Laptop Price Prediction

Introduction:

In our project, we aimed to develop a predictive model for laptop price estimation. By using machine learning techniques, we analyzed laptop features to make accurate price predictions.

We explored different regression-based models, including multi-linear regression, SVR, decision tree, random forest, and XGBoost. Our goal was to understand how factors like processor speed, RAM, storage capacity, brand, and screen size influence laptop prices.

The outcome of our project has practical implications for consumers and industry professionals. By identifying the most accurate model, we can provide insights into the drivers of laptop prices. This helps consumers make informed decisions, aids retailers in pricing strategies, and assists manufacturers in optimizing laptop pricing.

Through our research, we contribute to the field of laptop price prediction and provide guidance to industry stakeholders. Our project showcases the power of machine learning in understanding laptop pricing trends.

Overall, as students, our project allows us to explore data analysis and machine learning while addressing a real-world problem. We are excited to share our findings and contribute to the knowledge of laptop pricing

```
library(tidyverse)
```

Import libaries

```
## Warning: package 'tidyverse' was built under R version 4.2.2

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.2

## Warning: package 'readr' was built under R version 4.2.2

## Warning: package 'purrr' was built under R version 4.2.2

## Warning: package 'dplyr' was built under R version 4.2.2

## Warning: package 'stringr' was built under R version 4.2.2

## Warning: package 'forcats' was built under R version 4.2.2
```

```
## Warning: package 'lubridate' was built under R version 4.2.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.0     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(stringr)
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.2.3
```

```
## corrplot 0.92 loaded
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.2.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.2.3
```

```
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.2.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.2.3
```

```
library(rpart)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.3

## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.2.3

##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.2.3

##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
data = read.csv("laptopData.csv")
```

Import dataset

```r
head(data)
```

Learning about data insights

```
##   Unnamed..0 Company  TypeName Inches                    ScreenResolution
## 1          0   Apple Ultrabook   13.3 IPS Panel Retina Display 2560x1600
## 2          1   Apple Ultrabook   13.3                           1440x900
## 3          2      HP  Notebook   15.6                  Full HD 1920x1080
## 4          3   Apple Ultrabook   15.4 IPS Panel Retina Display 2880x1800
## 5          4   Apple Ultrabook   13.3 IPS Panel Retina Display 2560x1600
## 6          5    Acer  Notebook   15.6                           1366x768
##                          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
## 5       Intel Core i5 3.1GHz  8GB           256GB SSD
## 6    AMD A9-Series 9420 3GHz  4GB           500GB HDD
##                            Gpu      OpSys Weight     Price
## 1 Intel Iris Plus Graphics 640      macOS 1.37kg  71378.68
## 2       Intel HD Graphics 6000      macOS 1.34kg  47895.52
## 3        Intel HD Graphics 620      No OS 1.86kg  30636.00
## 4           AMD Radeon Pro 455      macOS 1.83kg 135195.34
## 5 Intel Iris Plus Graphics 650      macOS 1.37kg  96095.81
## 6               AMD Radeon R5 Windows 10  2.1kg  21312.00
```

```r
summary(data)
```

```
##    Unnamed..0       Company            TypeName            Inches
##  Min.   :   0.0   Length:1303        Length:1303        Length:1303
##  1st Qu.: 327.0   Class :character   Class :character   Class :character
##  Median : 652.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 652.7
##  3rd Qu.: 980.0
##  Max.   :1302.0
##  NA's   :30
##  ScreenResolution       Cpu                Ram               Memory
##  Length:1303        Length:1303        Length:1303        Length:1303
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
```

```
##      Gpu                OpSys              Weight              Price
##  Length:1303        Length:1303        Length:1303        Min.   :  9271
##  Class :character   Class :character   Class :character   1st Qu.: 31915
##  Mode  :character   Mode  :character   Mode  :character   Median : 52161
##                                                           Mean   : 59956
##                                                           3rd Qu.: 79333
##                                                           Max.   :324955
##                                                           NA's   :30
```

```
str(data)
```

```
## 'data.frame':    1303 obs. of  12 variables:
##  $ Unnamed..0      : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Company         : chr  "Apple" "Apple" "HP" "Apple" ...
##  $ TypeName        : chr  "Ultrabook" "Ultrabook" "Notebook" "Ultrabook" ...
##  $ Inches          : chr  "13.3" "13.3" "15.6" "15.4" ...
##  $ ScreenResolution: chr  "IPS Panel Retina Display 2560x1600" "1440x900" "Full HD 1920x1080" "IPS Pa
##  $ Cpu             : chr  "Intel Core i5 2.3GHz" "Intel Core i5 1.8GHz" "Intel Core i5 7200U 2.5GHz"
##  $ Ram             : chr  "8GB" "8GB" "8GB" "16GB" ...
##  $ Memory          : chr  "128GB SSD" "128GB Flash Storage" "256GB SSD" "512GB SSD" ...
##  $ Gpu             : chr  "Intel Iris Plus Graphics 640" "Intel HD Graphics 6000" "Intel HD Graphics
##  $ OpSys           : chr  "macOS" "macOS" "No OS" "macOS" ...
##  $ Weight          : chr  "1.37kg" "1.34kg" "1.86kg" "1.83kg" ...
##  $ Price           : num  71379 47896 30636 135195 96096 ...
```

```
null_values = is.na(data)
table(null_values)
```

check for null values

```
## null_values
## FALSE  TRUE
## 15576    60
```

- Removing unnamed column

```
dataset = data[ ,-1]
```

```
null_counts = colSums(is.na(dataset))
print(null_counts)
```

Removing null values

```
##          Company         TypeName           Inches ScreenResolution
##                0                0                0                0
##              Cpu              Ram           Memory              Gpu
##                0                0                0                0
##            OpSys           Weight            Price
##                0                0               30
```

```r
dataset= drop_na(dataset)
```

```r
dataset = distinct(dataset)
```

duplicate values

```r
d = sapply(colnames(dataset), function(col) unique(dataset[[col]]))
print(d)
```

printing the unique values for all the columns

```
## $Company
##  [1] "Apple"     "HP"        "Acer"      "Asus"      "Dell"      "Lenovo"
##  [7] "Chuwi"     "MSI"       "Microsoft" "Toshiba"   "Huawei"    "Xiaomi"
## [13] "Vero"      "Razer"     "Mediacom"  "Samsung"   "Google"    "Fujitsu"
## [19] "LG"
##
## $TypeName
## [1] "Ultrabook"          "Notebook"           "Gaming"
## [4] "2 in 1 Convertible" "Workstation"        "Netbook"
##
## $Inches
##  [1] "13.3" "15.6" "15.4" "14"   "12"   "17.3" "13.5" "12.5" "13"   "18.4"
## [11] "13.9" "11.6" "25.6" "35.6" "12.3" "27.3" "24"   "33.5" "?"    "31.6"
## [21] "17"   "15"   "14.1" "11.3" "10.1"
##
## $ScreenResolution
##  [1] "IPS Panel Retina Display 2560x1600"
##  [2] "1440x900"
##  [3] "Full HD 1920x1080"
##  [4] "IPS Panel Retina Display 2880x1800"
##  [5] "1366x768"
##  [6] "IPS Panel Full HD 1920x1080"
##  [7] "IPS Panel Retina Display 2304x1440"
##  [8] "IPS Panel Full HD / Touchscreen 1920x1080"
##  [9] "Full HD / Touchscreen 1920x1080"
## [10] "Touchscreen / Quad HD+ 3200x1800"
## [11] "Touchscreen 2256x1504"
## [12] "Quad HD+ / Touchscreen 3200x1800"
## [13] "IPS Panel 1366x768"
## [14] "IPS Panel 4K Ultra HD / Touchscreen 3840x2160"
## [15] "IPS Panel Full HD 2160x1440"
## [16] "4K Ultra HD / Touchscreen 3840x2160"
## [17] "1600x900"
## [18] "IPS Panel 4K Ultra HD 3840x2160"
## [19] "4K Ultra HD 3840x2160"
## [20] "Touchscreen 1366x768"
```

```
## [21] "Touchscreen 2560x1440"
## [22] "IPS Panel Full HD 1366x768"
## [23] "IPS Panel 2560x1440"
## [24] "IPS Panel Full HD 2560x1440"
## [25] "IPS Panel Retina Display 2736x1824"
## [26] "Touchscreen 2400x1600"
## [27] "2560x1440"
## [28] "IPS Panel Quad HD+ 2560x1440"
## [29] "IPS Panel Quad HD+ 3200x1800"
## [30] "IPS Panel Quad HD+ / Touchscreen 3200x1800"
## [31] "IPS Panel Touchscreen 1366x768"
## [32] "1920x1080"
## [33] "IPS Panel Full HD 1920x1200"
## [34] "IPS Panel Touchscreen / 4K Ultra HD 3840x2160"
## [35] "IPS Panel Touchscreen 2560x1440"
## [36] "Touchscreen / Full HD 1920x1080"
## [37] "Quad HD+ 3200x1800"
## [38] "IPS Panel Touchscreen 1920x1200"
## [39] "Touchscreen / 4K Ultra HD 3840x2160"
## [40] "IPS Panel Touchscreen 2400x1600"
##
## $Cpu
##   [1] "Intel Core i5 2.3GHz"
##   [2] "Intel Core i5 1.8GHz"
##   [3] "Intel Core i5 7200U 2.5GHz"
##   [4] "Intel Core i7 2.7GHz"
##   [5] "Intel Core i5 3.1GHz"
##   [6] "AMD A9-Series 9420 3GHz"
##   [7] "Intel Core i7 2.2GHz"
##   [8] "Intel Core i7 8550U 1.8GHz"
##   [9] "Intel Core i5 8250U 1.6GHz"
##  [10] "Intel Core i3 6006U 2GHz"
##  [11] "Intel Core i7 2.8GHz"
##  [12] "Intel Core M m3 1.2GHz"
##  [13] "Intel Core i7 7500U 2.7GHz"
##  [14] "Intel Core i7 2.9GHz"
##  [15] "Intel Core i3 7100U 2.4GHz"
##  [16] "Intel Core i5 7300HQ 2.5GHz"
##  [17] "AMD E-Series E2-9000e 1.5GHz"
##  [18] "Intel Core i5 1.6GHz"
##  [19] "Intel Core i7 8650U 1.9GHz"
##  [20] "Intel Atom x5-Z8300 1.44GHz"
##  [21] "AMD E-Series E2-6110 1.5GHz"
##  [22] "AMD A6-Series 9220 2.5GHz"
##  [23] "Intel Celeron Dual Core N3350 1.1GHz"
##  [24] "Intel Core i3 7130U 2.7GHz"
##  [25] "Intel Core i7 7700HQ 2.8GHz"
##  [26] "Intel Core i5 2.0GHz"
##  [27] "AMD Ryzen 1700 3GHz"
##  [28] "Intel Pentium Quad Core N4200 1.1GHz"
##  [29] "Intel Celeron Dual Core N3060 1.6GHz"
##  [30] "Intel Core i5 1.3GHz"
##  [31] "AMD FX 9830P 3GHz"
##  [32] "Intel Core i7 7560U 2.4GHz"
```

```
##  [33] "AMD E-Series 6110 1.5GHz"
##  [34] "Intel Core i5 6200U 2.3GHz"
##  [35] "Intel Core M 6Y75 1.2GHz"
##  [36] "Intel Core i5 7500U 2.7GHz"
##  [37] "Intel Core i3 6006U 2.2GHz"
##  [38] "AMD A6-Series 9220 2.9GHz"
##  [39] "Intel Core i7 6920HQ 2.9GHz"
##  [40] "Intel Core i5 7Y54 1.2GHz"
##  [41] "Intel Core i7 7820HK 2.9GHz"
##  [42] "Intel Xeon E3-1505M V6 3GHz"
##  [43] "Intel Core i7 6500U 2.5GHz"
##  [44] "AMD E-Series 9000e 1.5GHz"
##  [45] "AMD A10-Series A10-9620P 2.5GHz"
##  [46] "AMD A6-Series A6-9220 2.5GHz"
##  [47] "Intel Core i5 2.9GHz"
##  [48] "Intel Core i7 6600U 2.6GHz"
##  [49] "Intel Core i3 6006U 2.0GHz"
##  [50] "Intel Celeron Dual Core 3205U 1.5GHz"
##  [51] "Intel Core i7 7820HQ 2.9GHz"
##  [52] "AMD A10-Series 9600P 2.4GHz"
##  [53] "Intel Core i7 7600U 2.8GHz"
##  [54] "AMD A8-Series 7410 2.2GHz"
##  [55] "Intel Celeron Dual Core 3855U 1.6GHz"
##  [56] "Intel Pentium Quad Core N3710 1.6GHz"
##  [57] "AMD A12-Series 9720P 2.7GHz"
##  [58] "Intel Core i5 7300U 2.6GHz"
##  [59] "AMD A12-Series 9720P 3.6GHz"
##  [60] "Intel Celeron Quad Core N3450 1.1GHz"
##  [61] "Intel Celeron Dual Core N3060 1.60GHz"
##  [62] "Intel Core i5 6440HQ 2.6GHz"
##  [63] "Intel Core i7 6820HQ 2.7GHz"
##  [64] "AMD Ryzen 1600 3.2GHz"
##  [65] "Intel Core i7 7Y75 1.3GHz"
##  [66] "Intel Core i5 7440HQ 2.8GHz"
##  [67] "Intel Core i7 7660U 2.5GHz"
##  [68] "Intel Core i7 7700HQ 2.7GHz"
##  [69] "Intel Core M m3-7Y30 2.2GHz"
##  [70] "Intel Core i5 7Y57 1.2GHz"
##  [71] "Intel Core i7 6700HQ 2.6GHz"
##  [72] "Intel Core i3 6100U 2.3GHz"
##  [73] "Intel Atom x5-Z8350 1.44GHz"
##  [74] "AMD A10-Series 9620P 2.5GHz"
##  [75] "AMD E-Series 7110 1.8GHz"
##  [76] "Intel Celeron Dual Core N3350 2.0GHz"
##  [77] "AMD A9-Series A9-9420 3GHz"
##  [78] "Intel Core i7 6820HK 2.7GHz"
##  [79] "Intel Core M 7Y30 1.0GHz"
##  [80] "Intel Xeon E3-1535M v6 3.1GHz"
##  [81] "Intel Celeron Quad Core N3160 1.6GHz"
##  [82] "Intel Core i5 6300U 2.4GHz"
##  [83] "Intel Core i3 6100U 2.1GHz"
##  [84] "AMD E-Series E2-9000 2.2GHz"
##  [85] "Intel Celeron Dual Core N3050 1.6GHz"
##  [86] "Intel Core M M3-6Y30 0.9GHz"
```

```
##  [87] "AMD A9-Series 9420 2.9GHz"
##  [88] "Intel Core i5 6300HQ 2.3GHz"
##  [89] "AMD A6-Series 7310 2GHz"
##  [90] "Intel Atom Z8350 1.92GHz"
##  [91] "Intel Xeon E3-1535M v5 2.9GHz"
##  [92] "Intel Core i5 6260U 1.8GHz"
##  [93] "Intel Pentium Dual Core N4200 1.1GHz"
##  [94] "Intel Celeron Quad Core N3710 1.6GHz"
##  [95] "Intel Core M 1.2GHz"
##  [96] "AMD A12-Series 9700P 2.5GHz"
##  [97] "Intel Core i7 7500U 2.5GHz"
##  [98] "Intel Pentium Dual Core 4405U 2.1GHz"
##  [99] "AMD A4-Series 7210 2.2GHz"
## [100] "Intel Core i7 6560U 2.2GHz"
## [101] "Intel Core M m7-6Y75 1.2GHz"
## [102] "AMD FX 8800P 2.1GHz"
## [103] "Intel Core M M7-6Y75 1.2GHz"
## [104] "Intel Core i5 7200U 2.50GHz"
## [105] "Intel Core i5 7200U 2.70GHz"
## [106] "Intel Atom X5-Z8350 1.44GHz"
## [107] "Intel Core i5 7200U 2.7GHz"
## [108] "Intel Core M 1.1GHz"
## [109] "Intel Atom x5-Z8550 1.44GHz"
## [110] "Intel Pentium Dual Core 4405Y 1.5GHz"
## [111] "Intel Pentium Quad Core N3700 1.6GHz"
## [112] "Intel Core M 6Y54 1.1GHz"
## [113] "Intel Core i7 6500U 2.50GHz"
## [114] "Intel Celeron Dual Core N3350 2GHz"
## [115] "Samsung Cortex A72&A53 2.0GHz"
## [116] "AMD E-Series 9000 2.2GHz"
## [117] "Intel Core M 6Y30 0.9GHz"
## [118] "AMD A9-Series 9410 2.9GHz"
##
## $Ram
##  [1] "8GB"  "16GB" "4GB"  "2GB"  "12GB" "64GB" "6GB"  "32GB" "24GB" "1GB"
##
## $Memory
##  [1] "128GB SSD"                     "128GB Flash Storage"
##  [3] "256GB SSD"                     "512GB SSD"
##  [5] "500GB HDD"                     "256GB Flash Storage"
##  [7] "1TB HDD"                       "128GB SSD +  1TB HDD"
##  [9] "256GB SSD +  256GB SSD"        "64GB Flash Storage"
## [11] "32GB Flash Storage"            "256GB SSD +  1TB HDD"
## [13] "256GB SSD +  2TB HDD"          "32GB SSD"
## [15] "2TB HDD"                       "64GB SSD"
## [17] "1.0TB Hybrid"                  "512GB SSD +  1TB HDD"
## [19] "1TB SSD"                       "256GB SSD +  500GB HDD"
## [21] "128GB SSD +  2TB HDD"          "512GB SSD +  512GB SSD"
## [23] "16GB SSD"                      "16GB Flash Storage"
## [25] "512GB SSD +  256GB SSD"        "512GB SSD +  2TB HDD"
## [27] "64GB Flash Storage +  1TB HDD" "180GB SSD"
## [29] "1TB HDD +  1TB HDD"            "32GB HDD"
## [31] "1TB SSD +  1TB HDD"            "?"
## [33] "512GB Flash Storage"           "128GB HDD"
```

```
## [35] "240GB SSD"                     "8GB SSD"
## [37] "508GB Hybrid"                  "1.0TB HDD"
## [39] "512GB SSD +  1.0TB Hybrid"     "256GB SSD +  1.0TB Hybrid"
##
## $Gpu
##   [1] "Intel Iris Plus Graphics 640"   "Intel HD Graphics 6000"
##   [3] "Intel HD Graphics 620"          "AMD Radeon Pro 455"
##   [5] "Intel Iris Plus Graphics 650"   "AMD Radeon R5"
##   [7] "Intel Iris Pro Graphics"        "Nvidia GeForce MX150"
##   [9] "Intel UHD Graphics 620"         "Intel HD Graphics 520"
##  [11] "AMD Radeon Pro 555"             "AMD Radeon R5 M430"
##  [13] "Intel HD Graphics 615"          "AMD Radeon Pro 560"
##  [15] "Nvidia GeForce 940MX"           "Nvidia GeForce GTX 1050"
##  [17] "AMD Radeon R2"                  "AMD Radeon 530"
##  [19] "Nvidia GeForce 930MX"           "Intel HD Graphics"
##  [21] "Intel HD Graphics 500"          "Nvidia GeForce 930MX "
##  [23] "Nvidia GeForce GTX 1060"        "Nvidia GeForce 150MX"
##  [25] "Intel Iris Graphics 540"        "AMD Radeon RX 580"
##  [27] "Nvidia GeForce 920MX"           "AMD Radeon R4 Graphics"
##  [29] "AMD Radeon 520"                 "Nvidia GeForce GTX 1070"
##  [31] "Nvidia GeForce GTX 1050 Ti"     "Intel HD Graphics 400"
##  [33] "Nvidia GeForce MX130"           "AMD R4 Graphics"
##  [35] "Nvidia GeForce GTX 940MX"       "AMD Radeon RX 560"
##  [37] "Nvidia GeForce 920M"            "AMD Radeon R7 M445"
##  [39] "AMD Radeon RX 550"              "Nvidia GeForce GTX 1050M"
##  [41] "Intel HD Graphics 515"          "AMD Radeon R5 M420"
##  [43] "Intel HD Graphics 505"          "Nvidia GTX 980 SLI"
##  [45] "AMD R17M-M1-70"                 "Nvidia GeForce GTX 1080"
##  [47] "Nvidia Quadro M1200"            "Nvidia GeForce 920MX "
##  [49] "Nvidia GeForce GTX 950M"        "AMD FirePro W4190M "
##  [51] "Nvidia GeForce GTX 980M"        "Intel Iris Graphics 550"
##  [53] "Nvidia GeForce 930M"            "Intel HD Graphics 630"
##  [55] "AMD Radeon R5 430"              "Nvidia GeForce GTX 940M"
##  [57] "Intel HD Graphics 510"          "Intel HD Graphics 405"
##  [59] "AMD Radeon RX 540"              "Nvidia GeForce GT 940MX"
##  [61] "AMD FirePro W5130M"             "Nvidia Quadro M2200M"
##  [63] "AMD Radeon R4"                  "Nvidia Quadro M620"
##  [65] "AMD Radeon R7 M460"             "Intel HD Graphics 530"
##  [67] "Nvidia GeForce GTX 965M"        "Nvidia GeForce GTX1080"
##  [69] "Nvidia GeForce GTX1050 Ti"      "Nvidia GeForce GTX 960M"
##  [71] "AMD Radeon R2 Graphics"         "Nvidia Quadro M620M"
##  [73] "Nvidia GeForce GTX 970M"        "Nvidia GeForce GTX 960<U+039C>"
##  [75] "Intel Graphics 620"             "Nvidia GeForce GTX 960"
##  [77] "AMD Radeon R5 520"              "AMD Radeon R7 M440"
##  [79] "AMD Radeon R7"                  "Nvidia Quadro M520M"
##  [81] "Nvidia Quadro M2200"            "Nvidia Quadro M2000M"
##  [83] "Intel HD Graphics 540"          "Nvidia Quadro M1000M"
##  [85] "AMD Radeon 540"                 "Nvidia GeForce GTX 1070M"
##  [87] "Nvidia GeForce GTX1060"         "Intel HD Graphics 5300"
##  [89] "AMD Radeon R5 M420X"            "AMD Radeon R7 Graphics"
##  [91] "Nvidia GeForce 920"             "Nvidia GeForce 940M"
##  [93] "Nvidia GeForce GTX 930MX"       "AMD Radeon R7 M465"
##  [95] "AMD Radeon R3"                  "Nvidia GeForce GTX 1050Ti"
##  [97] "AMD Radeon R7 M365X"            "AMD Radeon R9 M385"
```

```
##  [99] "Intel HD Graphics 620 "         "Nvidia Quadro 3000M"
## [101] "Nvidia GeForce GTX 980 "        "AMD Radeon R5 M330"
## [103] "AMD FirePro W4190M"             "AMD FirePro W6150M"
## [105] "AMD Radeon R5 M315"             "Nvidia Quadro M500M"
## [107] "AMD Radeon R7 M360"             "Nvidia Quadro M3000M"
## [109] "Nvidia GeForce 960M"            "ARM Mali T860 MP4"
##
## $OpSys
## [1] "macOS"        "No OS"        "Windows 10"   "Mac OS X"     "Linux"
## [6] "Windows 10 S" "Chrome OS"    "Windows 7"    "Android"
##
## $Weight
##   [1] "1.37kg"   "1.34kg"   "1.86kg"   "1.83kg"   "2.1kg"    "2.04kg"
##   [7] "1.3kg"    "1.6kg"    "2.2kg"    "0.92kg"   "1.22kg"   "2.5kg"
##  [13] "1.62kg"   "1.91kg"   "2.3kg"    "1.35kg"   "1.88kg"   "1.89kg"
##  [19] "1.65kg"   "2.71kg"   "1.2kg"    "1.44kg"   "2.8kg"    "2kg"
##  [25] "2.65kg"   "2.77kg"   "3.2kg"    "1.49kg"   "2.4kg"    "2.13kg"
##  [31] "2.43kg"   "1.7kg"    "1.4kg"    "1.8kg"    "1.9kg"    "3kg"
##  [37] "1.252kg"  "2.7kg"    "2.02kg"   "1.63kg"   "1.96kg"   "1.21kg"
##  [43] "2.45kg"   "1.25kg"   "1.5kg"    "2.62kg"   "1.38kg"   "1.58kg"
##  [49] "1.85kg"   "1.23kg"   "2.16kg"   "2.36kg"   "7.2kg"    "2.05kg"
##  [55] "1.32kg"   "1.75kg"   "0.97kg"   "2.56kg"   "1.48kg"   "1.74kg"
##  [61] "1.1kg"    "1.56kg"   "2.03kg"   "1.05kg"   "5.4kg"    "4.4kg"
##  [67] "1.90kg"   "1.29kg"   "2.0kg"    "1.95kg"   "2.06kg"   "1.12kg"
##  [73] "3.49kg"   "3.35kg"   "2.23kg"   "?"        "2.9kg"    "4.42kg"
##  [79] "2.69kg"   "2.37kg"   "4.7kg"    "3.6kg"    "2.08kg"   "4.3kg"
##  [85] "1.68kg"   "1.41kg"   "4.14kg"   "2.18kg"   "2.24kg"   "2.67kg"
##  [91] "4.1kg"    "2.14kg"   "1.36kg"   "2.25kg"   "2.15kg"   "2.19kg"
##  [97] "2.54kg"   "3.42kg"   "5.8kg"    "1.28kg"   "2.33kg"   "1.45kg"
## [103] "2.79kg"   "8.23kg"   "1.26kg"   "1.84kg"   "0.0002kg" "2.6kg"
## [109] "2.26kg"   "3.25kg"   "1.59kg"   "1.13kg"   "1.42kg"   "1.78kg"
## [115] "1.10kg"   "1.15kg"   "1.27kg"   "1.43kg"   "2.31kg"   "1.16kg"
## [121] "1.64kg"   "2.17kg"   "1.47kg"   "3.78kg"   "1.79kg"   "0.91kg"
## [127] "1.99kg"   "4.33kg"   "1.93kg"   "1.87kg"   "2.63kg"   "3.4kg"
## [133] "3.14kg"   "1.94kg"   "1.24kg"   "4.6kg"    "4.5kg"    "8.4kg"
## [139] "2.73kg"   "1.39kg"   "2.29kg"   "2.59kg"   "2.94kg"   "11.1kg"
## [145] "1.14kg"   "3.8kg"    "6.2kg"    "3.31kg"   "1.09kg"   "3.21kg"
## [151] "1.19kg"   "1.98kg"   "1.17kg"   "4.36kg"   "1.71kg"   "2.32kg"
## [157] "4.2kg"    "1.55kg"   "0.81kg"   "1.18kg"   "2.72kg"   "1.31kg"
## [163] "0.920kg"  "3.74kg"   "1.76kg"   "1.54kg"   "2.83kg"   "2.07kg"
## [169] "2.38kg"   "3.58kg"   "1.08kg"   "2.20kg"   "0.98kg"   "2.75kg"
## [175] "1.70kg"   "2.99kg"   "1.11kg"   "2.09kg"   "4kg"      "3.0kg"
## [181] "0.99kg"   "0.69kg"   "3.52kg"   "2.591kg"  "2.21kg"   "3.3kg"
## [187] "2.191kg"  "2.34kg"   "4.0kg"
##
## $Price
##   [1]  71378.68  47895.52  30636.00 135195.34  96095.81  21312.00 114017.60
##   [8]  61735.54  79653.60  41025.60  20986.99  18381.07 130001.60  26581.39
##  [15]  67260.67  80908.34  39693.60 152274.24  26586.72  52161.12  53226.72
##  [22]  13746.24  43636.32  35111.52  22305.14  58554.72  42624.00  69157.44
##  [29]  47738.88  13053.07  10602.72  23389.92  99580.32  53173.44  13266.72
##  [36]  19553.76  26037.40  46833.12  20725.92  79866.72  27864.91  36336.96
##  [43]  75604.32  69210.72  34045.92  24828.48  44808.48  21231.55  58767.84
##  [50]  20459.52  40908.38  31232.20 130482.72  22111.20  31914.72  50136.48
```

```
##  [57]  36763.20 105654.24  23373.40  12201.12  29250.72  50562.72  58021.92
##  [64]  50882.40  46353.60  58341.60  27652.32  45554.40  28238.40  52054.56
##  [71]  58403.40  80452.80  45820.80  21258.72  21045.60  71874.72  37242.72
##  [78]  31914.19  77202.72  87858.72  36709.92  63776.16  63669.60  55890.72
##  [85]  45128.16  31962.67  25840.80  30742.56  66546.72  38308.32  18594.72
##  [92]  34472.16  59620.32  71395.20  22105.87  63563.04  78854.40  67239.36
##  [99]  73473.12  74538.72  38468.16  86793.12  57755.52  60223.98  30049.92
## [106]  59567.04  25521.12 119427.12  33513.12  67718.88  24029.28  43263.36
## [113]  14811.31  74378.88  49443.84  34045.39  23922.72  47099.52  30476.16
## [120]  31861.44  52640.64  13445.74  49976.64  34898.40  59461.55  46300.32
## [127]  32074.56  19660.32 107305.92  18328.32  23816.16  66560.57  47898.72
## [134]  26533.44 100699.20  57648.96  32980.32  70063.20  21471.84  42890.40
## [141]  38787.84  57489.12  18541.44  95850.72  19367.81  56502.91  45501.12
## [148]  40173.12  16463.52  26053.92  49177.44  24455.52 149130.72  43316.64
## [155]  98514.72  42251.04  63882.72  82530.72 127712.16  41505.12  52693.92
## [162]  57808.80  13852.80  53274.67  37189.44  44701.92  48697.92 324954.72
## [169]  51095.52  55677.60  98301.60  26267.04  39533.76  93186.72 162770.40
## [176]  74485.44 103842.72  74964.96  49650.57  31381.92  54931.68  61218.72
## [183]  68145.12  36089.21  72620.64  42304.32 130873.80  44328.96  45768.05
## [190]  40972.32  47472.48  67612.32  21258.19  17582.40  45767.52  20779.20
## [197] 207259.20  45074.88  61005.60  47365.92  52480.80  29783.52 159786.72
## [204]  35964.00 108691.20  24988.32  37402.56  23757.55  56423.52 133146.72
## [211]  90522.72  60845.76  23656.32  35004.96  30103.20  42570.72  54239.04
## [218]  46886.40 104370.19  39164.53  37992.37  45234.72  22803.84  44169.12
## [225]  30849.12  50669.28  58448.16  62817.12  35112.05  63243.36  97449.12
## [232]  39373.92 153705.34  78215.04  27119.52 113060.16  34578.72  67399.20
## [239]  19180.27 105228.00  55571.04 111834.72  14652.00  44968.32  24503.47
## [246]  52214.40  68837.76  58288.32  48058.56  15557.76  55938.67  71128.80
## [253] 140605.92  50243.04  71075.52  67559.04  60952.32  14651.47  60885.72
## [260]  14646.67  38148.48  84129.12  60153.12  14865.12  85672.11  19980.00
## [267]  35324.64  69477.12  75071.52  92615.03  74751.84  51729.55  17155.63
## [274]  29696.67  76030.56  50349.60  43103.52  93240.00  22697.28 117162.72
## [281]  46300.85  26053.39  29463.84  15238.08  63456.48  21498.48  88178.40
## [288]  93181.39 121584.96  72940.32 113752.80 133679.52  55357.92  84768.48
## [295]  36975.79  41498.19  65510.96 144495.36 139860.00  16303.68  81465.12
## [302]  60978.96 119826.72  99793.44  93080.16  89510.40  21791.52  16221.10
## [309] 102564.00 103523.04  42038.45  31909.39  79920.00  28768.54  64755.45
## [316] 101178.72  31808.16  61751.52  60867.07 106506.72  15930.72  14332.32
## [323]  53812.80 130269.60  90309.60  18488.16 126912.96  39906.72  76137.12
## [330]  23539.10 106187.04  54757.99 137941.92  81731.52  71661.60 109010.88
## [337]  34093.87  28984.32  23176.80 111355.20  16197.12  40439.52  98994.24
## [344]  61485.12  67932.00  98133.77  39427.20 128298.24  72673.92  89084.16
## [351]  31254.05  38681.28  13261.39 124568.64  95797.44 103896.00  53918.83
## [358]  71928.00  47893.39  48538.08  18115.20  32979.79  85194.72  14119.20
## [365]  65214.72  32660.64  70489.44  44542.08  64961.11  74589.34  13053.60
## [372]  27783.92 158135.04  80133.12  68184.01  31168.80  73366.56 160520.39
## [379]  42486.00  39207.15  42517.44  37589.04  41824.80  24634.01  21152.16
## [386]  92121.12  59513.23 143802.72  28992.31  68198.40  11934.72  13586.40
## [393]  30310.99  32921.71  33566.40 104695.20  52747.20  99153.55  44222.40
## [400]  36496.80  99367.20 141884.64 145401.12  39907.25  53733.95  81912.14
## [407]  15717.60 125154.72  79813.44  89137.44  32447.52  94305.60  32127.84
## [414]  28185.12 107892.00  78534.72  88977.60  93932.64  64948.32  35616.61
## [421]  17529.12  24775.20 122490.72 261018.72  46833.65  23650.99  19127.52
## [428]  62231.04  74005.92 120831.58  20193.12  59886.72  78055.20  41345.28
```

```
## [435]  49656.96  12733.92  24935.04  34046.45 101232.00  78801.12  48304.71
## [442]  93635.34 127818.72  59087.52  27753.55 130536.00  62284.32  23976.00
## [449]  14598.72  48964.32 138474.72  64628.64  84395.52  37775.52  77250.67
## [456]  63499.10  19441.87  56689.92  60472.80  63722.88 167691.87  65481.12
## [463]  22324.32  28504.80  28717.92  79215.11  67026.24  21951.36  99519.05
## [470]  43580.38  55091.52  68944.32  26373.60  53168.11 109277.28 149916.60
## [477]  32639.86  29073.30  30316.32  16943.04 101657.71 137995.20  51841.44
## [484]  35644.32 100006.56  22857.12  33110.86  42357.60  98834.40  34898.93
## [491]  15877.44  76012.44  45664.69  63936.00  89864.18  44574.05 194972.83
## [498]  71847.01  26101.87  31409.63 103096.80  34632.00  51148.80 128884.32
## [505] 111593.89  21887.42  64308.96  35431.20  81784.80 175770.72  60031.11
## [512] 233845.92  25308.00  45282.67  83170.08 152859.79  55837.44  99047.52
## [519]  83063.52  25059.72  62938.07  54345.60 126273.60  33886.08 154458.72
## [526]  31003.63 191211.26 125208.00  93985.92 100752.48  71341.92  10810.51
## [533] 104961.60  62071.20 124142.40  43156.80 110017.87  39640.32  41558.40
## [540]  89457.12 128671.20  51202.08  42081.08  20512.80 133467.47  40226.40
## [547]  58075.20  99900.00  72354.24  18914.40 292986.72  63190.08  87912.00
## [554] 147832.29  17316.00  31435.20 163723.58  48484.80  38041.39  99633.60
## [561]  32767.20  54665.28 121318.56  46087.20 101391.84  41931.36 136343.52
## [568]  45323.16  68464.80  54185.76 119347.20  94731.84  56210.40  73952.64
## [575]  39160.80  90043.20  49816.80  48618.00  90576.00  36486.14  24279.70
## [582]  72988.27  15824.16 141138.72  78588.00  49497.12  47952.00  32713.92
## [589] 107257.97 109170.72 104588.11  79014.24  42037.92  64202.40 165168.00
## [596]  54291.79  46939.68  59668.80  36496.27  69103.63  97236.00  38889.07
## [603]  37725.44  15392.59  58874.40  88924.32  29762.21 146946.24  78438.82
## [610]  94572.00  77788.80  79333.39 168045.12  41292.00  93772.80  53759.52
## [617]  95371.20  45101.52  24808.23  43956.00 167778.72  37029.60 104587.57
## [624] 149184.00  62870.40 109218.67  53386.56  48751.20  42943.68  63349.92
## [631] 101658.24  14418.63  61272.00  20246.40 111301.92  67132.80  29144.16
## [638]  58607.47  55904.57  36443.52  81997.92  84715.20 100550.55  64468.80
## [645]  41025.07  79387.20 118761.12  42410.35  63159.71  55754.32  56476.80
## [652]  86526.72  57542.40  65480.59  91908.00  56633.98  53839.97  11231.42
## [659]  75924.00  55922.69  53280.00 112065.96  78268.32  82351.70 122010.67
## [666]  15339.31 139593.60 120093.12  37570.39  96916.32  52746.67 102777.12
## [673]  77682.24 211788.00  42517.97  70809.12  69264.00  25679.89  91294.75
## [680]  28771.20  50083.20 172627.20  34433.27  39960.00  54931.15  20619.36
## [687]  20965.15  18434.35  43601.69  34035.26  26640.00  60480.79  62176.16
## [694]  11135.52  75289.97  80516.20  46193.76  21205.44  40980.31 104908.32
## [701] 146519.47  54825.12 118601.28  69929.47  10442.88  80612.64  27899.01
## [708] 100965.60 109244.25  14811.84  40066.56  32820.48 210424.03  41771.52
## [715] 115709.24 130003.20  60888.38 122381.50  53807.47 124621.92  18061.92
## [722] 119916.23  25515.26  79536.38  42010.75 108744.48  94252.32  25414.03
## [729]  74059.20  36177.12  69530.40 114731.55  87219.36  44275.68  47686.13
## [736]  17742.24  48431.52  36816.48  61964.64  70702.56  19607.04 114552.00
## [743]  14492.16 109165.39 142790.40 103842.19   9270.72  67772.16  78647.14
## [750]  91288.35  78694.56  27804.70 186426.72  24988.85  85141.44  25467.84
## [757] 117119.56  39267.36  31838.53  19276.70  46620.00 123876.00  30529.44
## [764]  96596.64  17262.72  57116.16  23655.79  26107.20  47685.60  44382.77
## [771]  38841.12  15397.92  29303.47  42943.15  38378.65  33992.64  40705.92
```

#As there is no independent company named Vero but its parent is aspire, we will be renaming any location where Vero is present to Aspire.

```
dataset$Company = ifelse(dataset$Company == "Vero", "Aspire", dataset$Company)
```

```
dataset = dataset %>%
  filter(Inches!='?')
```

"?" is present in inches column, droppping those rows

```
dataset$ScreenResolution = sapply(dataset$ScreenResolution, function(x) tail(strsplit(x, " ")[[1]], 1))
```

clean the screen resolution column by extracting only resolution values

```
dataset = dataset %>%
  mutate(`Cpu Name` = sapply(strsplit(Cpu, " "), function(x) paste(x[1:3], collapse = " ")))

fetch_processor = function(text) {
  if (text == 'Intel Core i7' || text == 'Intel Core i5' || text == 'Intel Core i3') {
    return(text)
  } else {
    if (strsplit(text, " ")[[1]][1] == 'Intel') {
      return('Other Intel Processor')
    } else {
      return('AMD Processor')
    }
  }
}

dataset$`Cpu brand` = sapply(dataset$`Cpu Name`, fetch_processor)
```

clean the cpu column

```
dataset$ProcessSpeed = sapply(dataset$Cpu, function(x) as.numeric(substr(strsplit(x, " ")[[1]][length(s
dataset = dataset[-5]
dataset = dataset[-11]
```

from cpu column we are going to extract only processing speeds

```
dataset$Ram = as.integer(substr(dataset$Ram,1, nchar(dataset$Ram)-2))
```

cleaning ram column, by changing its datatype to int

cleaning memory column #we are going to create 2 columns, one as storage type and rom

```
unique_values = unique(dataset$Memory)


print(unique_values)
```

```
##  [1] "128GB SSD"                     "128GB Flash Storage"
##  [3] "256GB SSD"                     "512GB SSD"
##  [5] "500GB HDD"                     "256GB Flash Storage"
##  [7] "1TB HDD"                       "128GB SSD +  1TB HDD"
##  [9] "256GB SSD +  256GB SSD"        "64GB Flash Storage"
## [11] "32GB Flash Storage"            "256GB SSD +  1TB HDD"
## [13] "256GB SSD +  2TB HDD"          "32GB SSD"
## [15] "2TB HDD"                       "64GB SSD"
## [17] "1.0TB Hybrid"                  "512GB SSD +  1TB HDD"
## [19] "1TB SSD"                       "256GB SSD +  500GB HDD"
## [21] "128GB SSD +  2TB HDD"          "512GB SSD +  512GB SSD"
## [23] "16GB SSD"                      "16GB Flash Storage"
## [25] "512GB SSD +  256GB SSD"        "512GB SSD +  2TB HDD"
## [27] "64GB Flash Storage +  1TB HDD" "180GB SSD"
## [29] "1TB HDD +  1TB HDD"            "32GB HDD"
## [31] "1TB SSD +  1TB HDD"            "?"
## [33] "512GB Flash Storage"           "128GB HDD"
## [35] "240GB SSD"                     "8GB SSD"
## [37] "508GB Hybrid"                  "1.0TB HDD"
## [39] "512GB SSD +  1.0TB Hybrid"     "256GB SSD +  1.0TB Hybrid"
```

#Removing the values with '?'

```
dataset = dataset %>%
  filter(Memory !='?')
```

#Replace patterns in Memory column

```
dataset$Memory = gsub("\\.0", "", dataset$Memory)
dataset$Memory = gsub("GB", "", dataset$Memory)
dataset$Memory = gsub("TB", "000", dataset$Memory)
```

#Split Memory column into two columns

```
dataset = dataset %>%
  separate(Memory, into = c("first", "second"), sep = "\\+", fill = "right") %>%
  mutate(first = str_trim(first),
         second = str_trim(second))
```

#Create indicator variables for each storage type

```
dataset = dataset %>%
  mutate(Layer1HDD = if_else(str_detect(first, "HDD"), 1, 0),
         Layer1SSD = if_else(str_detect(first, "SSD"), 1, 0),
```

```
         Layer1Hybrid = if_else(str_detect(first, "Hybrid"), 1, 0),
         Layer1Flash_Storage = if_else(str_detect(first, "Flash Storage"), 1, 0),
         first = as.integer(gsub("\\D", "", first)),
         second = as.integer(gsub("\\D", "", second)),
         second = if_else(is.na(second), 0, second),
         Layer2HDD = if_else(str_detect(second, "HDD"), 1, 0),
         Layer2SSD = if_else(str_detect(second, "SSD"), 1, 0),
         Layer2Hybrid = if_else(str_detect(second, "Hybrid"), 1, 0),
         Layer2Flash_Storage = if_else(str_detect(second, "Flash Storage"), 1, 0))
```

#Calculate storage quantities

```
dataset = dataset %>%
  mutate(HDD = first * Layer1HDD + second * Layer2HDD,
         SSD = first * Layer1SSD + second * Layer2SSD,
         Hybrid = first * Layer1Hybrid + second * Layer2Hybrid,
         Flash_Storage = first * Layer1Flash_Storage + second * Layer2Flash_Storage)
```

#Remove unnecessary columns

```
dataset = subset(dataset, select = -c(first, second, Layer1HDD, Layer1SSD, Layer1Hybrid, Layer1Flash_St
```

```
table(dataset$Gpu)
```

Cleaning GPU column

```
##
##             AMD FirePro W4190M            AMD FirePro W4190M
##                              1                              2
##             AMD FirePro W5130M             AMD FirePro W6150M
##                              1                              1
##                 AMD R17M-M1-70                AMD R4 Graphics
##                              1                              1
##                 AMD Radeon 520                 AMD Radeon 530
##                             16                             39
##                 AMD Radeon 540             AMD Radeon Pro 455
##                              1                              1
##             AMD Radeon Pro 555             AMD Radeon Pro 560
##                              1                              1
##                  AMD Radeon R2         AMD Radeon R2 Graphics
##                              5                              4
##                  AMD Radeon R3                  AMD Radeon R4
##                              1                              3
##         AMD Radeon R4 Graphics                  AMD Radeon R5
##                              5                             11
##               AMD Radeon R5 430              AMD Radeon R5 520
##                              1                              1
##             AMD Radeon R5 M315             AMD Radeon R5 M330
##                              1                              5
```

```
##             AMD Radeon R5 M420            AMD Radeon R5 M420X
##                              7                              3
##             AMD Radeon R5 M430                 AMD Radeon R7
##                             20                              1
##          AMD Radeon R7 Graphics             AMD Radeon R7 M360
##                              1                              1
##            AMD Radeon R7 M365X             AMD Radeon R7 M440
##                              1                              3
##             AMD Radeon R7 M445             AMD Radeon R7 M460
##                             13                              2
##             AMD Radeon R7 M465             AMD Radeon R9 M385
##                              1                              1
##              AMD Radeon RX 540              AMD Radeon RX 550
##                              2                              4
##              AMD Radeon RX 560              AMD Radeon RX 580
##                              1                              5
##              ARM Mali T860 MP4             Intel Graphics 620
##                              1                              1
##              Intel HD Graphics          Intel HD Graphics 400
##                             22                             30
##           Intel HD Graphics 405          Intel HD Graphics 500
##                              9                             39
##           Intel HD Graphics 505          Intel HD Graphics 510
##                             12                              4
##           Intel HD Graphics 515          Intel HD Graphics 520
##                             13                            177
##           Intel HD Graphics 530         Intel HD Graphics 5300
##                              1                              2
##           Intel HD Graphics 540         Intel HD Graphics 6000
##                              1                              5
##           Intel HD Graphics 615          Intel HD Graphics 620
##                             14                            269
##          Intel HD Graphics 620           Intel HD Graphics 630
##                              1                              4
##        Intel Iris Graphics 540        Intel Iris Graphics 550
##                              2                              1
##   Intel Iris Plus Graphics 640   Intel Iris Plus Graphics 650
##                              8                              2
##        Intel Iris Pro Graphics         Intel UHD Graphics 620
##                              1                             66
##           Nvidia GeForce 150MX            Nvidia GeForce 920
##                              3                              1
##            Nvidia GeForce 920M           Nvidia GeForce 920MX
##                              4                             12
##          Nvidia GeForce 920MX             Nvidia GeForce 930M
##                              5                              5
##           Nvidia GeForce 930MX          Nvidia GeForce 930MX
##                             20                              5
##            Nvidia GeForce 940M           Nvidia GeForce 940MX
##                              1                             42
##            Nvidia GeForce 960M        Nvidia GeForce GT 940MX
##                              1                              5
##        Nvidia GeForce GTX 1050     Nvidia GeForce GTX 1050 Ti
##                             64                             27
```

```
##       Nvidia GeForce GTX 1050M      Nvidia GeForce GTX 1050Ti
##                              3                              2
##        Nvidia GeForce GTX 1060        Nvidia GeForce GTX 1070
##                             48                             29
##       Nvidia GeForce GTX 1070M        Nvidia GeForce GTX 1080
##                              1                              6
##       Nvidia GeForce GTX 930MX        Nvidia GeForce GTX 940M
##                              1                              1
##       Nvidia GeForce GTX 940MX        Nvidia GeForce GTX 950M
##                              4                              7
##         Nvidia GeForce GTX 960 Nvidia GeForce GTX 960<U+039C>
##                              2                              2
##        Nvidia GeForce GTX 960M        Nvidia GeForce GTX 965M
##                             12                              4
##        Nvidia GeForce GTX 970M        Nvidia GeForce GTX 980
##                              5                              1
##        Nvidia GeForce GTX 980M      Nvidia GeForce GTX1050 Ti
##                             10                              1
##         Nvidia GeForce GTX1060         Nvidia GeForce GTX1080
##                              1                              1
##           Nvidia GeForce MX130           Nvidia GeForce MX150
##                              6                             15
##             Nvidia GTX 980 SLI            Nvidia Quadro 3000M
##                              1                              1
##           Nvidia Quadro M1000M            Nvidia Quadro M1200
##                              4                              8
##           Nvidia Quadro M2000M            Nvidia Quadro M2200
##                              2                              2
##           Nvidia Quadro M2200M           Nvidia Quadro M3000M
##                              3                              1
##            Nvidia Quadro M500M            Nvidia Quadro M520M
##                              1                              2
##             Nvidia Quadro M620            Nvidia Quadro M620M
##                              5                              1
```

#Create a new 'Gpu brand' column

```
dataset$Gpu_brand <- sapply(strsplit(as.character(dataset$Gpu), " "), function(x) x[1])

table(dataset$Gpu_brand)
```

```
##
##    AMD    ARM  Intel Nvidia
##    169      1    684    388
```

```
dataset = dataset %>%
  filter(Gpu_brand != 'ARM')
```

#drop Gpu column

```
dataset = dataset[-6]
```

```
cat_os = function(inp) {
  if (inp %in% c("Windows 10", "Windows 7", "Windows 10 S")) {
    return("Windows")
  } else if (inp %in% c("macOS", "Mac OS X")) {
    return("Mac")
  } else {
    return("Others/No OS/Linux")
  }
}

dataset$OS_category = sapply(dataset$OpSys, cat_os)

dataset = dataset[-6]
```

cleaning Os coloumn

```
u = unique(dataset$Weight)
print(u)
```

Cleaning weight column

```
##   [1] "1.37kg"   "1.34kg"   "1.86kg"   "1.83kg"   "2.1kg"    "2.04kg"
##   [7] "1.3kg"    "1.6kg"    "2.2kg"    "0.92kg"   "1.22kg"   "2.5kg"
##  [13] "1.62kg"   "1.91kg"   "2.3kg"    "1.35kg"   "1.88kg"   "1.89kg"
##  [19] "1.65kg"   "2.71kg"   "1.2kg"    "1.44kg"   "2.8kg"    "2kg"
##  [25] "2.65kg"   "2.77kg"   "3.2kg"    "1.49kg"   "2.4kg"    "2.13kg"
##  [31] "2.43kg"   "1.7kg"    "1.4kg"    "1.8kg"    "1.9kg"    "3kg"
##  [37] "1.252kg"  "2.7kg"    "2.02kg"   "1.63kg"   "1.96kg"   "1.21kg"
##  [43] "2.45kg"   "1.25kg"   "1.5kg"    "2.62kg"   "1.38kg"   "1.58kg"
##  [49] "1.85kg"   "1.23kg"   "2.16kg"   "2.36kg"   "7.2kg"    "2.05kg"
##  [55] "1.32kg"   "1.75kg"   "0.97kg"   "2.56kg"   "1.48kg"   "1.74kg"
##  [61] "1.1kg"    "1.56kg"   "2.03kg"   "1.05kg"   "5.4kg"    "4.4kg"
##  [67] "1.90kg"   "1.29kg"   "2.0kg"    "1.95kg"   "2.06kg"   "1.12kg"
##  [73] "3.49kg"   "3.35kg"   "2.23kg"   "?"        "2.9kg"    "4.42kg"
##  [79] "2.69kg"   "2.37kg"   "4.7kg"    "3.6kg"    "2.08kg"   "4.3kg"
##  [85] "1.68kg"   "1.41kg"   "4.14kg"   "2.18kg"   "2.24kg"   "2.67kg"
##  [91] "4.1kg"    "2.14kg"   "1.36kg"   "2.25kg"   "2.15kg"   "2.19kg"
##  [97] "2.54kg"   "3.42kg"   "5.8kg"    "1.28kg"   "2.33kg"   "1.45kg"
## [103] "2.79kg"   "8.23kg"   "1.26kg"   "1.84kg"   "0.0002kg" "2.6kg"
## [109] "2.26kg"   "3.25kg"   "1.59kg"   "1.13kg"   "1.42kg"   "1.78kg"
## [115] "1.10kg"   "1.15kg"   "1.27kg"   "1.43kg"   "2.31kg"   "1.16kg"
## [121] "1.64kg"   "2.17kg"   "1.47kg"   "3.78kg"   "1.79kg"   "0.91kg"
## [127] "1.99kg"   "4.33kg"   "1.93kg"   "1.87kg"   "2.63kg"   "3.4kg"
## [133] "3.14kg"   "1.94kg"   "1.24kg"   "4.6kg"    "4.5kg"    "8.4kg"
## [139] "2.73kg"   "1.39kg"   "2.29kg"   "2.59kg"   "2.94kg"   "11.1kg"
## [145] "1.14kg"   "3.8kg"    "6.2kg"    "3.31kg"   "1.09kg"   "3.21kg"
## [151] "1.19kg"   "1.98kg"   "1.17kg"   "4.36kg"   "1.71kg"   "2.32kg"
## [157] "4.2kg"    "1.55kg"   "0.81kg"   "1.18kg"   "2.72kg"   "1.31kg"
```

```
## [163] "0.920kg"  "3.74kg"   "1.76kg"   "1.54kg"   "2.83kg"   "2.07kg"
## [169] "2.38kg"   "3.58kg"   "1.08kg"   "2.20kg"   "0.98kg"   "2.75kg"
## [175] "1.70kg"   "2.99kg"   "1.11kg"   "2.09kg"   "4kg"      "3.0kg"
## [181] "0.99kg"   "0.69kg"   "3.52kg"   "2.591kg"  "2.21kg"   "3.3kg"
## [187] "2.191kg"  "2.34kg"   "4.0kg"
```

```
dataset = dataset %>%
  filter(Weight != '?')
```

```
dataset$Weight <- as.numeric(substr(dataset$Weight, 1, nchar(dataset$Weight) - 2))
```

Remove last two characters and convert 'Weight' column to float

```
dataset$Inches = as.numeric(dataset$Inches)

summary(dataset)
```

Convert 'Inches' column to float

```
##    Company            TypeName             Inches      ScreenResolution
##  Length:1240        Length:1240        Min.   :10.10   Length:1240
##  Class :character   Class :character   1st Qu.:14.00   Class :character
##  Mode  :character   Mode  :character   Median :15.60   Mode  :character
##                                        Mean   :15.14
##                                        3rd Qu.:15.60
##                                        Max.   :35.60
##       Ram              Weight            Price         Cpu brand
##  Min.   : 1.000   Min.   : 0.0002   Min.   :  9271   Length:1240
##  1st Qu.: 4.000   1st Qu.: 1.5000   1st Qu.: 32592   Class :character
##  Median : 8.000   Median : 2.0400   Median : 52694   Mode  :character
##  Mean   : 8.527   Mean   : 2.0810   Mean   : 60557
##  3rd Qu.: 8.000   3rd Qu.: 2.3300   3rd Qu.: 79813
##  Max.   :64.000   Max.   :11.1000   Max.   :324955
##   ProcessSpeed        HDD              SSD             Hybrid
##  Min.   :0.900   Min.   :   0.0   Min.   :   0.0   Min.   :   0.000
##  1st Qu.:2.000   1st Qu.:   0.0   1st Qu.:   0.0   1st Qu.:   0.000
##  Median :2.500   Median :   0.0   Median : 256.0   Median :   0.000
##  Mean   :2.304   Mean   : 244.9   Mean   : 184.9   Mean   :   6.055
##  3rd Qu.:2.700   3rd Qu.: 500.0   3rd Qu.: 256.0   3rd Qu.:   0.000
##  Max.   :3.600   Max.   :2000.0   Max.   :1000.0   Max.   :1000.000
##  Flash_Storage      Gpu_brand         OS_category
##  Min.   :  0.000   Length:1240        Length:1240
##  1st Qu.:  0.000   Class :character   Class :character
##  Median :  0.000   Mode  :character   Mode  :character
##  Mean   :  4.529
##  3rd Qu.:  0.000
##  Max.   :512.000
```

#——-

##Analysis and Visualization

```
price_correlation = cor(dataset$Price, dataset[10:13])
```

correlation between price and storage type

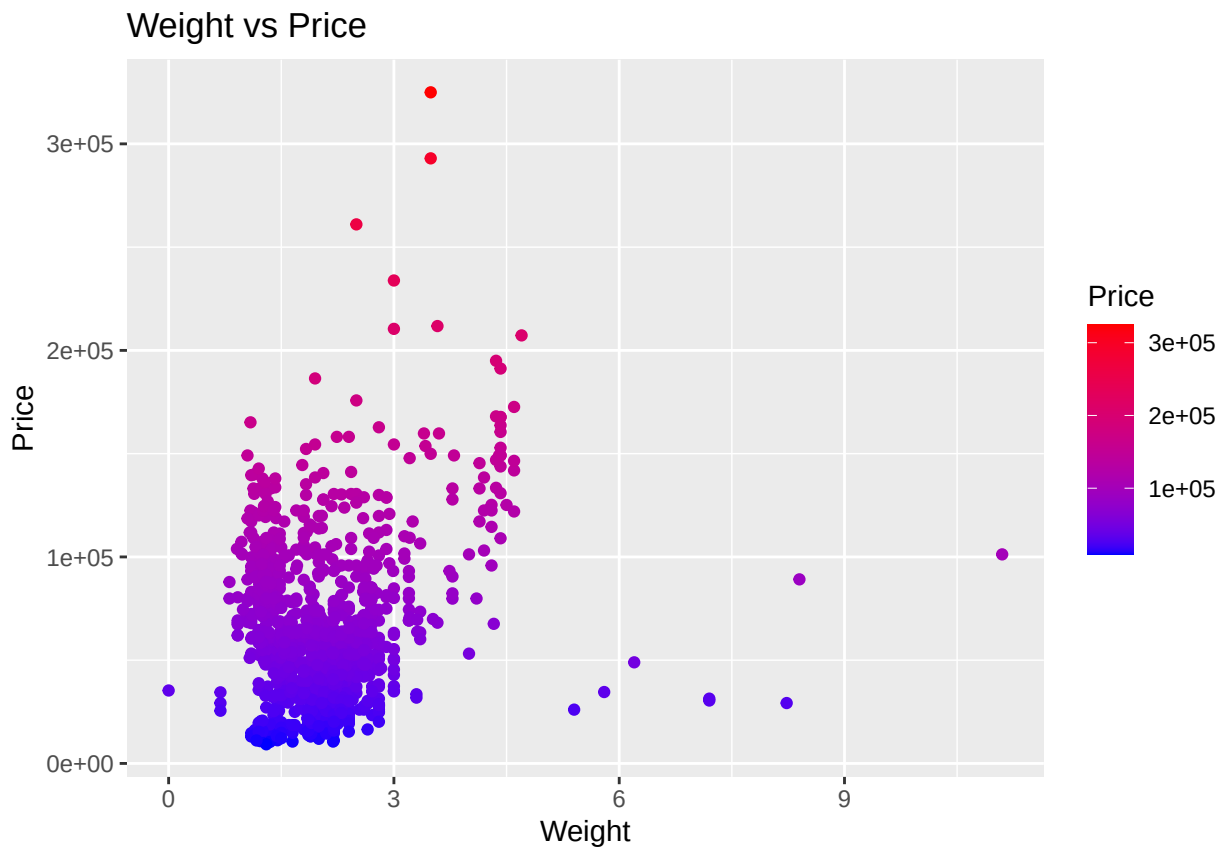
```
print(price_correlation)
```

Print the correlation coefficients for 'Price'

```
##             HDD       SSD      Hybrid Flash_Storage
## [1,] -0.3827848 0.6737999 -0.03040446   -0.03425068
```

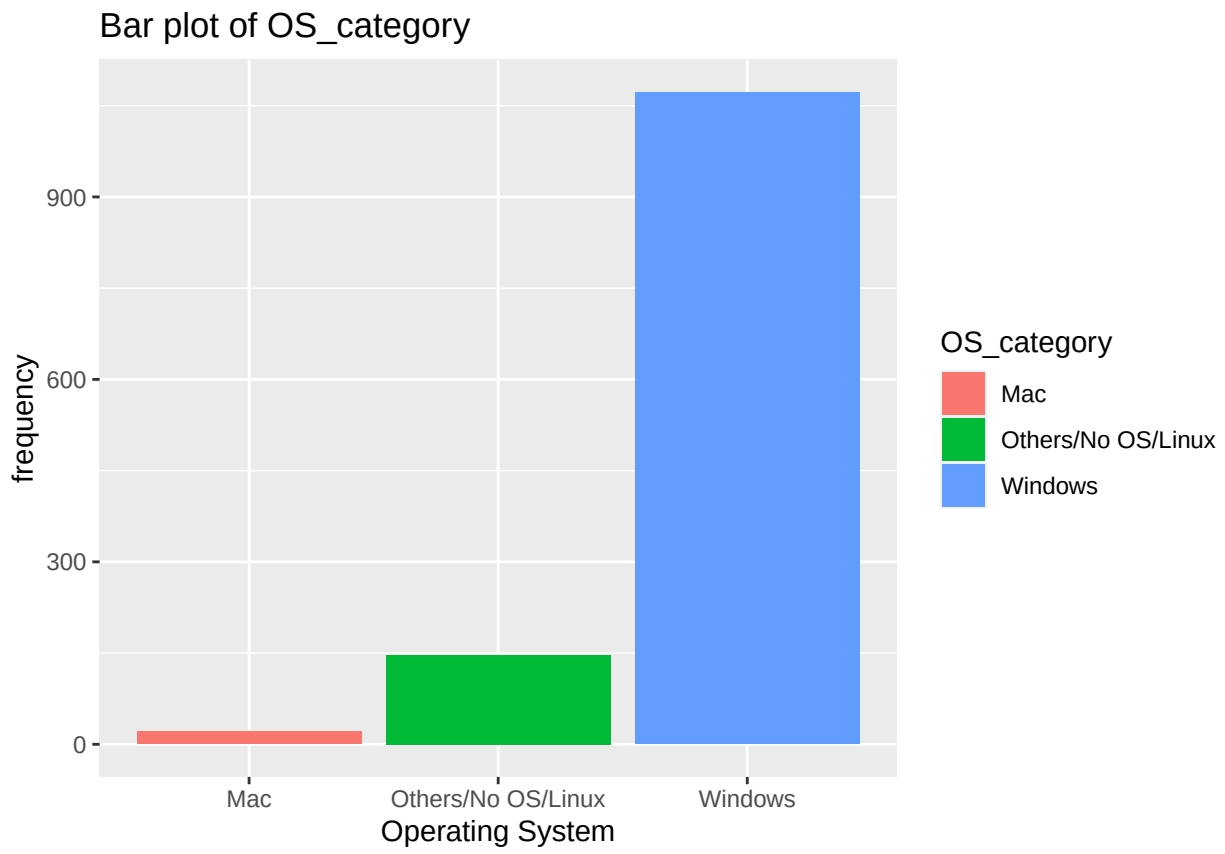
```
ggplot(data = dataset, aes(x = Weight, y = Price, color = Price)) +
  geom_point() +
  labs(x = "Weight", y = "Price", title = "Weight vs Price") +
  scale_color_gradient(low = "blue", high = "red")
```

4.How does the weight of laptops affect their price?



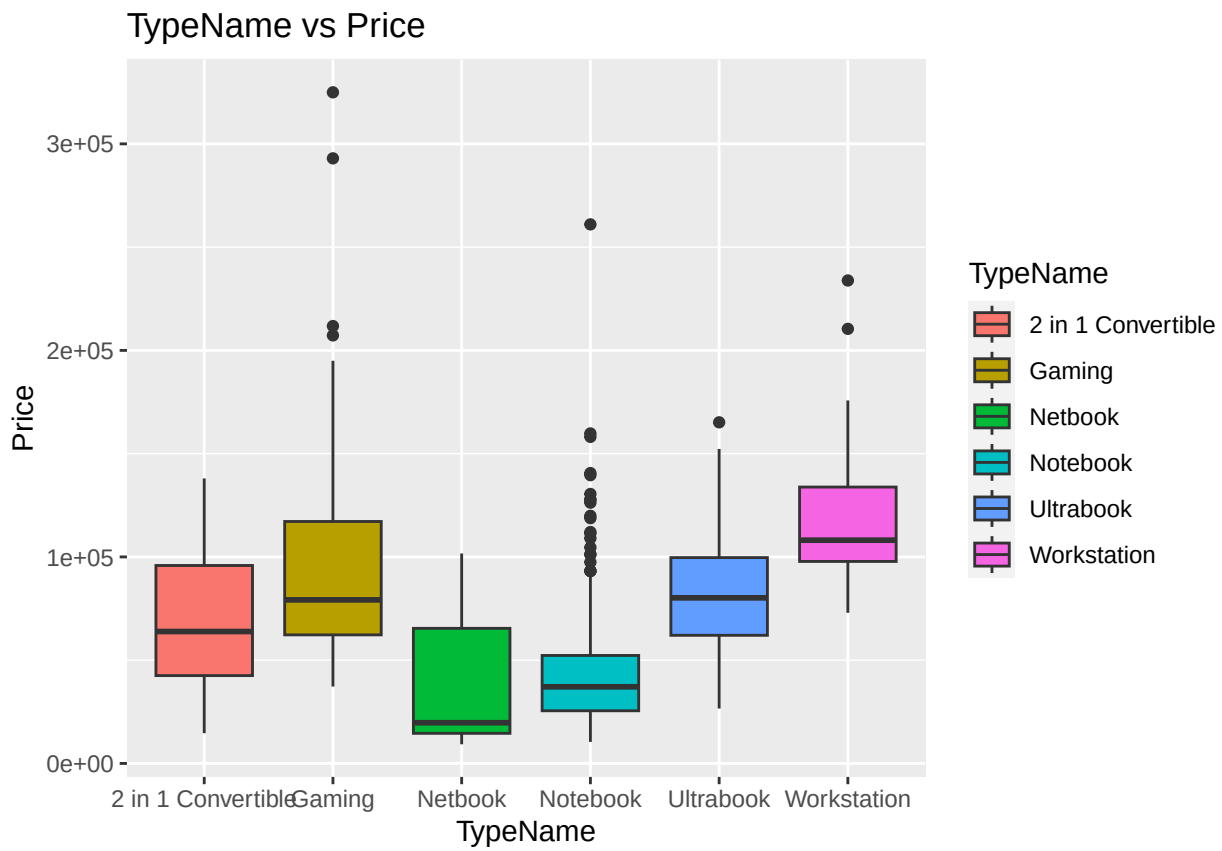

```
ggplot(data = dataset,aes(x = OS_category, fill= OS_category))+
  geom_bar()+
  xlab('Operating System')+
  ylab('frequency')+
  ggtitle("Bar plot of OS_category")
```

common operating system



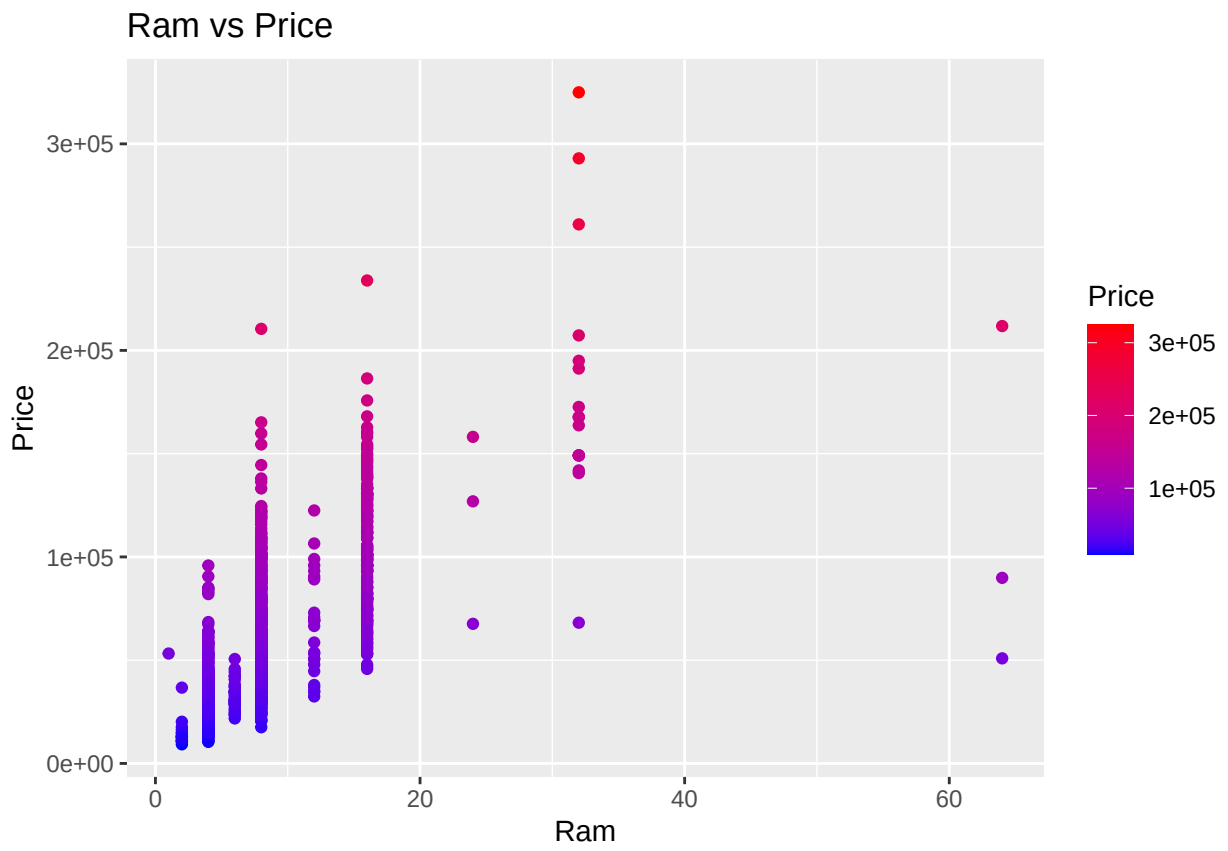

```
ggplot(data = dataset, aes(x = TypeName, y = Price, fill = TypeName)) +
  geom_boxplot() +
  labs(x = "TypeName", y = "Price", title = "TypeName vs Price")
```

1.How does the type of laptop affect its price?



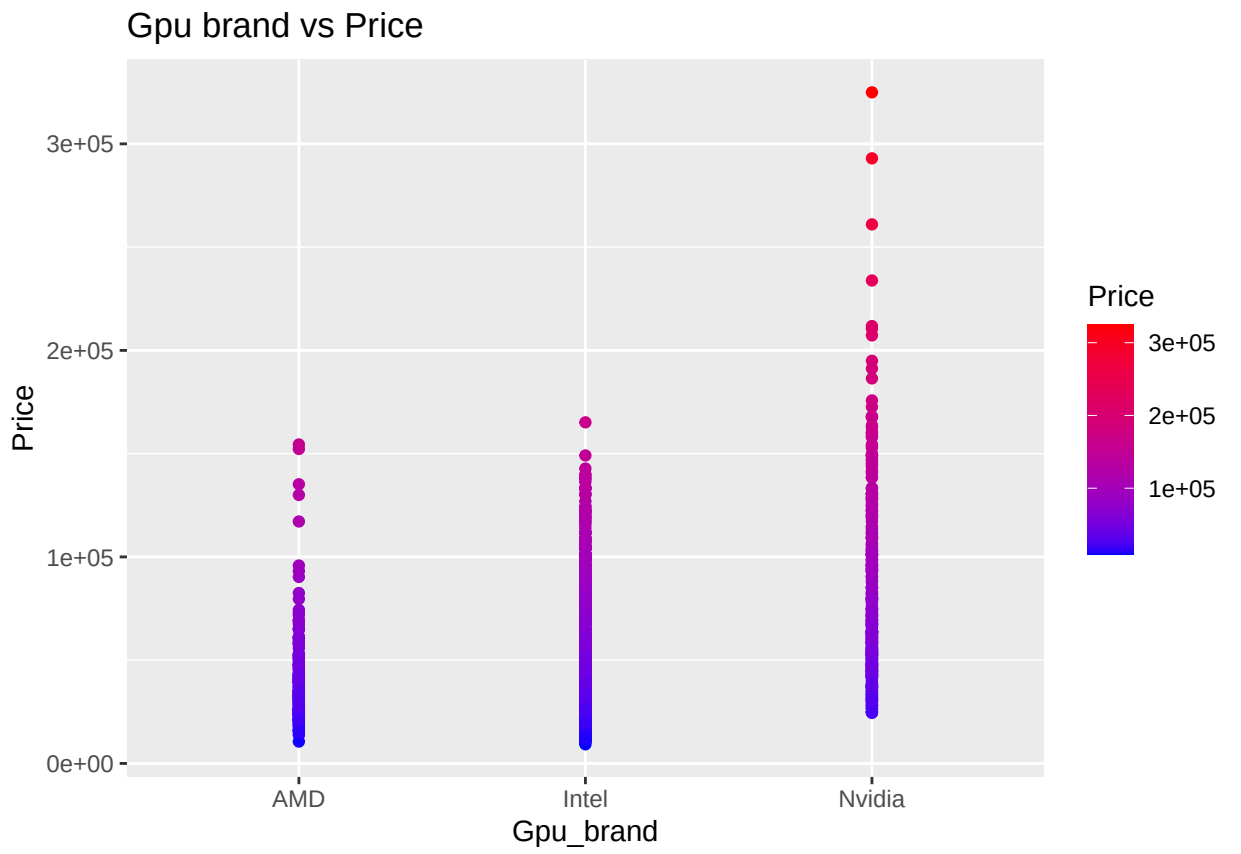

```
ggplot(data = dataset, aes(x = Ram, y = Price, color = Price)) +
  geom_point() +
  labs(x = "Ram", y = "Price", title = "Ram vs Price") +
  scale_color_gradient(low = "blue", high = "red")
```

2.How does the ram of laptop affect its price?



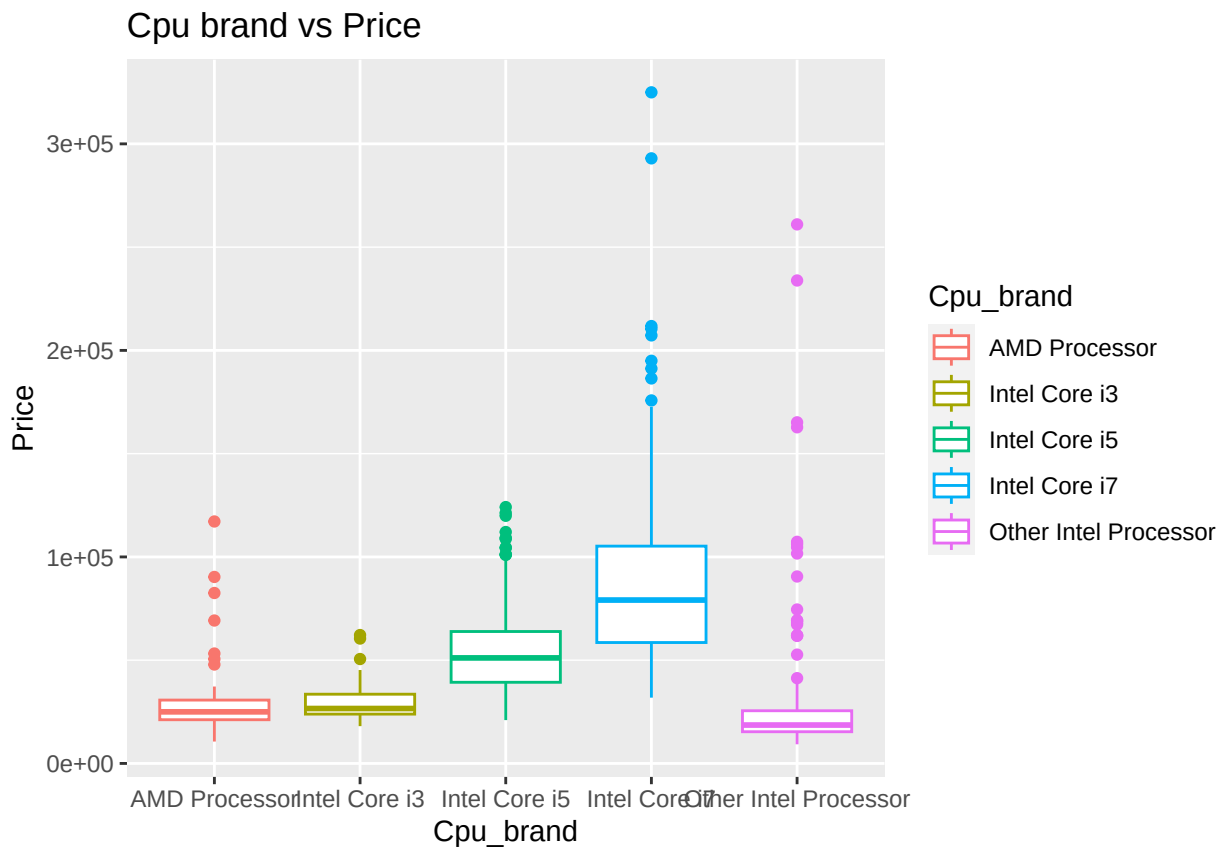

```
ggplot(data = dataset, aes(x = Gpu_brand, y = Price, color = Price)) +
  geom_point() +
  labs(x = "Gpu_brand", y = "Price", title = "Gpu brand vs Price") +
  scale_color_gradient(low = "blue", high = "red")
```

3.How does the Gpu brand of laptop affect its price?



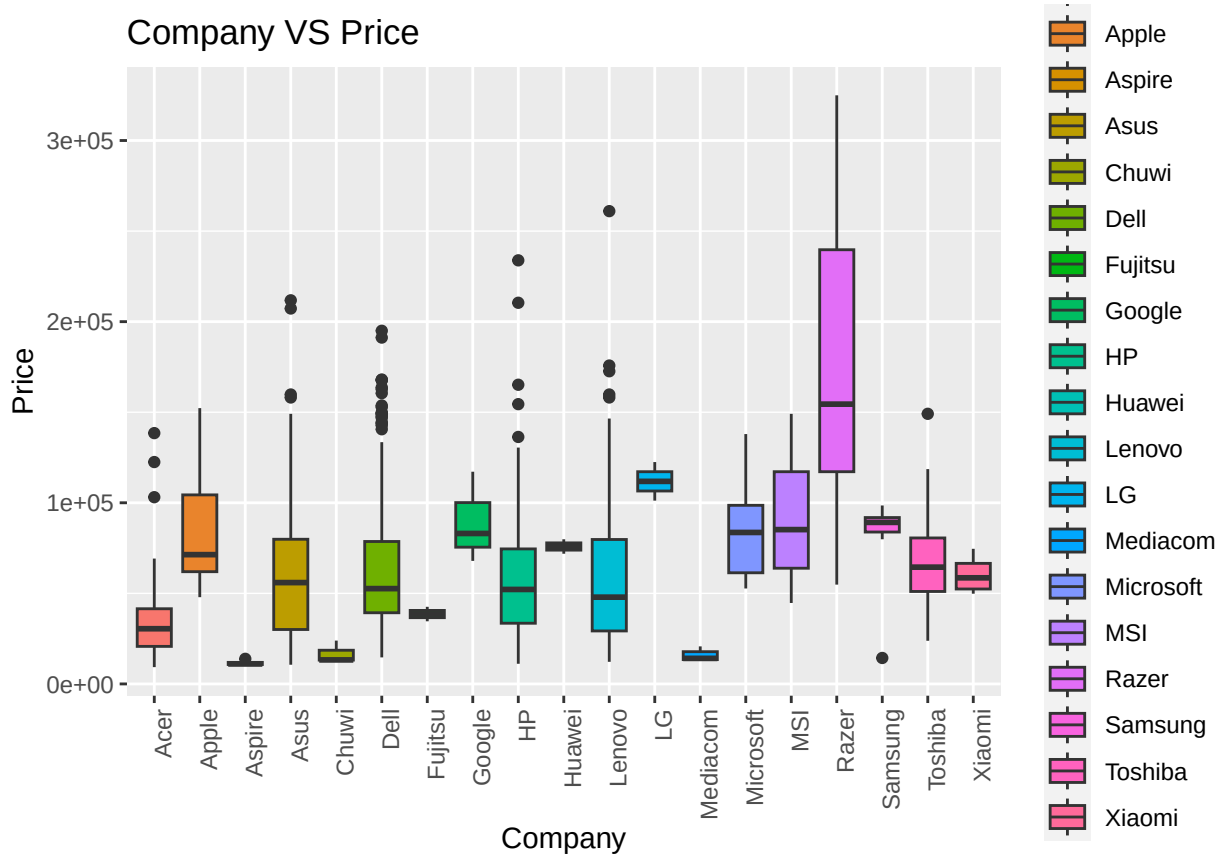

```
dataset = dataset %>%
  rename_with(~ "Cpu_brand", .cols = "Cpu brand")

ggplot(data = dataset, aes(x = Cpu_brand, y = Price, color = Cpu_brand)) +
  geom_boxplot() +
  labs(x = "Cpu_brand", y = "Price", title = "Cpu brand vs Price")
```

4.How does the Cpu brand of laptop affect its price?



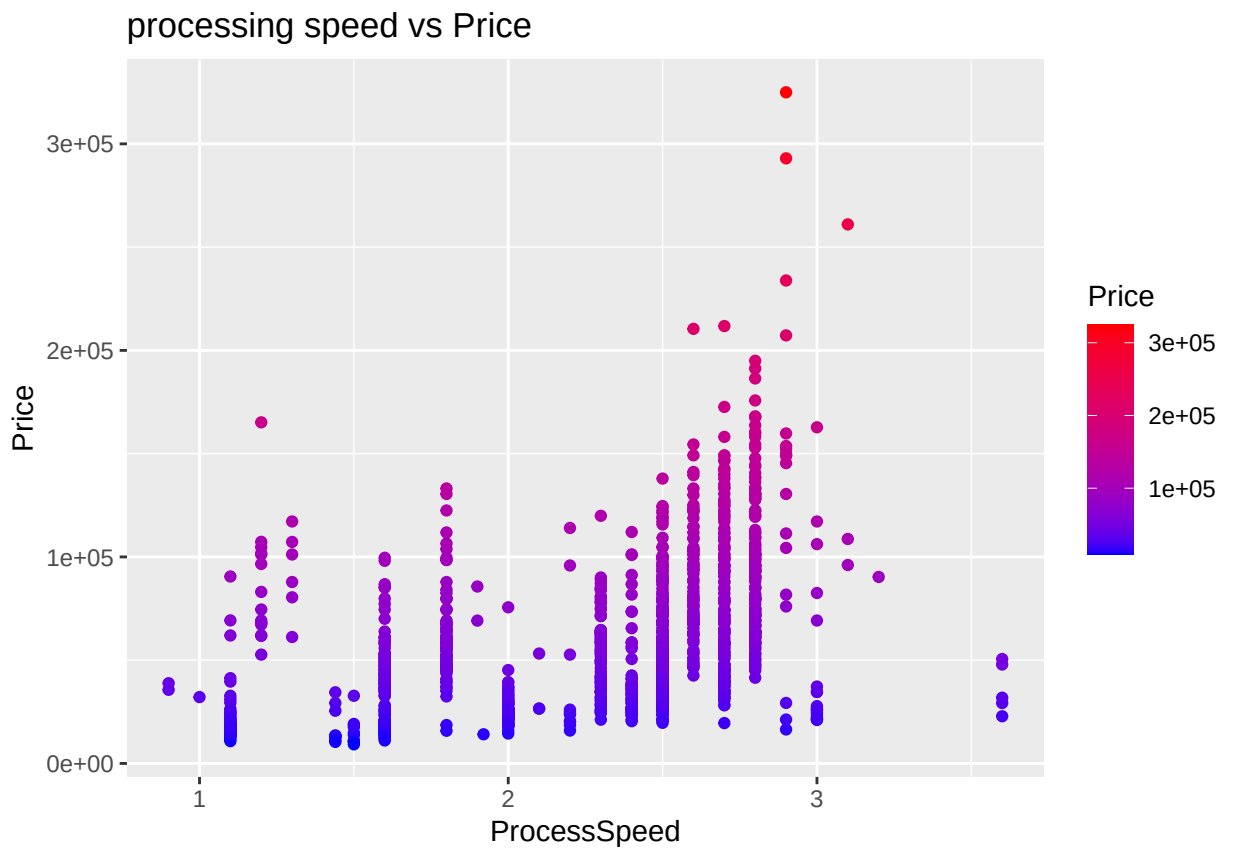

```
ggplot(data= dataset, aes(x= Company, y= Price, fill= Company))+
  geom_boxplot()+
  ggtitle("Company VS Price")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

5.Company name Vs Price



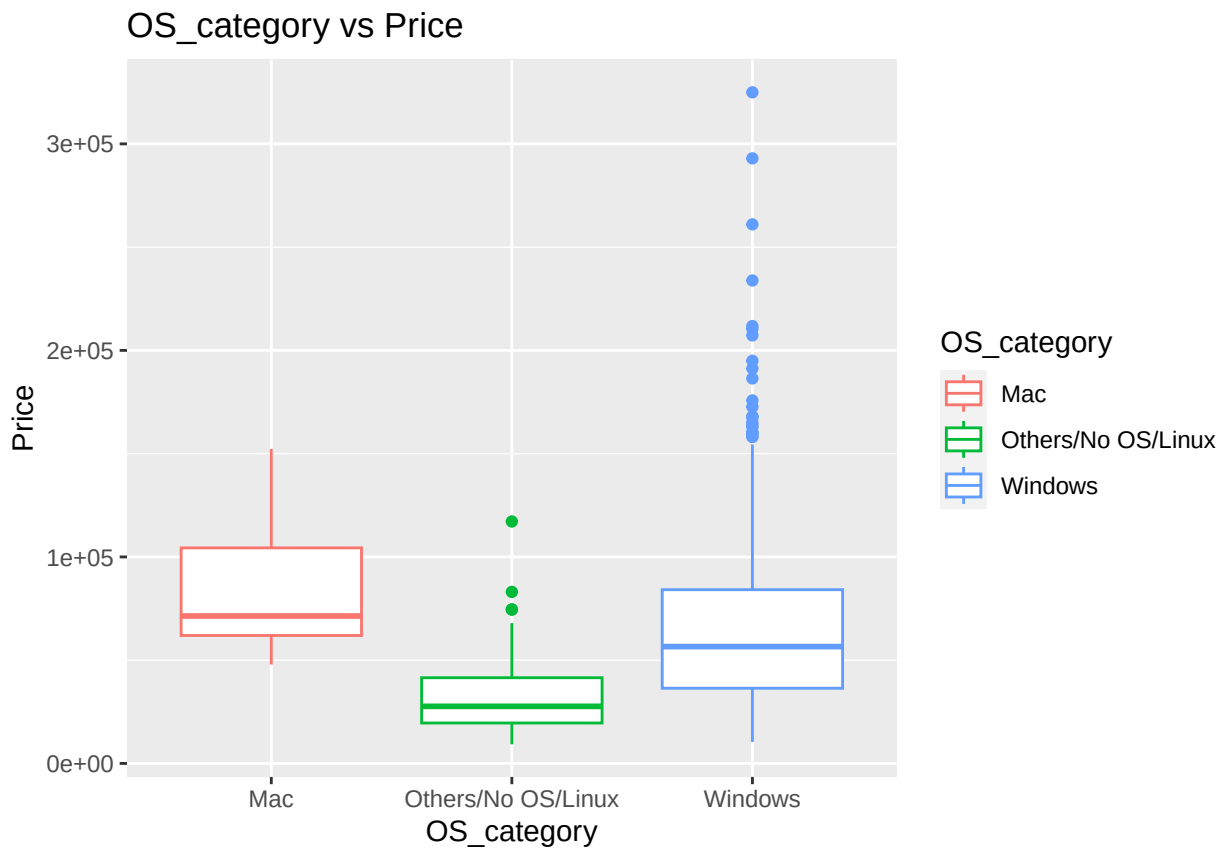

```
ggplot(data = dataset, aes(x = ProcessSpeed, y = Price, color = Price)) +
   geom_point() +
   labs(x = "ProcessSpeed", y = "Price", title = "processing speed vs Price") +
   scale_color_gradient(low = "blue", high = "red")
```

6.How does the processing speed of laptop affect its price?




```
ggplot(data = dataset, aes(x = OS_category, y = Price, color = OS_category)) +
  geom_boxplot() +
  labs(x = "OS_category", y = "Price", title = "OS_category vs Price")
```

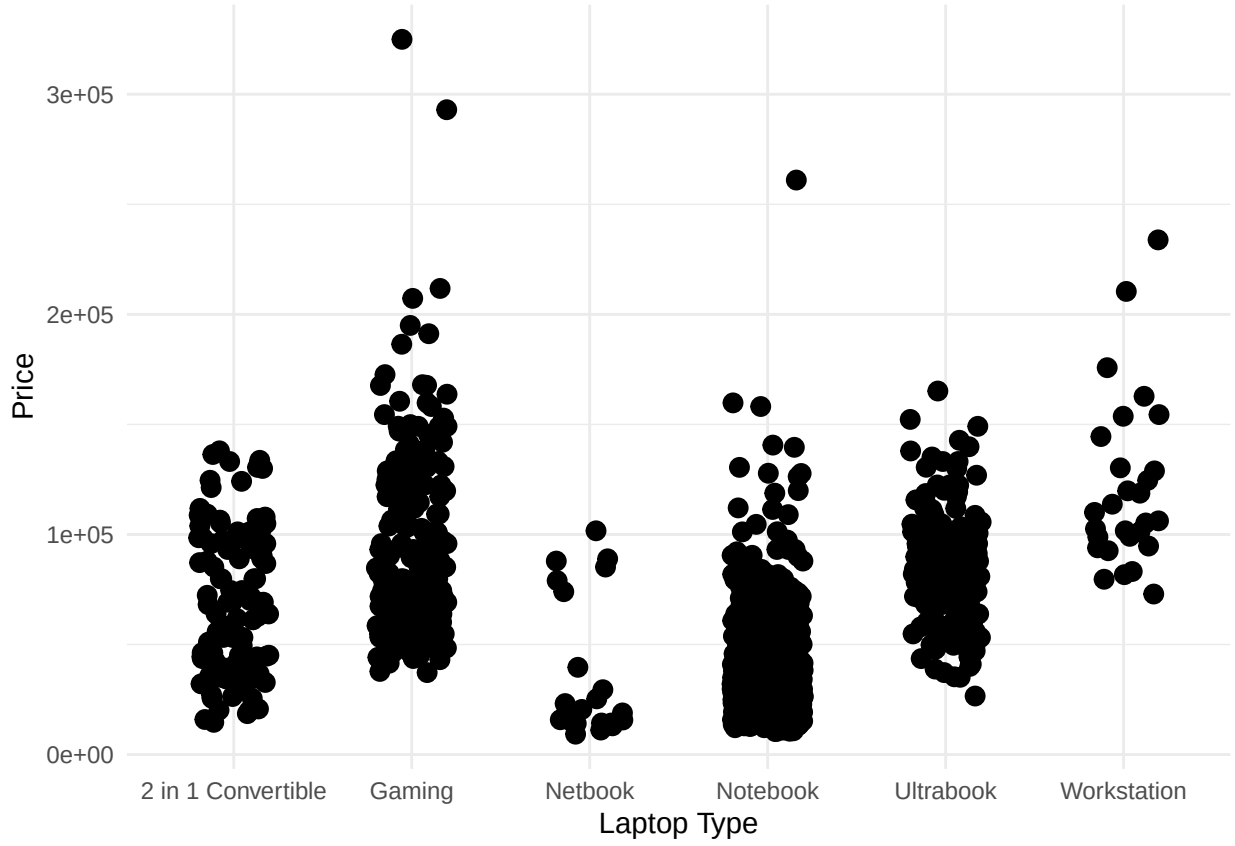
7. How does Operating system effects the price




```
ggplot(data = dataset, aes(x = TypeName, y = Price, fill = TypeName)) +
  geom_point(position = position_jitter(width = 0.2), size = 3) +
  theme_minimal() +
  labs(x = "Laptop Type", y = "Price") +
  guides(fill = FALSE) +
  theme(legend.position = "none")
```

8.How does the type of laptop affect its price?

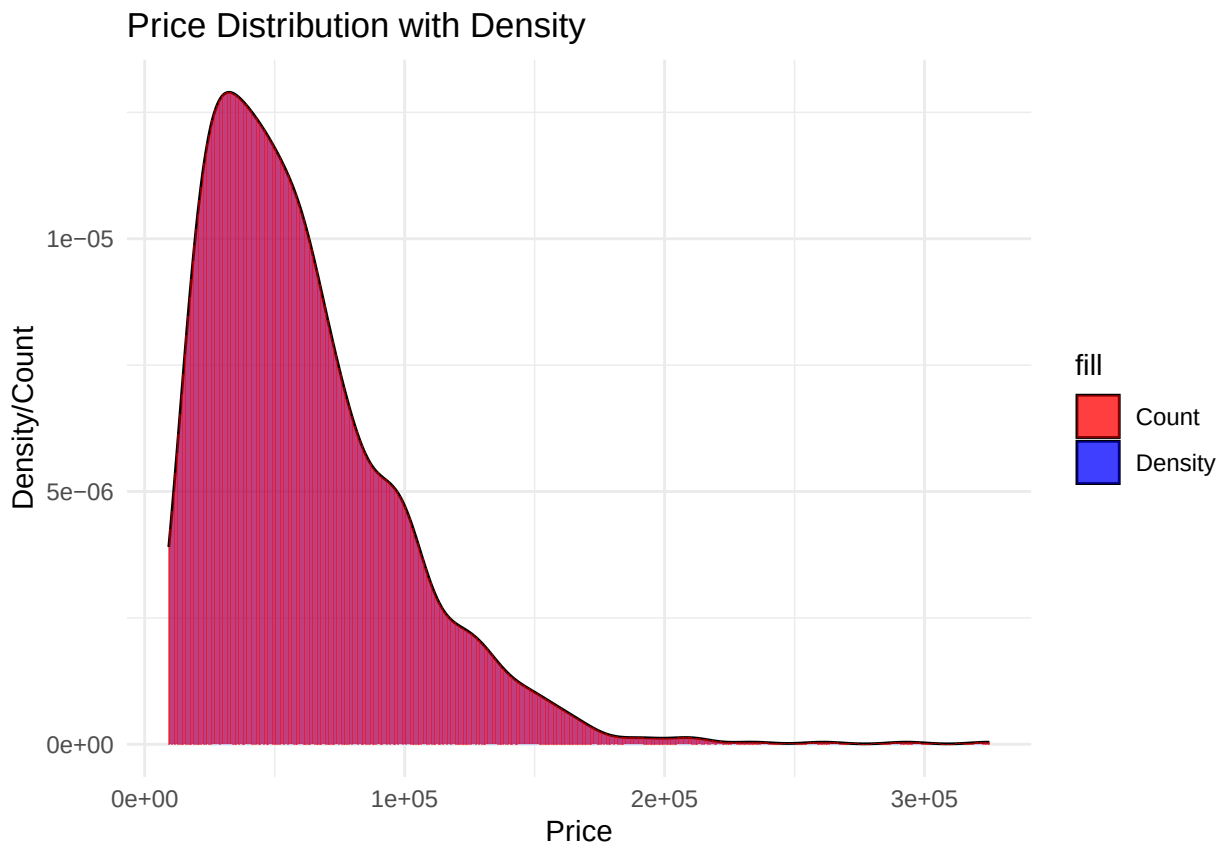
```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Price density

```
ggplot(dataset, aes(x = Price)) +
  geom_density(aes(fill = "Density"), alpha = 0.5) +
  geom_bar(aes(y = ..density.., fill = "Count"), alpha = 0.5, stat = "density") +
  scale_fill_manual(values = c("Density" = "blue", "Count" = "red")) +
  labs(title = "Price Distribution with Density", x = "Price", y = "Density/Count") +
  theme_minimal()
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

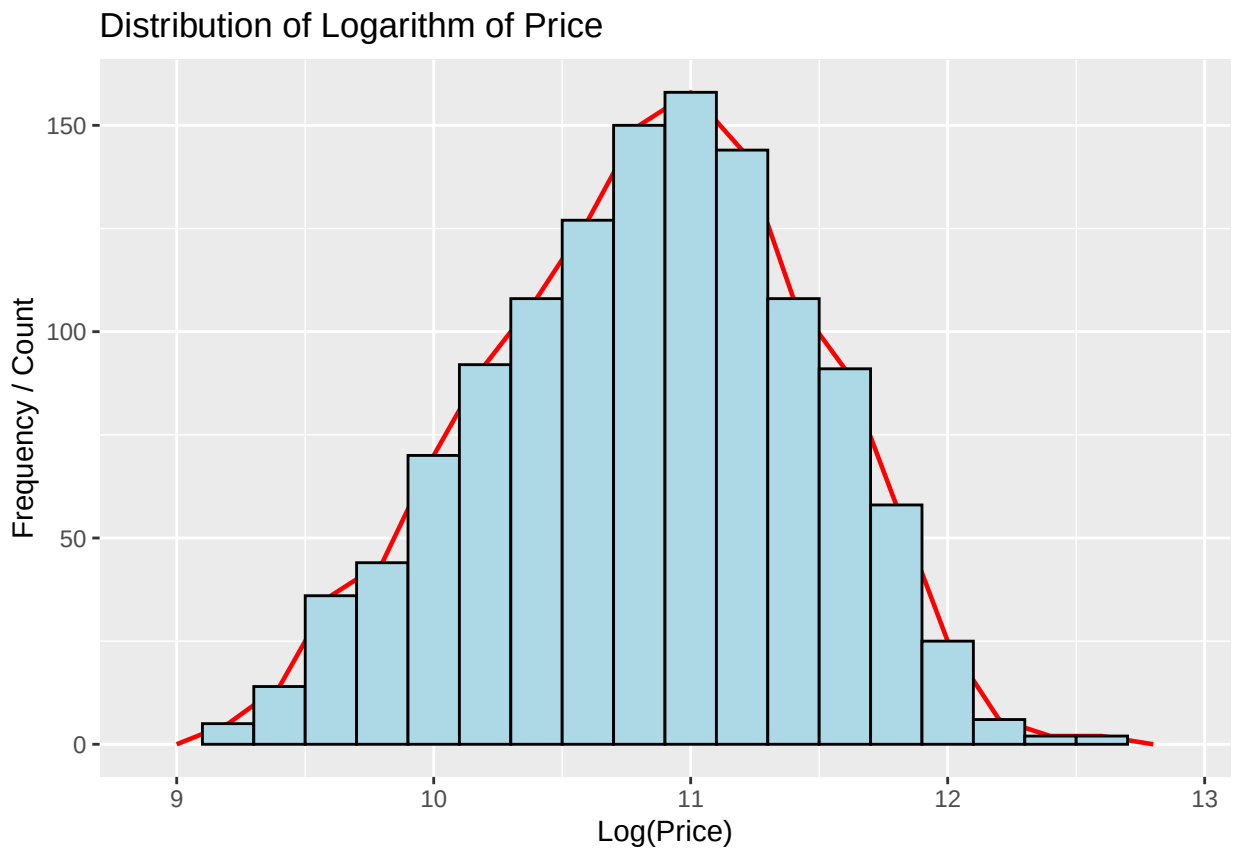


As price density is skewed and I'm trying to fit a regression model, I would like to to make a log transformation to make it normal.

```
ggplot(dataset, aes(x = log(Price))) +
  geom_freqpoly(binwidth = 0.2, size = .8, color = "red") +
  geom_histogram(binwidth = 0.2, fill = "lightblue", color = "black") +
  xlab("Log(Price)") +
  ylab("Frequency / Count") +
  ggtitle("Distribution of Logarithm of Price")
```

plot using log transformation

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```




```
dataset$Price = log(dataset$Price)
```

Using log transformation in price column

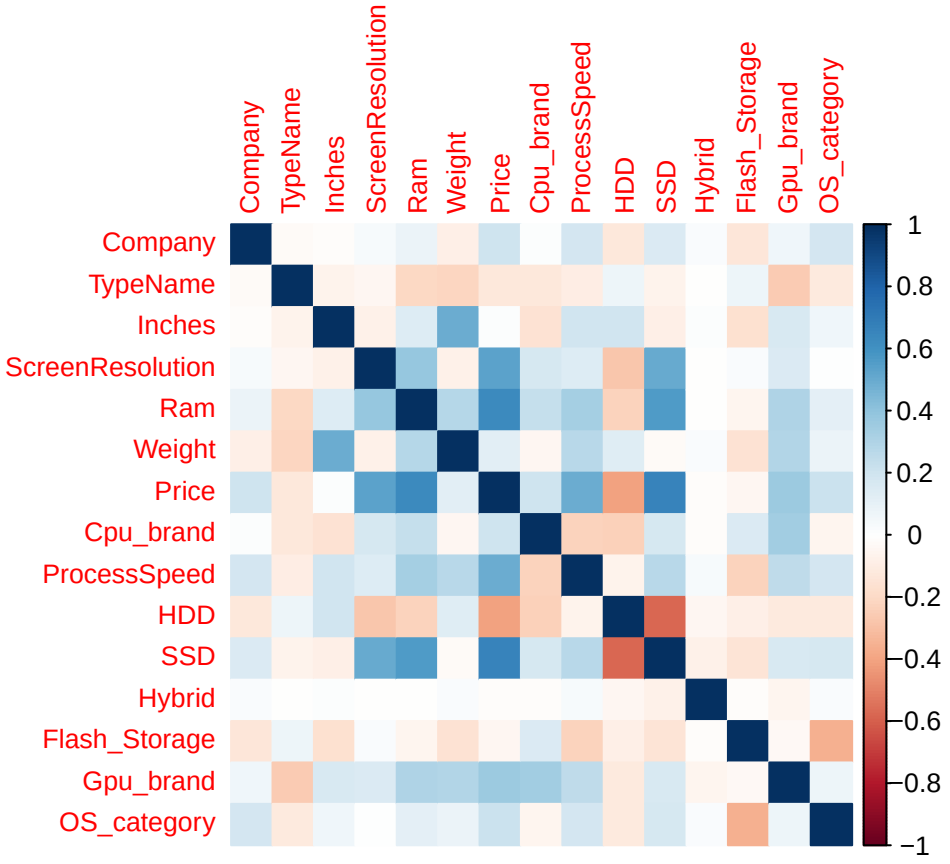
```
# Specify the categorical columns
catcols <- c("ScreenResolution", "Company", "TypeName", "OS_category", "Cpu_brand", "Gpu_brand")

# Encode categorical variables as integers using label encoding
for (col in catcols) {
  dataset[[col]] <- as.integer(factor(dataset[[col]]))
}
```

Encoding the catagorical variable

```
# Compute the correlation matrix
cor_matrix <- cor(dataset)
```

```r
# Create a heatmap of the correlation matrix
corrplot(cor_matrix, method = "color", type = "full", tl.cex = 0.8)
```

heatmap of the correlation matrix



Checking for multicollinearity

```r
vif_values <- vif(lm(Price ~ ., data = dataset))

# Print the VIF values
print(vif_values)
```

Compute the variance inflation factors (VIF)

```
##          Company         TypeName           Inches ScreenResolution
##         1.108696         1.136068         1.403210         1.433680
##              Ram           Weight       Cpu_brand     ProcessSpeed
##         1.982606         1.633193         1.494438         1.507107
##              HDD              SSD           Hybrid    Flash_Storage
##         1.712329         2.747172         1.039857         1.301244
##       Gpu_brand      OS_category
##         1.461526         1.215567
```

#----------

Building Models

- Removing outliers using robust regression

```
library(MASS)
model = rlm(Price ~ ., data = dataset)
residuals = residuals(model)


mad = median(abs(residuals - median(residuals)))
threshold = 3 * mad
outliers = which(abs(residuals) > threshold)


data_no_outliers = dataset %>%
  filter(!row_number() %in% outliers)
```

- Creating training set and test set

```
set.seed(123)
split = sample.split(data_no_outliers$Price, SplitRatio = .85)

training_set = subset(data_no_outliers, split== TRUE)
test_set = subset(data_no_outliers, split == FALSE)
y_test = test_set$Price
```

- 1. Linear regression

```
reg_1 = lm(formula = Price~.-Hybrid -Weight -TypeName,
           data = training_set)

summary(reg_1)
```

```
##
## Call:
## lm(formula = Price ~ . - Hybrid - Weight - TypeName, data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70954 -0.19807 -0.00062  0.20702  0.69805
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       8.703e+00  1.238e-01  70.289  < 2e-16 ***
## Company           8.431e-03  2.249e-03   3.750 0.000188 ***
## Inches           -1.166e-02  5.003e-03  -2.331 0.019980 *
## ScreenResolution  4.091e-02  3.570e-03  11.460  < 2e-16 ***
## Ram               3.800e-02  2.656e-03  14.307  < 2e-16 ***
## Cpu_brand         6.710e-02  1.148e-02   5.846 6.84e-09 ***
```

```
## ProcessSpeed      3.389e-01  2.239e-02  15.137  < 2e-16 ***
## HDD              -1.391e-04  2.668e-05  -5.213 2.26e-07 ***
## SSD               6.612e-04  8.282e-05   7.983 3.97e-15 ***
## Flash_Storage     1.901e-03  3.921e-04   4.847 1.45e-06 ***
## Gpu_brand         5.873e-02  1.601e-02   3.668 0.000258 ***
## OS_category       1.820e-01  2.544e-02   7.153 1.66e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2823 on 979 degrees of freedom
## Multiple R-squared:  0.7645, Adjusted R-squared:  0.7618
## F-statistic: 288.9 on 11 and 979 DF,  p-value: < 2.2e-16
```

```
#printing adjusted R-squared

summary(reg_1)$adj.r.squared
```

```
## [1] 0.7618109
```

```
y_pred = predict(reg_1, newdata= test_set)
```

- 2.SVR

```
reg_2 = svm(formula= Price~.,
            data= training_set,
            type= 'eps-regression',
            kernel= 'radial',
            sigma= 0.1,
            C = 1)
#Prediction

y_pred = predict(reg_2, newdata= test_set)

# Calculate R-squared score
r2_score = R2(y_test, y_pred)

# Calculate mean absolute error
mae = MAE(y_test, y_pred)

# Print R-squared score and mean absolute error
print(paste("R2 score:", r2_score))
```

```
## [1] "R2 score: 0.887999619892711"
```

```
print(paste("MAE:", mae))
```

```
## [1] "MAE: 0.162655355829982"
```

- 3.Decision Tree

```
reg_3 = rpart(formula = Price~.,
              data = training_set,
               control = rpart.control(minsplit = 50, cp=0.01),
              )

#Prediction

y_pred = predict(reg_3, newdata= test_set)

# Calculate R-squared score
r2_score = R2(y_test, y_pred)

# Calculate mean absolute error
mae = MAE(y_test, y_pred)

# Print R-squared score and mean absolute error
print(paste("R2 score:", r2_score))
```

```
## [1] "R2 score: 0.801885579007813"
```

```
print(paste("MAE:", mae))
```

```
## [1] "MAE: 0.21374722793359"
```

- 4.Random Forest

```
set.seed(1234)
reg_4 = randomForest(
  x= training_set[-7],
  y= training_set$Price,
  ntree= 200,
  mtry = 4,

)

#prediction

y_pred = predict(reg_4, newdata = test_set)

#calculate R2 score

r2_score = R2(y_pred, y_test)

#calculate MAE score

mae = MAE(y_pred, y_test)

#print R2 and mae score

print(paste("R2 score:", r2_score))
```

```
## [1] "R2 score: 0.921137585064212"
```

```r
print(paste("MAE Score:", mae))
```

```
## [1] "MAE Score: 0.141683922297544"
```

- 5.XGBoost

```r
reg_5 = xgboost(data = as.matrix(training_set[-7]), label = training_set$Price, nrounds = 50)
```

```
## [1]  train-rmse:7.259380
## [2]  train-rmse:5.094205
## [3]  train-rmse:3.578917
## [4]  train-rmse:2.519289
## [5]  train-rmse:1.778307
## [6]  train-rmse:1.260818
## [7]  train-rmse:0.901709
## [8]  train-rmse:0.653244
## [9]  train-rmse:0.483688
## [10] train-rmse:0.368500
## [11] train-rmse:0.293643
## [12] train-rmse:0.245416
## [13] train-rmse:0.212294
## [14] train-rmse:0.193705
## [15] train-rmse:0.180477
## [16] train-rmse:0.170195
## [17] train-rmse:0.164428
## [18] train-rmse:0.158178
## [19] train-rmse:0.153811
## [20] train-rmse:0.146430
## [21] train-rmse:0.142420
## [22] train-rmse:0.139965
## [23] train-rmse:0.137381
## [24] train-rmse:0.133081
## [25] train-rmse:0.131318
## [26] train-rmse:0.129510
## [27] train-rmse:0.128260
## [28] train-rmse:0.127628
## [29] train-rmse:0.123437
## [30] train-rmse:0.122402
## [31] train-rmse:0.121512
## [32] train-rmse:0.118727
## [33] train-rmse:0.116419
## [34] train-rmse:0.115439
## [35] train-rmse:0.115012
## [36] train-rmse:0.111101
## [37] train-rmse:0.109598
## [38] train-rmse:0.107908
## [39] train-rmse:0.105090
## [40] train-rmse:0.104302
## [41] train-rmse:0.103372
## [42] train-rmse:0.102787
## [43] train-rmse:0.102577
## [44] train-rmse:0.101548
```

```
## [45] train-rmse:0.099780
## [46] train-rmse:0.096996
## [47] train-rmse:0.096279
## [48] train-rmse:0.095422
## [49] train-rmse:0.093340
## [50] train-rmse:0.092720
```

```
y_pred = predict(reg_5, newdata =as.matrix(test_set[-7]))


#calculate R2 score

r2_score = R2(y_pred, y_test)

#calculate MAE score

mae = MAE(y_pred, y_test)

#print R2 and mae score

print(paste("R2 score:", r2_score))
```

```
## [1] "R2 score: 0.925263093342881"
```

```
print(paste("MAE Score:", mae))
```

```
## [1] "MAE Score: 0.131715662506014"
```

- Comparison between True value and predicted value using XGBoost

```
comparison = data.frame(predicted= exp(y_pred), True= exp(y_test))
print(comparison)
```

```
##     predicted      True
## 1   109989.48 135195.34
## 2    93425.23  96095.81
## 3    58280.26  61735.54
## 4    23521.18  20986.99
## 5    51411.16  39693.60
## 6    52284.80  53226.72
## 7    14572.50  13746.24
## 8    25779.93  22305.14
## 9    55409.24  53173.44
## 10   22298.59  19553.76
## 11   38364.31  31232.20
## 12   23522.64  23373.40
## 13   22410.92  25840.80
## 14   35949.01  30742.56
## 15   66505.67  66546.72
## 16   66642.23  57755.52
## 17   35176.28  30049.92
```

```
## 18   47400.31  59567.04
## 19   80170.21  67718.88
## 20   11501.92  14811.31
## 21   30034.51  23922.72
## 22   72219.66  37242.72
## 23   54024.86  58554.72
## 24  108864.20 100699.20
## 25   25154.94  23816.16
## 26   75165.28  95850.72
## 27   46158.07  41505.12
## 28   45670.16  48697.92
## 29   55539.54  51095.52
## 30   69806.83  74485.44
## 31   23589.61  23389.92
## 32  104147.17  74964.96
## 33   35480.55  29250.72
## 34   35667.24  36089.21
## 35  142491.25 130873.80
## 36  166904.49 207259.20
## 37   30201.48  31381.92
## 38   39755.74  35964.00
## 39   54396.73  79866.72
## 40   49312.08  54239.04
## 41   43094.28  45234.72
## 42   48209.52  45767.52
## 43   23362.43  22803.84
## 44   37152.10  30849.12
## 45   58913.34  63243.36
## 46  154435.63 153705.34
## 47   15633.71  14652.00
## 48   18462.45  24503.47
## 49   49269.40  60885.72
## 50   36525.08  38148.48
## 51   96364.58 111834.72
## 52   61900.95  69477.12
## 53   82912.29  74751.84
## 54   27207.15  26586.72
## 55   25047.08  22697.28
## 56   25044.26  29463.84
## 57   78340.26  63456.48
## 58  110691.35  93181.39
## 59   72099.51  72940.32
## 60   58850.95  49976.64
## 61   17750.83  16303.68
## 62  108864.20 119826.72
## 63  107044.75 102564.00
## 64   27008.86  31909.39
## 65   63271.07  61751.52
## 66   49786.13  60867.07
## 67   71466.96 106506.72
## 68   23464.03  18594.72
## 69  114429.44 105228.00
## 70   47970.03  52693.92
## 71   99026.08  95850.72
```

```
## 72   27571.05  23176.80
## 73   79705.87  67932.00
## 74   90378.21 101178.72
## 75   72303.80  71928.00
## 76   97078.43 158135.04
## 77   64826.12  68184.01
## 78   27412.82  36709.92
## 79   39655.62  31168.80
## 80   74996.44  61005.60
## 81   35682.59  41824.80
## 82  109029.61 104695.20
## 83   93549.69  85194.72
## 84   20147.55  18328.32
## 85   18107.11  15717.60
## 86  117170.88  79813.44
## 87   91573.15  90522.72
## 88   30127.78  35616.61
## 89   21151.58  18594.72
## 90   23905.92  23650.99
## 91   71448.97  62231.04
## 92   32858.58  41345.28
## 93   44862.27  49656.96
## 94   30225.02  34046.45
## 95   25668.75  30849.12
## 96   85698.30  93635.34
## 97  122864.78 130536.00
## 98   44882.98  48964.32
## 99   17741.64  22324.32
## 100  70153.47  79215.11
## 101  47470.57  39693.60
## 102  20920.53  21951.36
## 103  95754.10  99519.05
## 104  72278.16  60472.80
## 105  18247.66  22324.32
## 106  29088.69  32639.86
## 107 107293.72 101657.71
## 108  48895.49  42357.60
## 109  25559.68  34898.93
## 110  60323.77  66546.72
## 111  99134.46  76012.44
## 112  80086.00  79866.72
## 113  34841.08  35431.20
## 114  74462.43  81784.80
## 115 102856.59  99047.52
## 116  32233.79  34898.40
## 117  22415.43  17529.12
## 118 123471.11 122490.72
## 119 104249.42 104961.60
## 120  42144.08  43156.80
## 121  38819.72  39373.92
## 122  86001.14  93186.72
## 123  31781.36  41558.40
## 124 113068.47  95850.72
## 125  90940.71  99900.00
```

```
## 126  26402.06  35644.32
## 127  44313.14  34578.72
## 128  21370.52  24988.32
## 129  77902.12  98514.72
## 130  15922.14  18541.44
## 131  80086.00  95850.72
## 132  44619.66  42037.92
## 133  69293.92  71874.72
## 134  81533.42  87912.00
## 135  17750.83  16463.52
## 136  71481.48  88924.32
## 137  72396.74  77788.80
## 138  75893.50  67559.04
## 139  84560.33  93772.80
## 140  20396.38  24808.23
## 141  62365.51  63669.60
## 142  46030.90  48751.20
## 143  57016.04  59620.32
## 144  79463.90  71874.72
## 145  81890.01 111301.92
## 146  88081.42  77202.72
## 147  51694.84  64468.80
## 148  27753.58  42410.35
## 149  63220.88  63159.71
## 150  46979.67  55754.32
## 151  52526.69  63349.92
## 152  51714.56  56633.98
## 153  71713.56  75924.00
## 154  55181.01  58288.32
## 155  78687.61  77682.24
## 156  59989.25  58075.20
## 157  43049.87  35111.52
## 158  47921.47  42570.72
## 159  42933.31  34035.26
## 160  70491.88  84129.12
## 161  92872.24 124621.92
## 162  35941.22  31914.72
## 163  52050.17  44701.92
## 164  17900.32  19660.32
## 165  24927.00  21205.44
## 166  18012.59  14492.16
## 167  17024.66  15930.72
## 168  61670.33  68145.12
## 169  88742.62  93186.72
## 170  67555.88  78647.14
## 171  98799.78  87912.00
## 172  94150.25  96596.64
## 173 105345.58 101232.00
## 174  68817.84  65481.12
## 175  17873.33  15397.92
```

CONCLUSION

our project focused on predicting laptop prices using various machine learning models such as multi-linear regression, Support Vector Regression (SVR), decision tree, random forest, and XGBoost. After extensive analysis and evaluation, we found that the XGBoost model outperformed the other models in terms of accuracy and predictive power.

The XGBoost model demonstrated superior performance by effectively capturing the complex relationships between the laptop features and their corresponding prices. Its ability to handle non-linear relationships and feature interactions allowed it to make more accurate predictions compared to the other models.

While multi-linear regression, SVR, decision tree, and random forest models also provided reasonable results, the XGBoost model consistently exhibited higher accuracy and better overall performance. Its ensemble-based approach and optimization techniques enabled it to effectively handle both numerical and categorical features, providing more robust predictions.

It's worth noting that the choice of the most suitable model may depend on factors such as the size of the dataset, the specific characteristics of the laptop features, and the desired trade-off between interpretability and accuracy. However, in our project, the XGBoost model emerged as the most accurate and reliable choice.

The findings of our project highlight the significance of utilizing advanced machine learning algorithms, such as XGBoost, for predicting laptop prices. These models can offer valuable insights to consumers, retailers, and manufacturers, aiding in decision-making processes related to pricing, marketing, and product development.

Overall, our project demonstrates that the XGBoost model is a powerful tool for accurately predicting laptop prices, providing a foundation for further research and application in the domain of laptop pricing analysis.