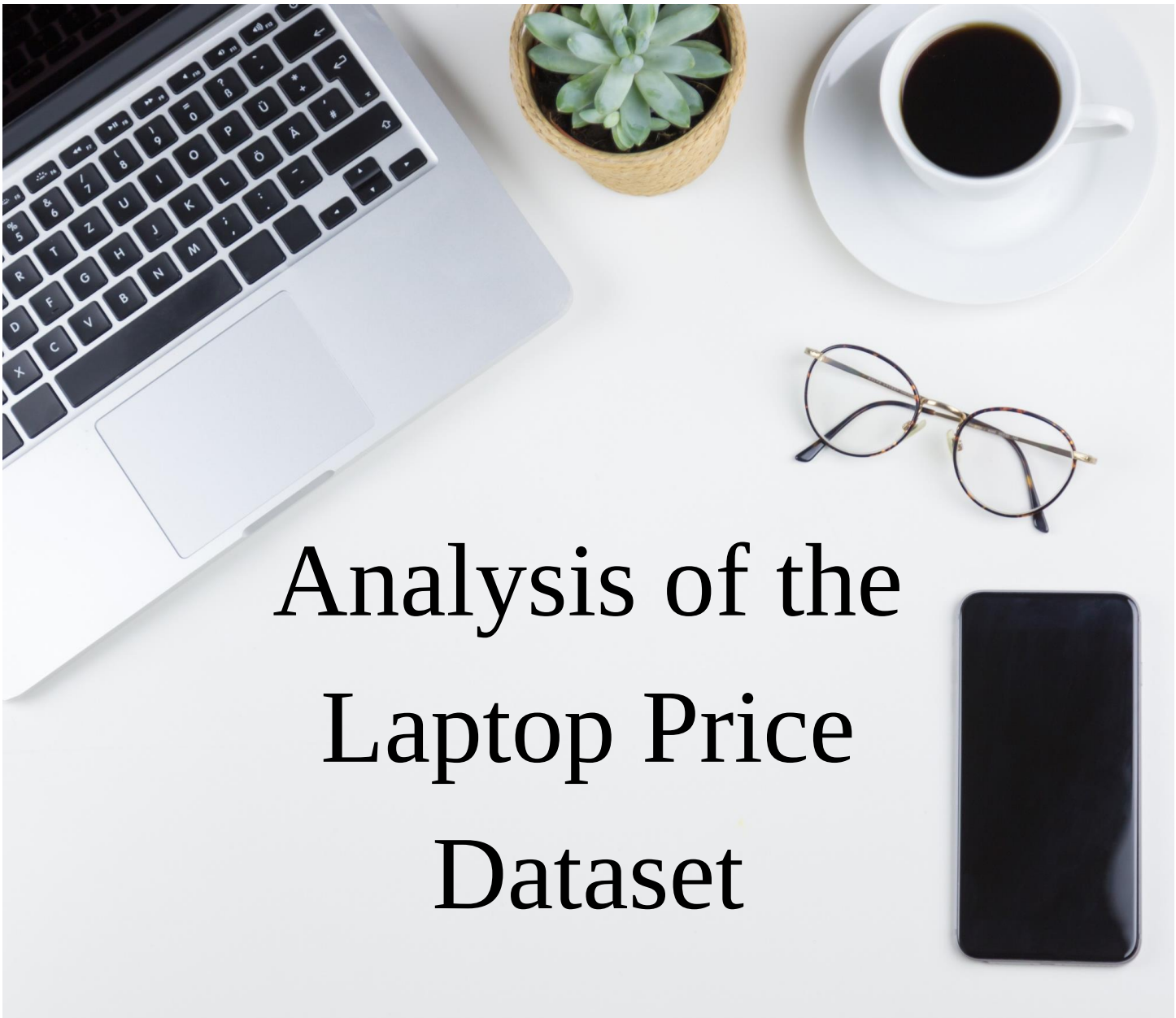


Analysis of the Laptop Price Dataset

Data Analysis Project 2

Group 4

M. D. D. De Costa s14533

K. G. S. K. Wickramasinghe s14594

S. P. P. M. Sudasinghe s14510

Contents

List of Tables

Introduction

In the new normal laptops are an essential part in our day to day lives. Everyone through all the ages uses a laptop in this day and time. The laptops are more popular than the traditional desktop, mostly due to its portability.

Description of the problem

Each different laptop has its own different software and hardware that causes the price of the laptop to fluctuate. Our focus is to analyze, identify and compare the key factors that affect this change in prices and to predict how these different factors affect the price of a laptop.

Objectives

To develop a predictive model, that will predict the price of a laptop, when its features are given.

Description of the dataset

The laptop prices dataset has 1303 records and 12 variables of different laptops.

```
'data.frame':   1275 obs. of  12 variables:
 $ Company         : Factor w/ 19 levels "Acer","Apple",..: 2 2 8 2 2 1 2 2 3 1 ...
 $ Product         : chr  "MacBook Pro" "Macbook Air" "250 G6" "MacBook Pro" ...
 $ 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: chr  "IPS Panel Retina Display 2560x1600" "1440x900" "Full HD 1920x1080" "IPS Panel Retina Display 2880x1800" ...
 $ Cpu             : chr  "Intel Core i5 2.3GHz" "Intel Core i5 1.8GHz" "Intel Core i5 7200U 2.5GHz" "Intel Core i7 2.7GHz" ...
 $ 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 620" "AMD Radeon Pro 455" ...
 $ OpSys           : Factor w/ 9 levels "Android","Chrome OS",..: 5 5 6 5 5 7 4 5 7 7 ...
 $ Weight          : chr  "1.37kg" "1.34kg" "1.86kg" "1.83kg" ...
 $ Price_euros     : num  1340 899 575 2537 1804 ...
```

Data Cleaning

By analysis techniques in Rstudio, we found out that there are no missing values in any of the variables. So, data imputation has not been done. Also, we managed to find out that there are 28 duplicate records in our dataset, that we removed. So, in the final dataset we have 1275 records.

From the above Rstudio output we can see that the variables ‘Ram’ and ‘Weight’ are actually quantitative variables, but they are characterized as character variables due to the existence of the words in the variable. So, in data preprocessing we removed the letters and made them numeric variables.

- Removal of the letter’s ‘GB’ from variable ‘Ram’
- Removal of the letter’s ‘kg’ from variable ‘Weight’

Feature Engineering and Preprocessing

To make better sense with our data we manipulated some of the variables so that they give meaningful correlation with the price.

- Creation of 4 variables namely ‘SSD’, ‘HDD’, ‘Hybrid’ and ‘FlashStorage’ from the variable ‘Memory’.
- Creation of ‘Xres’ and ‘Yres’ from ‘ScreenResolution’.
- Creation of Pixels Per Inch (PPI) from the variables ‘Xres’, ‘Yres’ and ‘Inches’.
- Presence of ‘IPS’ screen from variable ‘ScreenResolution’
- Presence of touch screen from variable ‘ScreenResolution’
- Value of clock speed from variable ‘Cpu’
- Creation of ‘CpuBrand’ variable from ‘Cpu’
- Creation of ‘GpuBrand’ variable from ‘Gpu’
- Transforming variable ‘OpSys’ so that it gives more meaning.

After extracting all the data from the initial variables into more meaningful variables, we decided to remove some of the initial variables. So, finally we have 16 variables.

```
'data.frame':   1275 obs. of  16 variables:
 $ Company     : Factor w/ 19 levels "Acer","Apple",..: 2 2 8 2 2 1 2 2 3 1 ...
 $ TypeName    : Factor w/ 6 levels "2 in 1 Convertible",..: 5 5 4 5 5 4 5 5 5 5 ...
 $ Ram         : int  8 8 8 16 8 4 16 8 16 8 ...
 $ OpSys       : Factor w/ 5 levels "Linux","Mac",..: 2 2 3 2 2 5 2 2 5 5 ...
 $ Weight      : num  1.37 1.34 1.86 1.83 1.37 2.1 2.04 1.34 1.3 1.6 ...
 $ Price_euros : num  1340 899 575 2537 1804 ...
 $ IPS         : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 1 2 1 1 2 ...
 $ TouchScreen : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ PPI         : num  227 128 141 221 227 ...
 $ cpuBrand    : Factor w/ 5 levels "AMD Processor",..: 3 3 3 4 3 1 4 3 4 3 ...
 $ ClockSpeed  : num  2.3 1.8 2.5 2.7 3.1 3 2.2 1.8 1.8 1.6 ...
 $ SSD         : num  128 0 256 512 256 0 0 0 512 256 ...
 $ HDD         : num  0 0 0 0 0 500 0 0 0 0 ...
 $ FlashStorage: num  0 128 0 0 0 0 256 256 0 0 ...
 $ Hybrid      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ GpuBrand    : Factor w/ 4 levels "AMD","ARM","Intel",..: 3 3 3 1 3 1 3 3 4 3 ...
```

Important results of the descriptive analysis

- It was observed that the distribution of the response variable; laptop price in euros is positively skewed, indicating that the laptops with low prices are sold and purchased more than the branded ones.
- Market price of laptops is mostly controlled by the Lenovo, Dell and HP companies since these are the highest selling laptops currently.
- There's a moderately strong relationship with the PPI (Pixels Per Inch) and the response variable (Laptop Price).
- Laptops with SSD are expensive than others. Cheapest option we have laptops with a Flash Storage. Since the hybrid storage is a combination of SSD and HDD, laptops with a hybrid storage are expensive than laptops with an HDD.
- Razer, Apple, LG, Microsoft, Google, MSI laptops are expensive, and others are in the inexpensive range. Since Razer is a gaming pc brand, it is the most expensive one.
- Laptops with touch screens are expensive which is true in real life.
- Laptop prices will depend on the processors. And as obvious the price of i7 processors is high, then of i5 processor, i3 and AMD processor lies at the almost same range.
- Higher RAM capacities reflect higher prices in laptops. Price is having a very strong positive correlation with RAM.
- As usual laptops with Mac operating system are most expensive.
- It was observed some variables which are highly correlated with each other, hence assumed that multicollinearity might exist among data.

Important results of the advanced analysis

As we saw in our descriptive analysis, the dataset contains of outlier observations and our response variable is highly skewed. So, we will be running each model for Y1,Y2,Y3,Y4.

- Y1 = Price_euros with outlier data points
- Y2 = Price_euros without outlier data points
- Y3 = log(Price_euros) with outlier data points
- Y4 = log(Price_euros) without outlier data points

With the objective of predicting the price of laptops(in Euros), we will be running ridge regression, lasso regression, elastic net regression and random forest regression for Y1,Y2,Y3,Y4. So, in total we will be using 16 models to find the best fit model.

Ridge, Lasso and Elastic Net Models

First, let's look at the Ridge, Lasso and Elastic Net models for the above 4 differently transformed Y value.

Transformation	Model	λ-Best	Train RMSE	Test RMSE
Y1 = Price_euros with outlier data points	Ridge	53.30748	332.6857	306.1911
	Lasso	0.3760737	330.3914	306.1911
	Elastic Net	16.20452	331.7613	305.091
Y2 = Price_euros without outlier data points	Ridge	44.8544	291.1064	303.8303
	Lasso	2.95112	290.6653	303.8303
	Elastic Net	5.90224	290.7309	300.7845
Y3 = log(Price_euros) with outlier data points	Ridge	0.04286539	397.1269	333.2663
	Lasso	0.0005799892	400.4345	327.8525
	Elastic Net	0.001056929	400.262	337.7982
Y4 = log(Price_euros) without outlier data points	Ridge(1SE)	0.1313569	312.2822	349.1623
	Lasso(1SE)	0.005956878	312.9285	360.3786
	Elastic Net	0.0004591	310.9809	370.0219

Table 1

By looking at the above results we can clearly see that the model Elastic Net for Y2 has the best fit (lowest test RMSE).

So now, we will move onto the Random Forest Regression model.

Random Forest Regression Model

After looking at the results of Ridge, Lasso and Elastic Net models now, we will be using Random Forest regression on our 4 sets of data.

In Random Forest we set multiple decision trees in parallel. So, even though each individual decision tree has a higher variance, the Random Forest model has a lower variance.

Let's look at how the Random Forest Model worked for our 4 sets of data.

Model	Train RMSE	Test RMSE
Y1 = Price_euros with outlier data points	142.7174	230.2287
Y2 = Price_euros without outlier data points	129.2734	250.1811
Y3 = log(Price_euros) with outlier data points	167.3005	241.2828
Y4 = log(Price_euros) without outlier data points	137.1447	273.5084

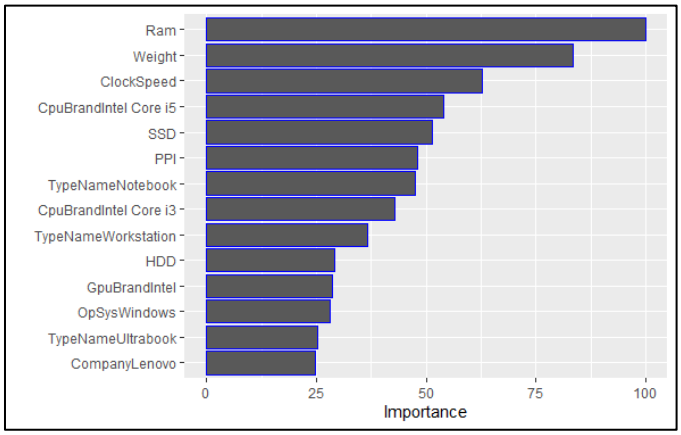
Table 2

By looking at the train RMSE values, we can clearly see that the Random Forest model built with Y1 data is the best model from the 4 Random Forest Regression models that we ran.

Then we looked at the MAPE value of the model with Y1. We got the value of 15.09% which is a good value.

Best Model Selection

By comparing the Test MSE of all the 16 models that we ran, we can see that the smallest is for the Random Forest Model with Y1 data. Therefore, we choose that model as our best model and use that model to build our data product.



Issues Encountered and Solutions

- Problem: The test MSE remained high after fitting the multiple linear regression with regularized techniques.
  Solution: Use more powerful machine learning algorithms as Random Forest

- Problem: Presence of outliers in the dataset and the response variable being skewed.
  Solution: Running all 16 Machine Learning algorithms for Y1,Y2,Y3,Y4. (Hipson, 2019)

Discussion and Conclusions

This study was conducted to model how the laptop prices differ with different factors and to build a predictive model. According to the initial descriptive analysis we saw that there were some outliers in the dataset, presence of multicollinearity among some variables in the data set and the response variable was highly skewed. So, we ran 16 models, and we selected the best one. According to the Test RMSE value, the best model out of all is the Random Forest model with outliers without log transformation.

The Random Forest model with 10-fold cross validation gave the Test RMSE of 230.2287 when the number of trees ("ntree") is 500 and number of variables ("mtry") is 44 and the 81.35% of the total variance explained by the fitted model.

Furthermore, the most accurate Random Forest model for Y1 was used to develop the data product application which can be used to get the predictions of laptop price in Euros. The data product was deployed on ShinyApps.io

Bibliography

(n.d.). Retrieved from shiny.rstudio: https://shiny.rstudio.com/tutorial/

(n.d.). Retrieved from https://www.youtube.com/watch?v=0SbqhCQvj6w&t=327s

Agrawal, R. (2021, 11). *laptop-price-prediction-practical-understanding-of-machine-learning-project-lifecycle*. Retrieved from analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/11/laptop-price-prediction-practical-understanding-of-machine-learning-project-lifecycle/

Boehmke, B. C. (n.d.). *regularized-regression*. Retrieved from bradleyboehmke: https://bradleyboehmke.github.io/HOML/regularized-regression.html

Hipson, W. (2019, 01). *a-new-way-to-handle-multivariate-outliers*. Retrieved from r-bloggers: https://www.r-bloggers.com/2019/01/a-new-way-to-handle-multivariate-outliers/

Nantasenamat, C. (2021, 09). Retrieved from Youtube: https://youtu.be/9uFQECk30kA

Appendix: R Code and Technical Details

```
library(dplyr)
library(psych)
library(glmnet)
```

```
library(caret)
library(vip)
library(randomForest)
```

```
# omit outliers
excluded = names_outliers
data_clean = trainset[-excluded, ]
```

```
# Y1 = Price_euros with outlier data points
# Center Y, X will be standardized in the modelling function
X1 = model.matrix(Price_euros ~.-1, data = trainset)
testX1 = model.matrix(Price_euros ~.-1, data = testset)

# Y
Y1 = trainset$Price_euros
testY1= testset$Price_euros

# Ridge Regression
# Fitting
ridge1 <- glmnet(x = X1, y = Y1, alpha = 0)
#  Doing cross validation to select the best lambda
set.seed(1234)
ridgeCV1 <- cv.glmnet(x = X1, y = Y1,alpha = 0)
bestlam1 = ridgeCV1$lambda.min
bestlam1
# coefficients
predict(ridge1,type = "coefficients",s = bestlam1)

# compute RMSE values
pred_train1=predict(ridge1, s=bestlam1,newx=X1)
RMSE( pred_train1,  Y1)
pred_test1=predict(ridge1, s=bestlam1,newx=testX1)
RMSE( pred_test1,  testY1)
```

```
# Lasso
# fitting
lasso1 = glmnet(x = X1, y = Y1, alpha = 1)
#  Doing cross validation to select the best lambda
# set.seed(1234)
lassoCV1 =  cv.glmnet(x = X1, y = Y1, alpha = 1)
lam.best1=lassoCV1$lambda.min
lam.best1
# coefficients
predict(lasso1,type = "coefficients",s = lam.best1)
# compute RMSE values
pred_train1=predict(lasso1, s=lam.best1,newx=X1)
RMSE( pred_train1, Y1)
pred_test1=predict(ridge1, s=bestlam1,newx=testX1)
RMSE(pred_test1, testY1)
```

```
#Enet
cv_glmnet1 <- train(
  x = X1,
  y = Y1,
  method = "glmnet",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)
# model with lowest RMSE
cv_glmnet1$bestTune
# results for model with lowest RMSE
cv_glmnet1$results %>%
  filter(alpha == cv_glmnet1$bestTune$alpha,
         lambda == cv_glmnet1$bestTune$lambda)
enet_model1=glmnet(X1,Y1,alpha = cv_glmnet1$bestTune$alpha,
                   lambda=cv_glmnet1$bestTune$lambda,standardize = T)
# coefficients
predict(enet_model1,type = "coefficients")
# compute RMSE values
pred_train1=predict(enet_model1,newx=X1)
RMSE(pred_train1,Y1)
pred_test1=predict(enet_model1,newx=testX1)
RMSE(pred_test1, testY1)
```

```
#random forest
set.seed(1234)
modelRF2=train( Price_euros ~ ., data = trainset, method= 'rf',
                ntree = 500,  trControl=trainControl(method = 'cv',number=10),
                importance = TRUE )
modelRF2
#evaluate the model performance on the train set
sqrt(mean((trainset$Price_euros - predict(modelRF2, trainset))^2))
#evaluate the model performance on the test set
sqrt(mean((testset$Price_euros -  predict(modelRF2, testset))^2))
```

```
# Y2 = Price_euros without outlier data points
# Center Y, X will be standardized in the modelling function
X2 = model.matrix(Price_euros ~.-1, data = data_clean)
testX2 = model.matrix(Price_euros ~.-1, data = testset)

# transform y with log transformation
Y2 = data_clean$Price_euros
testY2= testset$Price_euros

# Ridge Regression
# Fitting
ridge2 <- glmnet(x = X2, y = Y2, alpha = 0)
#  Doing cross validation to select the best lambda
set.seed(1234)
ridgeCV2 <- cv.glmnet(x = X2, y = Y2,alpha = 0)
bestlam2 = ridgeCV2$lambda.min
bestlam2
# coefficients
predict(ridge2,type = "coefficients",s = bestlam2)

# compute RMSE values
pred_train2=predict(ridge2, s=bestlam2,newx=X2)
RMSE( pred_train2,  Y2)
pred_test2=predict(ridge2, s=bestlam2,newx=testX2)
RMSE( pred_test2,  testY2)
```

```
# Lasso
# fitting
lasso2 = glmnet(x = X2, y = Y2, alpha = 1)
#  Doing cross validation to select the best lambda
# set.seed(1234)
lassoCV2 =  cv.glmnet(x = X2, y = Y2, alpha = 1)
lam.best2=lassoCV2$lambda.min
lam.best2
# coefficients
predict(lasso2,type = "coefficients",s = lam.best2)
# compute RMSE values
pred_train2=predict(lasso2, s=lam.best2,newx=X2)
RMSE( pred_train2, Y2)
pred_test2=predict(ridge2, s=bestlam2,newx=testX2)
RMSE(pred_test2, testY2)
```

```
#Enet
cv_glmnet2 <- train(
  x = X2,
  y = Y2,
  method = "glmnet",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)

# model with lowest RMSE
cv_glmnet2$bestTune
# results for model with lowest RMSE
cv_glmnet2$results %>%
  filter(alpha == cv_glmnet2$bestTune$alpha,
         lambda == cv_glmnet2$bestTune$lambda)
enet_model2=glmnet(X2,Y2,alpha = cv_glmnet2$bestTune$alpha,
                   lambda=cv_glmnet2$bestTune$lambda,standardize = T)
# coefficients
predict(enet_model2,type = "coefficients")
# compute RMSE values
pred_train2=predict(enet_model2,newx=X2)
RMSE(pred_train2,Y2)
pred_test2=predict(enet_model2,newx=testX2)
RMSE(pred_test2, testY2)
```

```
#random forest
set.seed(1234)
modelRF3=train( Price_euros ~ ., data = data_clean, method= 'rf',
                ntree = 500,  trControl=trainControl(method = 'cv',number=10),
                importance = TRUE )
modelRF3
#evaluate the model performance on the train set
sqrt(mean((data_clean$Price_euros - predict(modelRF3, data_clean))^2))
#evaluate the model performance on the test set
sqrt(mean((testset$Price_euros -  predict(modelRF3, testset))^2))
```

```
# Y3 = log(Price_euros) with outlier data points
# Center Y, X will be standardized in the modelling function
X3 = model.matrix(log(Price_euros) ~.-1, data = trainset)
testX3 = model.matrix(log(Price_euros) ~.-1 + ., data = testset)

# transform y with log transformation
Y3 =  log(trainset$Price_euros)
testY3=  log(testset$Price_euros)

# Ridge Regression
# Fitting
ridge3 <- glmnet(x = X3, y = Y3, alpha = 0)
#  Doing cross validation to select the best lambda
ridgeCV3 <- cv.glmnet(x = X3, y = Y3,alpha = 0)
bestlam3 = ridgeCV3$lambda.min
bestlam3
# coefficients
predict(ridge3,type = "coefficients",s = bestlam3)

# compute RMSE values
pred_train3=predict(ridge3, s=bestlam3,newx=X3)
RMSE(exp(pred_train3), exp(Y3))
pred_test3=predict(ridge3, s=bestlam3,newx=testX3)
RMSE(exp(pred_test3), exp(testY3))
```

```
# Lasso
# fitting
lasso3 = glmnet(x = X3, y = Y3, alpha = 1)
#  Doing cross validation to select the best lambda
lassoCV3 =  cv.glmnet(x = X3, y = Y3, alpha = 1)
lam.best3=lassoCV3$lambda.min
lam.best3
# coefficients
predict(lasso3,type = "coefficients",s = lam.best3)
# compute RMSE values
pred_train3=predict(lasso3, s=lam.best3,newx=X3)
RMSE(exp(pred_train3), exp(Y3))
pred_test3=predict(ridge3, s=bestlam3,newx=testX3)
RMSE(exp(pred_test3), exp(testY3))
```

```
#Enet
cv_glmnet3 <- train(
  x = X3,
  y = Y3,
  method = "glmnet",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)

# model with lowest RMSE
cv_glmnet3$bestTune
# results for model with lowest RMSE
cv_glmnet3$results %>%
  filter(alpha == cv_glmnet3$bestTune$alpha,
         lambda == cv_glmnet3$bestTune$lambda)
enet_model3=glmnet(X3,Y3,alpha = cv_glmnet3$bestTune$alpha,
                   lambda=cv_glmnet3$bestTune$lambda,standardize = T)
# coefficients
predict(enet_model3,type = "coefficients")
# compute RMSE values
pred_train3=predict(enet_model3,newx=X3)
RMSE(exp(pred_train3), exp(Y3))
pred_test3=predict(enet_model3,newx=testX3)
RMSE(exp(pred_test3), exp(testY3))
```

```
#random forest
set.seed(1234)
modelRF4=train( log(Price_euros) ~ ., data = trainset, method= 'rf',
                ntree = 500,  trControl=trainControl(method = 'cv',number=10),
                importance = TRUE )
modelRF4
#evaluate the model performance on the train set
sqrt(mean((trainset$Price_euros - exp(predict(modelRF4, trainset)))^2))
#evaluate the model performance on the test set
sqrt(mean((testset$Price_euros - exp(predict(modelRF4, testset)))^2))
```

```r
# Y4 = log(Price_euros) without outlier data points
# Center Y, X will be standardized in the modelling function
X = model.matrix(Price_euros ~.-1, data = data_clean)
testX = model.matrix(Price_euros ~.-1, data = testset)

# transform y with log transformation
Y = log(data_clean$Price_euros)
testY= log(testset$Price_euros)

# Ridge Regression
# Fitting
ridge <- glmnet(x = X, y = Y, alpha = 0)
#  Doing cross validation to select the best lambda
set.seed(1234)
ridgeCV <- cv.glmnet(x = X, y = Y,alpha = 0)
bestlam = ridgeCV$lambda.1se
bestlam
# coefficients
predict(ridge,type = "coefficients",s = bestlam)
# compute RMSE values
pred_train=predict(ridge, s=bestlam,newx=X)
RMSE(exp(pred_train), exp(Y))
pred_test=predict(ridge, s=bestlam,newx=testX)
RMSE(exp(pred_test), exp(testY))
```

```r
# Lasso
# fitting
lasso = glmnet(x = X, y = Y, alpha = 1)
#  Doing cross validation to select the best lambda
# set.seed(1234)
lassoCV =  cv.glmnet(x = X, y = Y, alpha = 1)
lam.best=lassoCV$lambda.1se
lam.best
# coefficients
predict(lasso,type = "coefficients",s = lam.best)
# compute RMSE values
pred_train=predict(lasso, s=lam.best,newx=X)
RMSE(exp(pred_train), exp(Y))
pred_test=predict(lasso, s=lam.best,newx=testX)
RMSE(exp(pred_test), exp(testY))
```

```r
#Enet
cv_glmnet <- train(
  x = X,
  y = Y,
  method = "glmnet",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)

# model with lowest RMSE
cv_glmnet$bestTune
# results for model with lowest RMSE
cv_glmnet$results %>%
  filter(alpha == cv_glmnet$bestTune$alpha,
         lambda == cv_glmnet$bestTune$lambda)
enet_model=glmnet(X,Y,alpha = cv_glmnet$bestTune$alpha,
                  lambda=cv_glmnet$bestTune$lambda,standardize = T)
# coefficients
predict(enet_model,type = "coefficients")

# compute RMSE values
pred_train=predict(enet_model,newx=X)
RMSE(exp(pred_train), exp(Y))
pred_test=predict(enet_model,newx=testX)
RMSE(exp(pred_test), exp(testY))
```

```r
#random forest
set.seed(1234)
modelRF1=train( log(Price_euros) ~ ., data = data_clean, method= 'rf',
                ntree = 500,  trControl=trainControl(method = 'cv',number=10),
                importance = TRUE )
modelRF1
#evaluate the model performance on the train set
sqrt(mean((data_clean$Price_euros - exp(predict(modelRF1, data_clean)))^2))
#evaluate the model performance on the test set
sqrt(mean((testset$Price_euros - exp(predict(modelRF1, testset)))^2))


# Y1 = Price_euros with outlier data points
# Center Y, X will be standardized in the modelling function
X1 = model.matrix(Price_euros ~.-1, data = trainset)
testX1 = model.matrix(Price_euros ~.-1, data = testset)
```

```r
selectedModel=modelRF2
#MAPE
mean(abs((testset$Price_euros- predict(selectedModel,testset))/testset$Price_euros))
# Important Variables
vip(selectedModel,num_features = 14,aesthetics = list(color = "blue"))
# Save model to RDS file
saveRDS(selectedModel, "model.rds")
```

Data Product

```
                selectInput('CpuBrand',label='Cpu Brand:',
                            choices = list('AMD Processor'='AMD Processor',
                                           'Intel Core i3'='Intel Core i3',
                                           'Intel Core i5'='Intel Core i5',
                                           'Intel Core i7'='Intel Core i7',
                                           'Other Intel Processor'='Other Intel Processor'),
                            selected = 'Intel Core i5'),
                sliderInput("ClockSpeed", label = "Clock Speed(GHz):",
                            min = 0.9, max = 3.6,
                            value = 2.0, step=0.01),
                sliderInput("SSD", label = "SSD(GB):",
                            min = 0, max = 1024,
                            value = 256,step = 4),
                sliderInput("HDD", label = "HDD(GB):",
                            min = 0, max = 2000,step = 4,
                            value = 500),
                sliderInput("FlashStorage", label = "Flash Storage(GB):",
                            min = 0, max = 512,step = 4,
                            value = 512),
                sliderInput("Hybrid", label = "Hybrid(GB):",
                            min = 0, max = 1000,step = 4,
                            value = 1000),
                selectInput('GpuBrand',label='GPU Brand:',
                            choices = list('AMD'='AMD',
                                           'ARM'='ARM',
                                           'Intel'='Intel',
                                           'Nividia'='Nividia'
                            ),
                            selected = 'Intel Core i5'),
```

```
# import libraries
library(shiny)
library(shinythemes)
library(shinyWidgets)
library(data.table)
# Read in the RF model
model <-readRDS("model.rds")
# User interface
ui <- tagList(fluidPage(
                theme = shinytheme("superhero"),
                setBackgroundImage(
                  src = "https://images.unsplash.com/photo-1510519138101-570d1dca3d66?ixlib=rb-1.2.

                ),

                # Page header
                headerPanel('Laptop Price Prediction'),

                # Input values
                sidebarPanel(
                  HTML("<h3>Input parameters</h3>"),
                  style = "overflow-y:scroll; max-height: 600px; position:relative;",
                  selectInput("Company", label = "Company:",
                              choices = list("Acer" = "Acer", "Apple" = "Apple", "Asus" = "Asus",
                                             'Chuwi'='Chuwi','Dell'='Dell','Fujitsu'='Fujitsu',
                                             'Google'='Google','HP'='HP','Huawei'='Huawei',
                                             'Lenovo'='Lenovo','LG'='LG','Mediacom'='Mediacom',
                                             'Microsoft'='Microsoft','MSI'='MSI','Razer'='Razer',
                                             'Samsung'='Samsung','Toshiba'='Toshiba','Vero'='Vero',
                                             'Xiaomi'='Xiaomi'),
                              selected = "Asus"),
```

```
                selectInput('TypeName',label='Type Name:',
                            choices = list('2 in 1 Convertible'='2 in 1 Convertible',
                                           'Gaming'='Gaming','Netbook'='Netbook','Notebook'='Notebook',
                                           'Ultrabook'='Ultrabook','Workstation'='Workstation'),
                            selected = 'Notebook'),
                sliderInput("Ram", label="Ram(GB):",
                            min=2,max=64,
                            value = 8),
                selectInput('OpSys',label = 'Operating System:',
                            choices = list('Linux'='Linux','Mac'='Mac','No OS'='No OS',
                                           'Others'='Others','Windows'='Windows'),
                            selected = 'Windows'),
                sliderInput("Weight", label="Weight(kg):",
                            min = 0.69, max = 4.70,
                            value = 3.0, step=0.01),
                selectInput('IPS',label='IPS:',
                            choices = list('Yes'='Yes','No'='No'),
                            selected = 'Yes'),
                selectInput('TouchScreen',label='Touch Screen:',
                            choices = list('Yes'='Yes','No'='No'),
                            selected = 'Yes'),
                sliderInput("size", label="Screen Size(Inches):",
                            min = 10, max = 20,
                            value = 14, step=0.1),
                selectInput("screen", label = "Screen Resolution:",
                            choices = list('1366x768'='1366x768','1440x900'='1440x900','1600x900'='1600x900',
                                           '1920x1080'='1920x1080','1920x1200'='1920x1200','2160x1440'='2160x1440',
                                           '2256x1504'='2256x1504','2304x1440'='2304x1440','2400x1600'='2400x1600',
                                           '2560x1440'='2560x1440','2560x1600'='2560x1600','2736x1824'='2736x1824',
                                           '2880x1800'='2880x1800','3200x1800'='3200x1800','3840x2160'),
                            selected = '1920x1080'),
```

```
    actionButton("submitbutton", "Submit", class = "btn btn-primary"),
    HTML('<br><br><br><br><br>')
  ),

  mainPanel(
    tags$style(
      "body {overflow-y: hidden;}"
    ),
    tags$label(h3('Status/Output')), # Status/Output Text Box
    verbatimTextOutput('contents'),
    tableOutput('Result') ,# Prediction results table
  )
  ),
tags$footer("Designed by: Group No:4/ST3082/2021-22", align = "center", style = "
position:absolute;
bottom:0;
width:100%;
height:30px;
color: white;
padding: 5px;
background-color: black;
z-index: 1000;"))
```

```
# Server
server <- function(input, output, session) {

  # Input Data
  datasetInput <- reactive({

    df <- data.frame(
      Name = c("Company",
               "TypeName",
               "Ram",
               "OpSys",
               "Weight",
               "IPS","TouchScreen",'size',"screen","CpuBrand",
               "ClockSpeed","SSD","HDD","FlashStorage","Hybrid","GpuBrand"),
      Value = as.character(c(input$Company,
                             input$TypeName,
                             input$Ram,
                             input$OpSys,
                             input$Weight,
                             input$IPS,
                             input$TouchScreen,input$size,input$screen,input$CpuBrand,input$ClockSpeed,
                             input$SSD,input$HDD,input$FlashStorage,
                             input$Hybrid,input$GpuBrand)),
      stringsAsFactors = F)

    Price_euros <- 'Price_euros'
    df <- rbind(df, Price_euros)
    input <- transpose(df)
    write.table(input,"input.csv", sep=",", quote = FALSE, row.names = FALSE, col.names = FALSE)
```

```
    test1 <- read.csv(paste("input", ".csv", sep=""), header = TRUE)
    test1$Xres=as.integer(substr(test1$screen,1,4))
    test1$Yres=as.integer(substr(test1$screen,6,9))
    test1$PPI=sqrt((test1$Xres^2)+(test1$Yres^2))/test1$size
    test=subset(test1, select=-c(screen,size,Xres,Yres))



    test$Company <- factor(test$Company, levels = c('Acer','Apple','Asus','Chuwi','Dell','Fujitsu','Google','HP',
                                                    'Huawei','Lenovo','LG','Mediacom','Microsoft','MSI','Razer',
                                                    'Samsung',
                                                    'Toshiba','Vero','Xiaomi'))
    test$TypeName <- factor(test$TypeName, levels = c('2 in 1 Convertible','Gaming','Netbook','Notebook','Ultrabook' ,
                                                      'Workstation'))
    test$OpSys <- factor(test$OpSys, levels = c('Linux','Mac','No OS','Others','Windows'))
    test$CpuBrand <- factor(test$CpuBrand, levels = c('AMD Processor','Intel Core i3','Intel Core i5','Intel Core i7',
                                                      'Other Intel Processor'))
    test$GpuBrand <- factor(test$GpuBrand, levels = c('AMD','Intel','ARM','Nividia'
    ))


    Output <- data.frame(Prediction_in_Euros=predict(model,test))
    print(Output)

  })
```

```
  # Status/Output Text Box
  output$contents <- renderPrint({
    if (input$submitbutton>0) {
      isolate("Calculation complete.")
    } else {
      return("Server is ready for calculation.")
    }
  })

  # Prediction results
  output$Result <- renderTable({
    if (input$submitbutton>0) {
      isolate(datasetInput())
    }
  })

}

# Create the shiny app
shinyApp(ui = ui, server = server)
```