which is expected as these devices are meant to be smaller, hence a minor need for pixels. In contrast, it is surprising that convertible laptops/tablets can be found at higher resolutions as these screens tend to be smaller. Another interesting factor is that the type of laptop found from 2.5m to 7.5m + varies a lot; in other words, gaming laptops don't dominate in higher screen resolutions at higher prices as we saw in previous graphs.

The standard linear model adapted to our graph shows a price increase when the screen resolution increases; it is hard to explain why because the observations seem to be very well mixed. I am inclined to assume that 2.5m resolution contains more costlier-laptop observations than 7.5m+ but since this resolution includes a vast range in hardware and price where most observations are found at lower-ends it takes the line to a lower point. Even though 7.5m+ has less costly-laptop observation than 2.5m the line finds an increase due to the amount of cheap laptops adopting 2.5m instead of 7.5m+.

I can conclude that there is evidence to show that the majority of high-resolution laptops are found at higher prices were as smaller resolution laptops also have high prices but these resolutions also includes a high amount of lower price laptops, i.e.(there is a higher chance of 2.5m pixel resolution laptop to be cheaper than it is at 7.5m+)

```
# Removes all digits for each observation
displayType = gsub("\\d(.*)$", "", displayType)
# Removes the space at the end of the observations
displayType = gsub("\\s$", "", displayType)
# Turns observations with empty strings into NA
displayType = gsub("^$", 0, displayType)

IPSPanel = as.numeric(as.logical(grepl("IPS Panel", displayType)))
TouchScreen = as.numeric(as.logical(grepl("Touchscreen", displayType)))

# Creates a dataframe with each of the display options
laptops$IPSPanel = IPSPanel
laptops$TouchScreen = TouchScreen

# qqplot showcasing the distribution of remote laptop price by Display Type filled with TypeName
IPSPanel = ggplot(laptops, aes(x=IPSPanel, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=3)+ ggt
itle('Price of laptops by Display Type') + xlab('IPS Panel') + ylab('Price Euros')+ geom_smooth(method=lm, se=FAL
SE, col='#B69BF5') + theme_classic()

# qqplot showcasing the distribution of remote laptop price by Display Type filled with TypeName
Touch = ggplot(laptops, aes(x=TouchScreen, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=3)+ xla
b('TouchScreen') + ylab('')+ geom_smooth(method=lm, se=FALSE, col='#B69BF5') + theme_classic()

ggarrange(IPSPanel, Touch + rremove("x.text"),
          labels = c("IPSPanel", "TouchScreen"),
          align = "hv",
          label.y = 1,
          common.legend = TRUE,
          legend = "top",
          hjust = -6,
          ncol = 2, nrow = 1)
```



I have removed the screen resolution number to deal only with the display type. I have also removed the Full HD tag since this indicates that a display is 1920x1080, and this information is already part of the resolution column. Two new columns have been created, showing whether a computer has or doesn't have the IPS or TouchScreen attributes.

The results show both a positive increase in laptop price when a computer has the displays features of IPS or TouchScreen. We can see IPS Panel is used for costly computers as seen in the Notebook, gaming and working stations. We can argue that the majority of computers with IPS display tend to have a higher price.

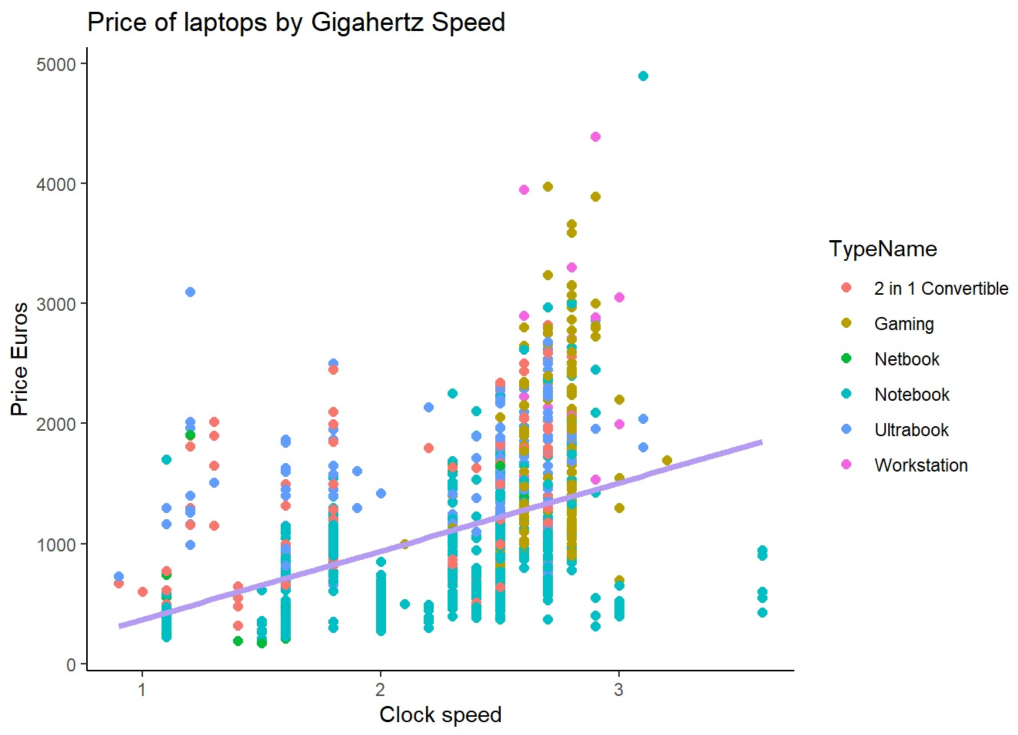
On the other hand, TouchScreen observations belong mainly to convertible laptops due to their tablet form and removable keyboards. Even though these devices are not the most expensive due to their limited room for intensive hardware, the linear model still leans towards a slight increase in price due to the number of low-cost laptops found in the without-touchscreen column.

I can conclude that there is evidence to suggest that touchscreen and IPS panel display increase the probabilities for that laptop to be found at higher prices.

CPU

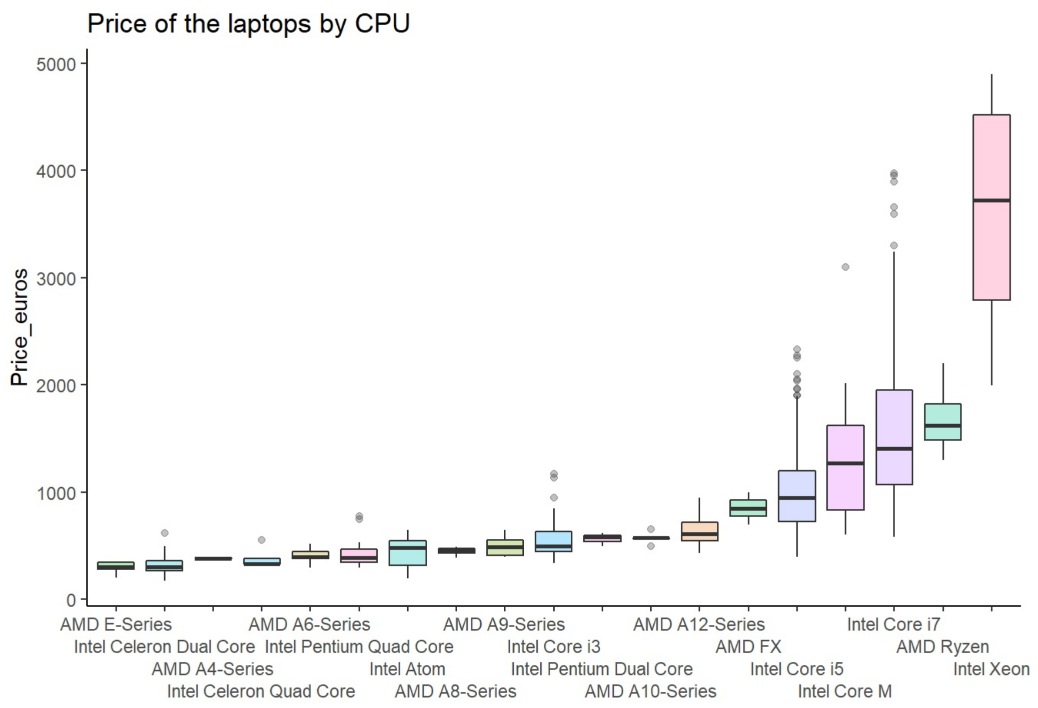
The laptop's CPU contains too many qualitative levels (118), it needs to be divided into two columns, a qualitative one denoting only CPU type and a quantitative denoting the gigahertz.

```
# Creates a copy of CPU to work with.
CPU = laptops$Cpu

# Extracts the gigahertz from the CPU column and assigns it as a new column to the dataframe
gigahertz = stri_extract_first_regex(CPU, "\\d?\\.?\\d{1,2}GHz$")
laptops$CPUGigahertz = as.numeric(stri_extract_first_regex(gigahertz, "^\\d?\\.?\\d"))

# qqplot showcasing the distribution of laptop price by gigahertz filled with TypeName
ggplot(laptops, aes(x=CPUGigahertz, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=20)+
ggtitle('Price of laptops by Gigahertz Speed') + xlab('Clock speed') + ylab('Price Euros')+
geom_smooth(method=lm, se=FALSE, col='#B69BF5', size=1.5) + theme_classic()
```



The laptop's Gigahertz speed results show a positive correlation towards the laptops price; the higher the clock speed of the CPU, the higher the price.

```
# Deletes the gigahertz and versions of CPU to leave only the name and series and
# Assigns it as a new column to the laptop dataframe.
CPUType = gsub("\\s\\d?\\.?\\d{1,2}GHz$", "", CPU)
CPUType = gsub("\\s(v|V)\\d", "", CPUType)
CPUType = gsub("\\s\\w{1,3}\\S?\\w{3,9}$", "", CPUType)
laptops$CPU = as.factor(gsub("\\sm3", "", CPUType))

# Drops the Cpu column
laptops$Cpu = NULL

# Bar chart to explore the price of the laptops by CPU
ggplot(laptops, aes(x=reorder(CPU, Price_euros), y=Price_euros, fill=CPU)) + ggtitle('Price of the laptops by CPU
')+ xlab("") + geom_boxplot(alpha=0.3) + theme_classic() + theme(legend.position="none") + scale_x_discrete(guide
= guide_axis(n.dodge=4))
```



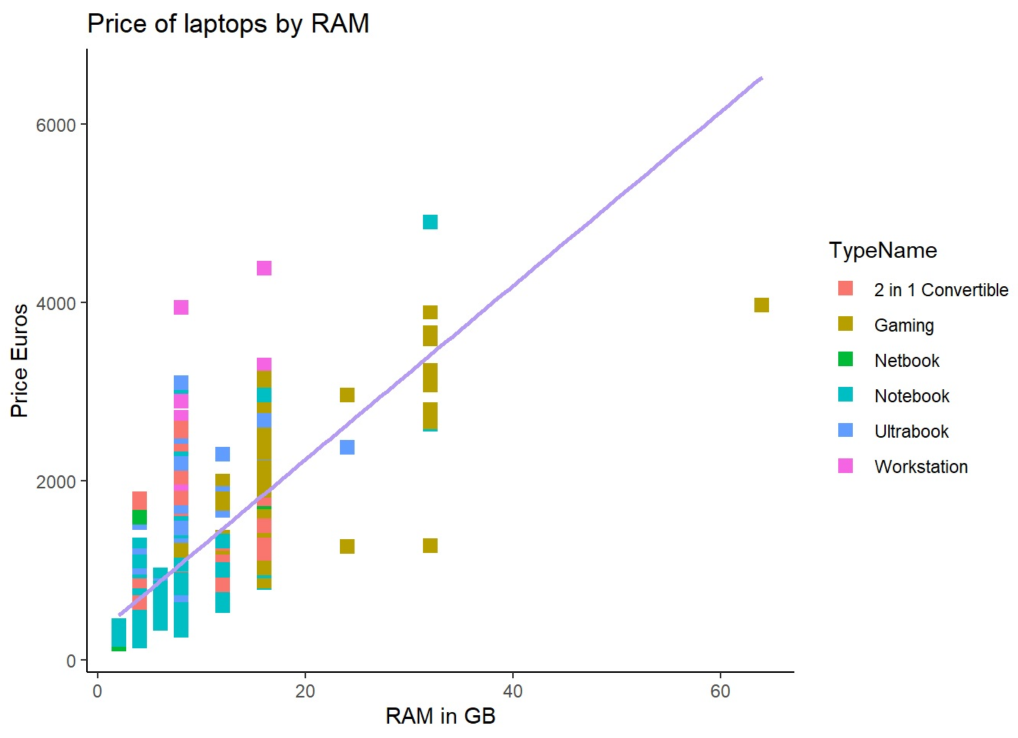
The laptop CPU results are as expected; it shows that laptops with lower-end CPU have a lower price, and as the CPU increases the price of the laptop also increases e.g.(AMD-A4, A6, A8, A9, A10, A12, Ryzen). Finally, the last two CPUs belong to gaming and content creation laptops which rank highest in price.

I can conclude that there is a strong correlation when it comes to CPU and the laptops price.

Ram

The ram column will be converted to a qualitative variable by removing the “GB” in each observation.

```
# Removes all non numerical characters from the string and assigns the result as a new column.
laptops$Ram = as.numeric(gsub("\\D", "", laptops$Ram))

# qqplot showcasing the distribution of remote laptop price by RAM filled with TypeName
ggplot(laptops, aes(x=Ram, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=15)+
ggtitle('Price of laptops by RAM') + xlab('RAM in GB') + ylab('Price Euros')+
geom_smooth(method=lm, se=FALSE, col='#B69BF5') + theme_classic()
```



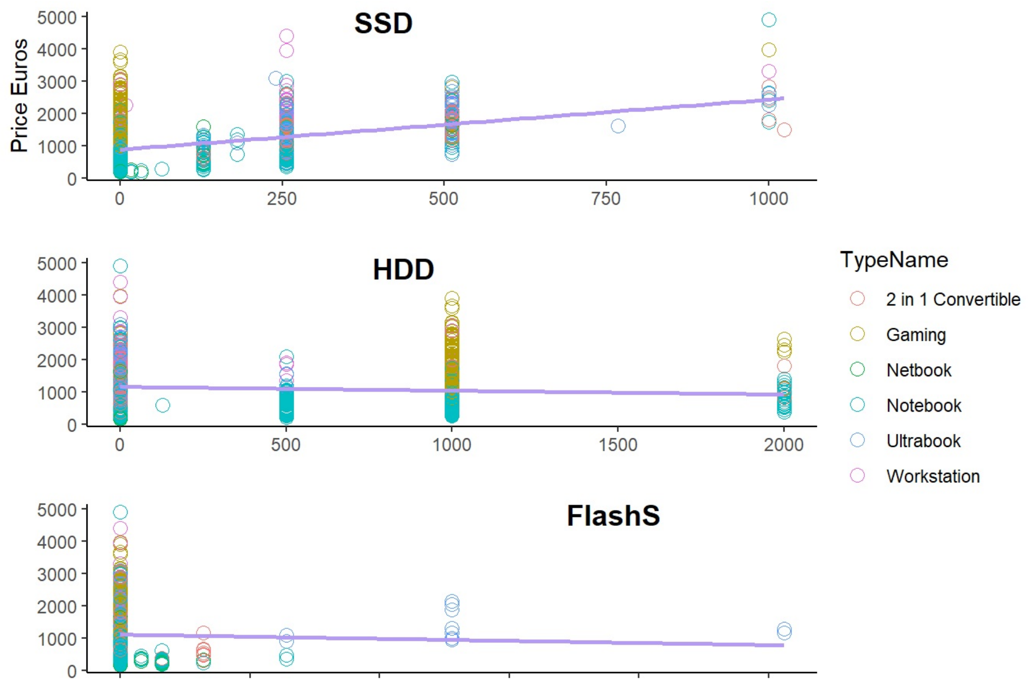
At first, we can recognize this graph as a positive linear relation with many outliers. There is enough evidence to suggest that the price of a computer increases when RAM increases. As shown by the colour filling, we can see that most netbook, notebooks which are light laptops usually not gear with expensive hardware, have a lower price. We can also see that laptops for content creation or gaming tend to have better hardware in general and are found at a higher price. We can conclude that RAM increases when other hardware such as CPU and GPU increases because

RAM is a complimentary resource. The graph also shows that laptops with specific RAM such as 16GB can vary between 1,000 to 3,000 euros due to these laptops having cheap or costly hardware hence the outliers. We can conclude that high-end hardware requires more RAM than low-end hardware, showing a reasonable correlation between ram and price.

Memory

The laptop's memory has redundancy in its qualitative format; it will be divided into columns, one for each memory type and converted to quantitative variables.

```r
#Fixing the format with regex to convert to  numeric columns
memory = laptops$Memory
memory = gsub("512GB SSD \\+  512GB SSD", "1024GB SSD", memory)
memory = gsub("1TB HDD", "1000GB HDD", memory)
memory = gsub("2TB HDD", "2000GB HDD", memory)
memory = gsub("1TB SSD", "1000GB SSD", memory)
memory = gsub("1.0TB HDD", "1000GB HDD", memory)
memory = gsub("1000GB HDD \\+  1000GB HDD", "2000GB HDD", memory)
memory = gsub("1.0TB Hybrid", NA, memory)
memory = gsub("512GB SSD \\+  256GB SSD", "768GB SSD", memory)
memory = gsub("256GB SSD \\+  256GB SSD", "512GB SSD", memory)

# Removes NA rows which were Hybrid memory.
laptops$Memory = memory
laptops = na.omit(laptops)

# Extracts the SSD values and assigns it as a new column in dataframe.
SSD = stri_extract_first_regex(laptops$Memory, "\\d{1,5}GB\\sSSD$")
SSD = as.numeric(gsub("\\D", "", SSD))
# If the value is empty it assgines a 0 to it.
SSD[is.na(SSD)] = 0
laptops$SSD = SSD

# Extracts the HHD values and assigns it as a new column in dataframe.
HDD = stri_extract_first_regex(laptops$Memory, "\\d{1,5}GB\\sHDD$")
HDD = as.numeric(gsub("\\D", "", HDD))
# If the value is empty it assgines a 0 to it.
HDD[is.na(HDD)] = 0
laptops$HDD = HDD

# Extracts the FlashS values and assigns it as a new column in dataframe.
FlashS = stri_extract_first_regex(laptops$Memory, "\\d{1,5}GB\\sFlash Storage$")
FlashS = as.numeric(gsub("\\D", "", FlashS))
# If the value is empty it assgines a 0 to it.
FlashS[is.na(FlashS)] = 0
laptops$FlashS = FlashS

# Removes the Memory column
laptops$Memory = NULL
# Preview of new columns
#tail(laptops[,15:17], 3)

# qqplot showcasing the distribution of remote laptop price by SSD filled with TypeName
SSD = ggplot(laptops, aes(x=SSD, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=21)+xlab('') + yl
ab('Price Euros')+
      geom_smooth(method=lm, se=FALSE, col='#B69BF5') + theme_classic() + theme(legend.position = "none")
# qqplot showcasing the distribution of remote laptop price by HDD filled with TypeName
HDD = ggplot(laptops, aes(x=HDD, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=21)+xlab('') + yl
ab('')+
      geom_smooth(method=lm, se=FALSE, col='#B69BF5') + theme_classic() + theme(legend.position = "none")
# qqplot showcasing the distribution of remote laptop price by FlashS filled with TypeName
FlashS = ggplot(laptops, aes(x=FlashS, y= Price_euros)) + geom_point(aes(col=TypeName), size=3, shape=21)+xlab(''
) + ylab('')+
      geom_smooth(method=lm, se=FALSE, col='#B69BF5') + theme_classic() + theme(legend.position = "none")

ggarrange(SSD, HDD, FlashS + rremove("x.text"),
          labels = c("SSD", "HDD", "FlashS"),
          align = "hv",
          label.y = 1,
          common.legend = TRUE,
          legend = "right",
          hjust = -6,
          ncol = 1, nrow = 3)
```



Regex has been used to change and simplify the format of the memory column. Regex was then used to extract and transform the values to numeric. The new format allows a laptop’s memory to be a numeric variable while keeping the category; this is important because SSD memory is much more expensive than HDD, and we want to avoid mixing them up.

Starting with SSD, i.e.(solid-state drive), this type of memory has a much higher price; hence, we can find a positive linear relationship with many outliers, as shown in the graph.; This means that there is modest evidence suggesting that the laptop’s price grows when more SSD memory is added to the laptop. We know there are outliers due to other laptop hardware.

HDD, i.e.(hard-disk drive), this type of memory is the most frequent in laptops. Our graph shows a weak negative linear relation with many outliers; regardless of how much HDD memory is being added to the laptop, the price usually will not rise and instead has weak evidence suggesting that it would drop. An explanation for this strange behavior can be assumed by implying that the laptops on the 0 to 1000GB also have SSD memory making them more expensive than 2000GB of HDD.

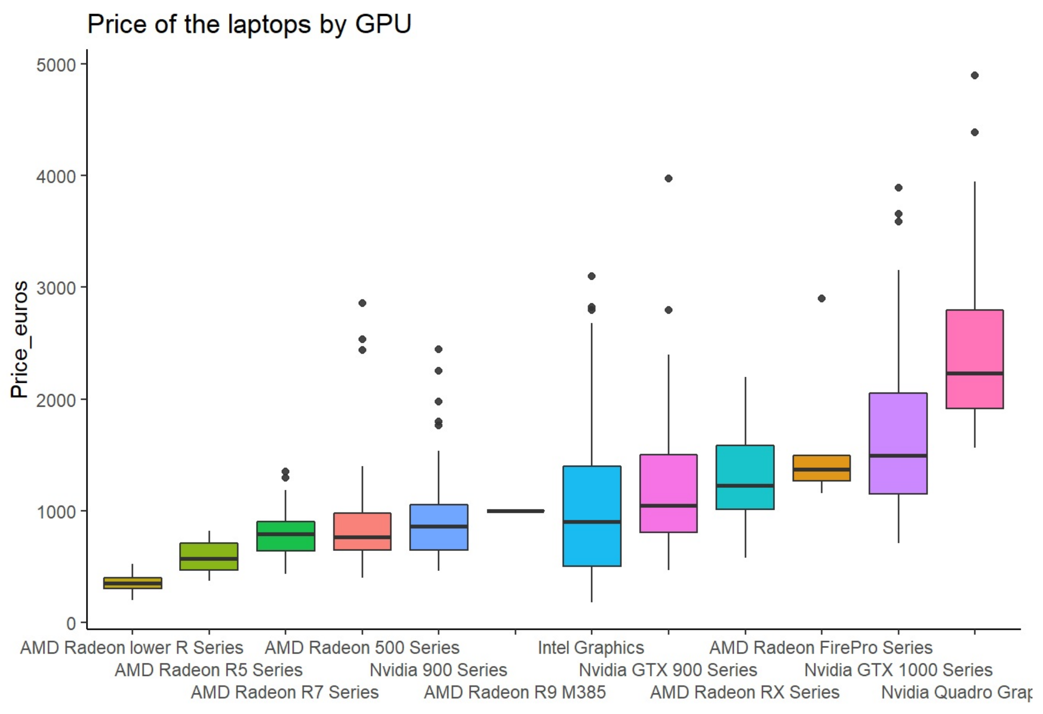
The FlashS Graph won’t show the range of memory in GB on the X-axis; this should run from 0 to 500. We can see the same behaviour as in HDD and conclude that laptops with low or no flash storage have HDD and SDD hence more expensive.

I can conclude that SSD has enough evidence by itself to assume that an increase in memory also increases the price making it a solid independent variable. On the other hand, HDD and Flash Storage show a price decrease when memory increases, but this is due to both of these variables being dependant on the other memory types that could be simultaneously found in that same laptop device.

GPU

The laptop’s GPU contains too many qualitative levels (110); It will be simplified by grouping the graphics cards by brand and series, e.g.(AMD 500s, Nvidia 900s).

```
# Transofrms the column to character to enable regex
GPU = as.character(laptops$Gpu)

# The following Regex code changes the format of Intel GPUs
GPU[grepl("^Intel", GPU)] = "Intel Graphics"

# The following Regex code changes the format of Nvidia GPUs
GPU[grepl("^Nvidia Quadro", GPU)] = "Nvidia Quadro Graphics"
GPU[grepl("Nvidia GeForce 9\\d{1,2}\\s?\\w{1,2}?", GPU)] = "Nvidia 900 Series"
GPU[grepl("Nvidia GeForce GTX\\s?\\d{4}", GPU)] = "Nvidia GTX 1000 Series"
GPU[grepl("GTX\\s?(9\\d{2})|(MX)", GPU)] = "Nvidia GTX 900 Series"

# The following Regex code changes the format of AMD GPUs
GPU[grepl("R5", GPU)] = "AMD Radeon R5 Series"
GPU[grepl("R7", GPU)] = "AMD Radeon R7 Series"
GPU[grepl("AMD Radeon 5\\d{2}", GPU)] = "AMD Radeon 500 Series"
GPU[grepl("RX", GPU)] = "AMD Radeon RX Series"
GPU[grepl("FirePro", GPU)] = "AMD Radeon FirePro Series"
GPU[grepl("(R3)|(R2)|(R4)", GPU)] = "AMD Radeon lower R Series"
GPU[grepl("AMD Radeon Pro \\d{3}", GPU)] = "AMD Radeon 500 Series"
GPU[grepl("AMD R17M-M1-70", GPU)] = "AMD Radeon 500 Series"

# Remove unknown text
GPU = gsub("ARM Mali T860 MP4", NA, GPU)
laptops$GPU = as.factor(GPU)
laptops = na.omit(laptops)

#Drops the Gpu column
laptops$Gpu = NULL

# Bar chart to explore the price of the laptops by Graphics Card
ggplot(laptops, aes(x=reorder(GPU, Price_euros), y=Price_euros, fill=GPU)) + ggtitle('Price of the laptops by GPU
')+ xlab("") + geom_boxplot(alpha=0.9) + theme_classic() + theme(legend.position="none") + scale_x_discrete(guide
= guide_axis(n.dodge=3))
```



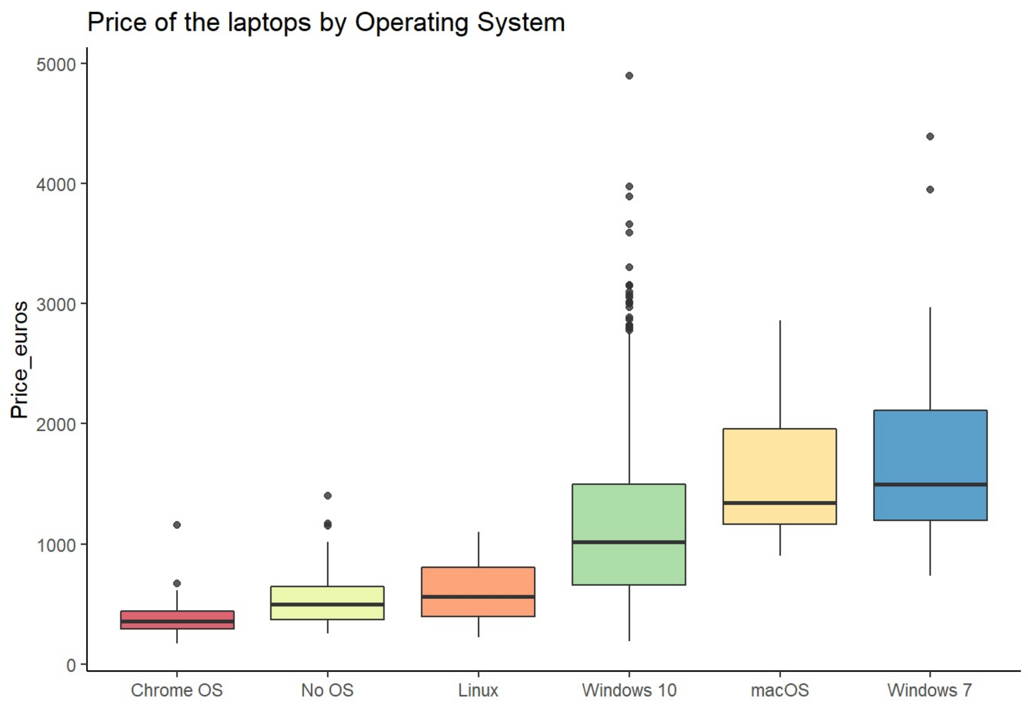
The laptop GPU results are as expected; it shows that laptops with lower-end GPU have a lower price, and as the CPU evolves, e.g.(Nvidia 900, Nvidia GTX 900, Nvidia GTX 1000), the price of the laptop is higher.

I can conclude that there is a strong correlation when it comes to GPU and laptops price.

OpSys

If there is any OpSys with a small number of observations, I might decide to remove it.

```
# Transforms column into character for regex operation
OpSys = as.character(laptops$OpSys)
# Groups values which have little observations.
OpSys[grepl("Android", OpSys)] = "Linux"
OpSys[grepl("Mac", OpSys)] = "macOS"
OpSys[grepl("Windows 10", OpSys)] = "Windows 10"
# Transform the values back into factors
laptops$OpSys = as.factor(OpSys)

# Bar chart to explore the price of the laptops by Operating System
ggplot(laptops, aes(x=reorder(OpSys, Price_euros), y=Price_euros, fill=OpSys)) + ggtitle('Price of the laptops by
Operating System')+ xlab("") + geom_boxplot(alpha=0.8) + theme_classic() + theme(legend.position="none") + scale_
fill_brewer(palette="Spectral")
```



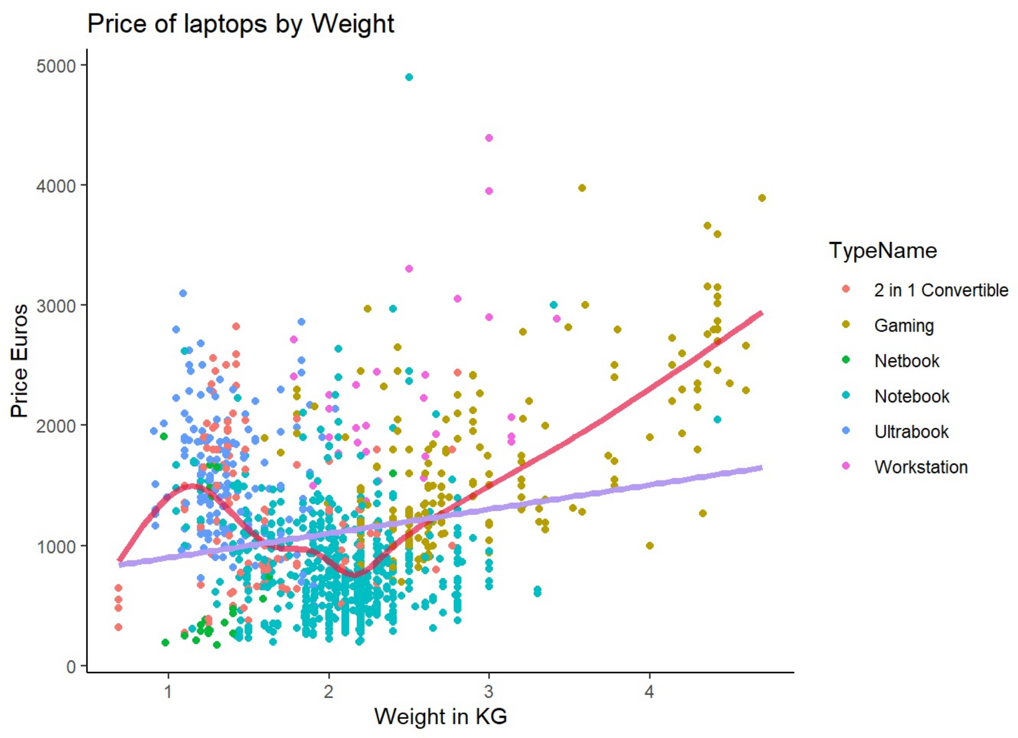
The first three variables show to have the smaller price, this makes sense because 1. Chrome OS is based mainly on Chromebooks, a new generation of laptops that connects and uses the hardware from a google server, meaning that this laptop does not possess costly hardware. 2. Linux is a free operating system which makes it cheaper. 3. Computers with no operating system are cheaper than the ones that come with one.

Computers with operating systems that need to be purchased such as windows and mac are more expensive which is to be expected.

Weight

The Weight column will be converted to a qualitative variable by removing the “kg” in each observation.

```
# Removes the kg from each row and leaves a numeric value
laptops$Weight = as.numeric(gsub("kg", "", laptops$Weight))

# qqplot showcasing the distribution of laptops price by Weight filled with TypeName
ggplot(laptops, aes(x=Weight, y= Price_euros)) + geom_point(aes(col=TypeName), size=1.5, shape=19)+
ggtitle('Price of laptops by Weight') + xlab('Weight in KG') + ylab('Price Euros')+ theme_classic()+ stat_smooth(
geom='line', alpha=0.7, se=FALSE, col='#EA1744', size=1.5) + geom_smooth(method=lm, se=FALSE, col='#B69BF5', size
=1.5)
```



The results show a weak positive relationship; this means the higher the weight, the higher the laptops price. It is interesting to see that most Ultrabooks are on the top left side of the graph, creating a curve on the non-standard line; this would suggest that even though these laptops have costly hardware, they are not heavy because the material end technology used is also focused on giving the customer a light and portable device hence making the laptop even more expensive. This is not the case for gaming devices; these laptops also have costly hardware but are not meant for customers with light device requirements.

I can conclude that weight is not a very good conductive of laptop price due to the expensive requirements of owning a light and powerful laptop. When it comes to more general materials and technologies, we see an increase in price when weight increases. Based on this dataset, I can conclude that weight does have a slight positive correlation with price.

Renaming and rearrenging columns

I am aware that renaming columns should be one of the first steps, but that would mean renaming all variables alongside the project. Due to time constraint and on this occasion, I will keep the renaming in this section.

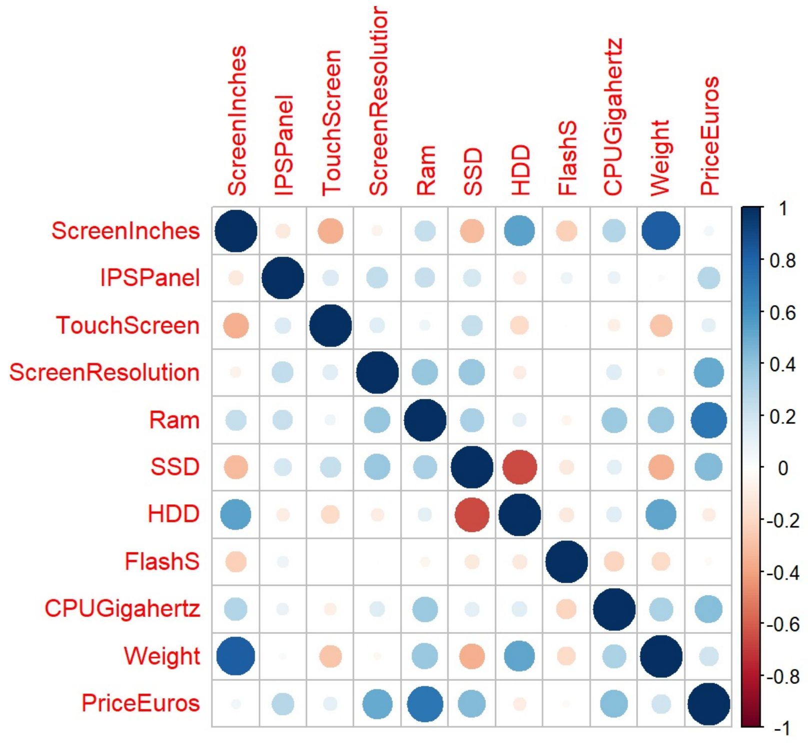
```
names(laptops)[names(laptops) == "Inches"] = "ScreenInches"
names(laptops)[names(laptops) == "Price_euros"] = "PriceEuros"
names(laptops)[names(laptops) == "TypeName"] = "Category"

laptopsDS = laptops[, c(1,2,3,9,10,4,5,13,14,15,12,11,16,6,7,8)]
head(laptopsDS, 1)
```

```
##   Company  Category ScreenInches IPSPanel TouchScreen ScreenResolution Ram SSD
## 1   Apple Ultrabook         13.3        1           0          4096000   8 128
##   HDD FlashS           CPU CPUGigahertz            GPU OpSys Weight PriceEuros
## 1   0      0 Intel Core i5          2.3 Intel Graphics macOS   1.37    1339.69
```

Numeric correlation plot

This chart summarises all the numeric analysis done in the prior sections. It also shows the strength of the correlation between the independent variables and the dependant variable “PriceEuros”.

```
# Creates a copy of the dataset which contains only the numeric variables
quantiLaptopsDS = laptopsDS[, c(3,4,5,6,7,8,9,10,12,15,16)]
# Creates a plot which shows the correlation of numeric variables.
corrplot(cor(quantiLaptopsDS))
```



It is interesting to see that Ram is the most significant numeric factor in price increase, followed by screen resolution, SSD and CPUGigahertz. We can also see a slight correlation with IPS displays, TouchScreen and Weight. It is interesting to have a hierarchy of the features that most influence the laptops price.

# Supervised Learning Experiment

This section will look into training a linear model with the updated dataset to examine the significance of each variable. Depending on the results creating a new dataset containing only significant variables will be considered.

Next section experiments with subsetting in order to rank the variables from most to least significant. Depending on the results creating a new dataset containing only the most influential variable will be considered.

For model training, I have selected 10 fold cross-validation no repeats. Models selected are “Linear Model” and “Super Vector Machine”. A model will be trained and further evaluated in the last section using RMSE and R-squared for every dataset.

Multiple Linear Regression

This section summarizes a linear model trained on the full dataset to analyse results and review significant variables.

```
# Trains a linear model
laptopsDS.lm.all = lm(PriceEuros~., data = laptopsDS)
# Prints the summary of the linear model
summary(laptopsDS.lm.all)
```

```
##
## Call:
## lm(formula = PriceEuros ~ ., data = laptopsDS)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1277.92  -176.20   -23.03   132.86  1544.26
##
## Coefficients: (1 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.443e+02  2.507e+02   0.576 0.564984
## CompanyApple                 2.900e+02  1.154e+02   2.514 0.012077 *
## CompanyAsus                  1.177e+02  4.301e+01   2.736 0.006313 **
## CompanyDell                  1.235e+02  4.013e+01   3.078 0.002134 **
## CompanyHP                    1.884e+02  3.974e+01   4.741 2.39e-06 ***
## CompanyLenovo                1.271e+02  3.935e+01   3.230 0.001270 **
## CompanyMSI                   3.888e+02  6.387e+01   6.088 1.54e-09 ***
## CompanyToshiba               2.740e+02  5.790e+01   4.733 2.48e-06 ***
## CategoryGaming              -2.472e+02  7.682e+01  -3.217 0.001329 **
## CategoryNetbook             -1.400e+02  8.460e+01  -1.654 0.098308 .
## CategoryNotebook            -2.334e+02  5.425e+01  -4.303 1.83e-05 ***
## CategoryUltrabook            9.596e+01  5.223e+01   1.837 0.066424 .
## CategoryWorkstation         -9.171e+01  1.172e+02  -0.782 0.434161
## ScreenInches                -5.369e+01  1.468e+01  -3.658 0.000265 ***
## IPSPanel                     3.368e+01  2.365e+01   1.424 0.154597
## TouchScreen                 -6.209e+01  4.490e+01  -1.383 0.166999
## ScreenResolution             6.520e-05  6.814e-06   9.567  < 2e-16 ***
## Ram                          4.845e+01  2.877e+00  16.839  < 2e-16 ***
## SSD                          3.720e-01  9.183e-02   4.051 5.43e-05 ***
## HDD                         -1.741e-02  2.944e-02  -0.591 0.554429
## FlashS                       6.300e-03  3.788e-01   0.017 0.986733
## CPUAMD A12-Series           -1.228e+02  1.690e+02  -0.727 0.467445
## CPUAMD A4-Series            -6.384e+01  4.097e+02  -0.156 0.876198
## CPUAMD A6-Series            -9.074e+01  2.533e+02  -0.358 0.720183
## CPUAMD A8-Series             5.265e+00  2.040e+02   0.026 0.979411
## CPUAMD A9-Series            -5.788e+01  1.530e+02  -0.378 0.705235
## CPUAMD E-Series              3.639e+01  2.918e+02   0.125 0.900784
## CPUAMD FX                   -2.366e+02  3.497e+02  -0.676 0.498862
## CPUAMD Ryzen                 4.480e+02  2.290e+02   1.956 0.050693 .
## CPUIntel Atom               -2.867e+01  2.011e+02  -0.143 0.886646
## CPUIntel Celeron Dual Core   2.586e+01  1.437e+02   0.180 0.857264
## CPUIntel Celeron Quad Core   2.742e+01  1.969e+02   0.139 0.889266
## CPUIntel Core i3             6.934e+01  1.337e+02   0.519 0.604163
## CPUIntel Core i5             2.286e+02  1.318e+02   1.734 0.083105 .
## CPUIntel Core i7             3.168e+02  1.314e+02   2.411 0.016082 *
## CPUIntel Core M              3.405e+02  1.595e+02   2.135 0.032936 *
## CPUIntel Pentium Dual Core  -3.690e+01  2.253e+02  -0.164 0.869913
## CPUIntel Pentium Quad Core   8.250e+01  1.495e+02   0.552 0.581200
## CPUIntel Xeon                1.045e+03  2.111e+02   4.950 8.50e-07 ***
## CPUGigahertz                 1.403e+02  3.001e+01   4.674 3.29e-06 ***
## GPUAMD Radeon FirePro Series  4.402e+02  1.586e+02   2.776 0.005590 **
## GPUAMD Radeon lower R Series  1.715e+01  2.386e+02   0.072 0.942693
## GPUAMD Radeon R5 Series      6.782e+00  6.286e+01   0.108 0.914107
## GPUAMD Radeon R7 Series     -1.288e+02  7.804e+01  -1.650 0.099205 .
## GPUAMD Radeon R9 M385        1.710e+02  4.524e+02   0.378 0.705421
## GPUAMD Radeon RX Series      1.876e+02  1.250e+02   1.500 0.133775
## GPUIntel Graphics            1.218e+02  4.952e+01   2.460 0.014033 *
## GPUNvidia 900 Series        -5.947e+00  5.570e+01  -0.107 0.914994
## GPUNvidia GTX 1000 Series    2.154e+02  6.921e+01   3.113 0.001899 **
## GPUNvidia GTX 900 Series    -5.155e+01  6.290e+01  -0.820 0.412635
## GPUNvidia Quadro Graphics    6.847e+02  1.139e+02   6.011 2.45e-09 ***
## OpSysLinux                  -8.505e+01  8.678e+01  -0.980 0.327283
## OpSysmacOS                          NA         NA      NA       NA
## OpSysNo OS                  -1.676e+02  8.892e+01  -1.885 0.059720 .
## OpSysWindows 10              5.326e+01  7.812e+01   0.682 0.495486
## OpSysWindows 7               3.512e+02  9.283e+01   3.783 0.000163 ***
## Weight                       2.413e+02  3.299e+01   7.315 4.71e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 304.5 on 1185 degrees of freedom
## Multiple R-squared:  0.8005, Adjusted R-squared:  0.7912
## F-statistic: 86.43 on 55 and 1185 DF,  p-value: < 2.2e-16
```

```
#Prints prediction error rate
percentErr = c()
percentErr = (100*summary(laptopsDS.lm.all)$sigma/mean(laptopsDS$PriceEuros))
cat("Percentage error = ",percentErr[1], "%")
```

```
## Percentage error =  27.37126 %
```

The residuals show the distance from the data to the fitted line. We can see that the min(1277.92) and the max(1544.26) are symmetrically distributed around the line, meaning both have approximately the same distance from 0. Likewise, 1Q(-176) and 3Q(132) also show an approximate distance from 0. We can also see the median(-23.03) is somewhat close to 0.

We can also analyse the contribution of each variable in our model. Below I will conclude which variables reject the null hypothesis meaning the P-values are smaller than 0.05, i.e. there is a statistically significant relationship between the variable and the target.

H0: there is no relationship between this variable and PriceEuros.

H1: There is some relationship between this variable and PriceEuros.

The following variables reject the null hypothesis. I have also included categorical variables which contain levels that reject the null hypothesis. Company, Category, ScreenInches, ScreenResolution, Ram, SSD, CPU, CPUGigahertz, GPU, Opsystem, Weight.

The following variables do not have enough evidence to reject the null hypothesis. IPSPanel, TouchScreen, FlashS, HDD.

Here four variables do not make a significant contribution to the prediction of the price in euros. Later I will try to remove those variables from our predictors to see if it affects the goodness of fit of our model.

We have obtained an RSE of 304.5, which results in a Percentage error of 27.37%, which is high.

We can see now that the R-square = 0.8005. It means that the combined linear regression explains 80.05% of the variance in Price on all predictors.

F-Statistics also shows a p-value < 0.05 (2.2e-16) which is highly significant.

Subset Selection

The exhaustive search in subset helps us find the best combination of variables as well as the most influencial for predicting laptop's price.

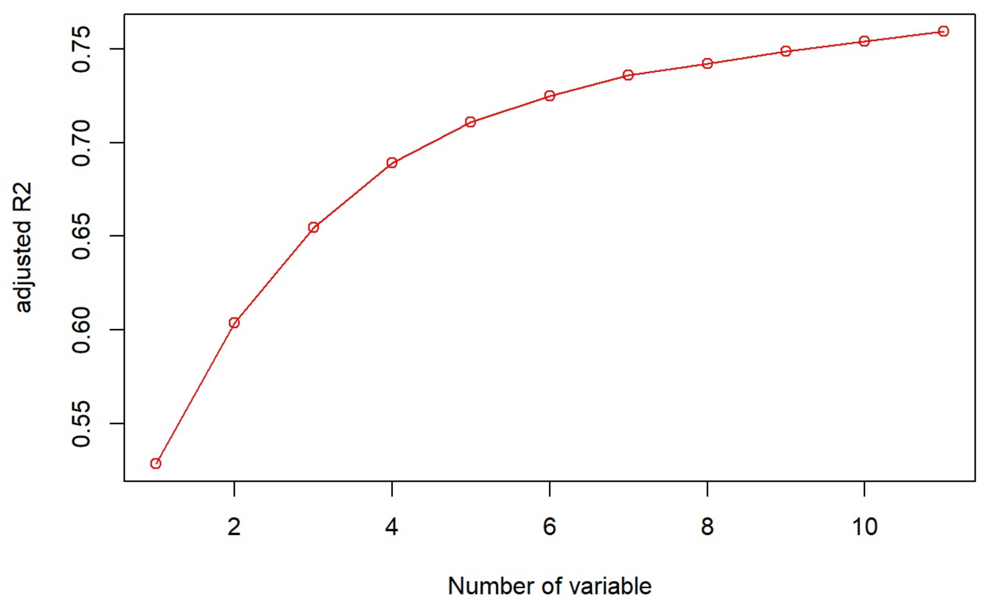
```
# Creates a exhaustive subset of the dataset
regfit.full.exhaustive <- regsubsets(PriceEuros ~ .,data = laptopsDS,
method = "exhaustive", really.big=T, nvmax = 10)
```

```
## Reordering variables and trying again:
```

```
subset.summary.exhaustive <- summary(regfit.full.exhaustive)
#
#subset.summary.exhaustive$outmat
sigVariables = c("RAM", "categoryNotebook",  "GPUNvidia Quadro Graphics", "ScreenResolution", "CPUGigahertz", "Op
SysWindows 7", "OpSysWindows 10", "CPUIntel Core i5", "CPUIntel Core i7", "Intel Graphics" ,"CPUIntel Xeon")
rank = c(1:11)
sigVariablesRank = data.frame(rank, sigVariables)
sigVariablesRank
```

```
##    rank              sigVariables
## 1     1                       RAM
## 2     2          categoryNotebook
## 3     3 GPUNvidia Quadro Graphics
## 4     4          ScreenResolution
## 5     5              CPUGigahertz
## 6     6             OpSysWindows 7
## 7     7            OpSysWindows 10
## 8     8          CPUIntel Core i5
## 9     9          CPUIntel Core i7
## 10   10            Intel Graphics
## 11   11             CPUIntel Xeon
```

```
# Graph for adjusted R-square visualization.
plot(subset.summary.exhaustive$adjr2,type = "o",col = "red",ylab = "adjusted R2",xlab = "Number of variable")
```



I have listed the rank from the most significant to the least significant (results have been manually added to a table and outmat commented out due to original result being too long). We can see RAM popping up as the most significant result. We can also see the elbow generated in our graph, after the 7th value there is fewer variable significance and could consider removing those variable to evaluate a linear model.

Creating different models

In this section, I will create two new datasets that remove all the values that had no enough evidence to accept the alternative hypothesis and another model that only includes the top 7 variables shown in our subset experiment.

```
# Removing variables with no significant value
laptopsDSSignOnly = laptopsDS
laptopsDSSignOnly$HDD = NULL
laptopsDSSignOnly$FlashS = NULL
laptopsDSSignOnly$IPSPanel = NULL
laptopsDSSignOnly$TouchScreen = NULL
# Displaying header results
head(laptopsDSSignOnly, 1)
```

```
##   Company  Category ScreenInches ScreenResolution Ram SSD           CPU
## 1   Apple Ultrabook         13.3          4096000   8 128 Intel Core i5
##   CPUGigahertz            GPU OpSys Weight PriceEuros
## 1          2.3 Intel Graphics macOS   1.37    1339.69
```

```
# Removing all variables which are not part of the subset
laptopsDSSubset = laptopsDSSignOnly
laptopsDSSubset$Weight = NULL
laptopsDSSubset$ScreenInches = NULL
laptopsDSSubset$SSD = NULL
laptopsDSSubset$Company = NULL
# Displaying header results
head(laptopsDSSubset, 1)
```

```
##    Category ScreenResolution Ram           CPU CPUGigahertz            GPU
## 1 Ultrabook          4096000   8 Intel Core i5          2.3 Intel Graphics
##   OpSys PriceEuros
## 1 macOS    1339.69
```

Linear Model

The three datasets have been created and used to train the linear models. Each model has been applied a 10kfold cross-validation. No tuning applied for a linear model.

```r
# Setting random seed for reproducible results
set.seed(2)
# Setting train control values
control = trainControl(method = "cv", number = 10)

# Training and testing of laptopDS with lm
lmPolyOriginal <- train(PriceEuros ~., data = laptopsDS, method = "lm",
                 trControl=control)
# Training and testing of laptopsDSSignOnly with lm
lmPolySignificant <- train(PriceEuros ~., data = laptopsDSSignOnly, method = "lm",
                 trControl=control)
# Training and testing of laptopsDSSubset with lm
lmPolySubset <- train(PriceEuros ~., data = laptopsDSSubset, method = "lm",
                 trControl=control)
```

SVM

The three datasets have been used to train an SVM model each. Each model has been applied a 10K fold cross-validation. I have trained each model with a tuneLength of 4; all models gave the same optimal hyperparameters (degree 1, scale 1 and C 2). Note(This section was commented out to avoid printing the long results). The SVM method used is symPoly; other SVM such as (svmLinear3, svmLinear, svmRadial) were used, but symPoly obtained the best results.

```r
# Setting random seed for reproducible results
set.seed(2)
# Setting train control values
control = trainControl(method = "cv", number = 10)

#RMSE was used to select the optimal model using the smallest value.
#The final values used for the model were degree = 1, scale = 1 and C = 2.
# Setting best hyperparamers for SVM model.
hyperparams = expand.grid(degree=1, scale=1, C=2)
# svmPolyOriginal <- train(PriceEuros ~., data = laptopsDS, method = "svmPoly",
#                  trControl=control,
#                  preProcess = c("center", "scale"),
#                  tuneLength = 4)
#
# svmPolySignificant <- train(PriceEuros ~., data = laptopsDSSignOnly, method = "svmPoly",
#                  trControl=control,
#                  preProcess = c("center", "scale"),
#                  tuneLength = 4)
#
# svmPolySubset <- train(PriceEuros ~., data = laptopsDSSubset, method = "svmPoly",
#                  trControl=control,
#                  preProcess = c("center", "scale"),
#                  tuneLength = 4)



# Training and testing of laptopsDS with SVM
svmPolyOriginal <- train(PriceEuros ~., data = laptopsDS, method = "svmPoly",
                 trControl=control,
                 preProcess = c("center", "scale"),
                 tuneGrid = hyperparams)
# Training and testing of laptopsDSSignOnly with SVM
svmPolySignificant <- train(PriceEuros ~., data = laptopsDSSignOnly, method = "svmPoly",
                 trControl=control,
                 preProcess = c("center", "scale"),
                 tuneGrid = hyperparams)
# Training and testing of laptopsDSSubset with SVM
svmPolySubset <- train(PriceEuros ~., data = laptopsDSSubset, method = "svmPoly",
                 trControl=control,
                 preProcess = c("center", "scale"),
                 tuneGrid = hyperparams)
```

Evaluation

In this section all the results determining the model’s goodness of fit have been aggregated. The following evaluation metrics have been selected. R-squared: explain how well the model fits the data. Residual Standard Error: Explains how much the response deviates from the regression line.

```r
# Adding all RMSE results to vector
predicteErrors = (c(lmPolyOriginal$results$RMSE,lmPolySignificant$results$RMSE,lmPolySubset$results$RMSE,svmPolyO
riginal$results$RMSE,svmPolySignificant$results$RMSE,svmPolySubset$results$RMSE))
# Adding all R2 results to vector
predicteAcc = c(lmPolyOriginal$results$Rsquared, lmPolySignificant$results$Rsquared, lmPolySubset$results$Rsquare
d, svmPolyOriginal$results$Rsquared, svmPolySignificant$results$Rsquared, svmPolySubset$results$Rsquared)

# Creating table headings
header = c("Model/Dataset", "RMSE", "Rsquared")
# Creating model/dataset column
modelName = c("LM/Original", "LM/Significant", "LM/Subset", "SVM/Original", "SVM/Significant", "SVM/Subset")

#Creating dataframe with results
finalResults = data.frame(modelName, predicteErrors, predicteAcc)
#Adding headers to df
names(finalResults) = header
# Sorting the results with RMSE by ascending order
finalResults = finalResults[order(finalResults$RMSE),]
finalResults
```

```
##     Model/Dataset     RMSE  Rsquared
## 1     LM/Original 311.1869 0.7804555
## 2  LM/Significant 314.2462 0.7737839
## 5 SVM/Significant 317.4783 0.7732472
## 4    SVM/Original 320.2940 0.7722763
## 3       LM/Subset 321.6436 0.7716971
## 6      SVM/Subset 330.8526 0.7671325
```

Conclusions

The results provided by our various models and datasets does not seem to vary much. There is only a 1.3% difference in the goodness of fit between the least and best model. RMSE does not change much either, I am inclined to conclude that no model or dataset was considerably more advantageous than the other.

Overall, the best model, “LM/Original,” fits 78% of the data; this is not an outstanding result, but it is a decent one. RMSE seems to deviate 311 euros from the original price of a laptop. Considering that the price ranges from 300 to 5000 I consider this a decent result but far for practical use.

It is interesting to see a difference of 00.9% between the original and subset dataset for LM and 00.5% in SVM, considering that the original dataset contains 15 columns and subset one contains 7, which is half. It is common knowledge in data science that simpler models are better, but at which cost?

Following on the previous topic and how seven factors can determine the price of a laptop as much as 15 brings into the light the most important variables when considering a laptop price. This report concludes that the most influential factors in computer price are RAM, Category, Graphics Card, Screen Resolution, CPU, Operating system and SSD memory in that order. It is surprising to see Screen Resolution as a top influence for price and cannot explain why CPU is not more relevant. Initially, I would have expected Graphics Card, CPU, RAM and Memory to be the top influencers in that order; A reason being is the vast amount of different levels that all of these laptop specifications contain, I would like to experiment with a dataset of more samples to shape a more robust model. In contrast, there were many specifications that appeared only in a few samples, and the model was unable to pick on these features. If a large dataset could be obtained, I would also like to explore allowing all CPU and GPU levels instead of grouping them up. Finally, other models could be used to try to achieve better goodness of fit.