

**Final Project Complete Research Paperwork**

**Group name**: Group Beta

**Members:** [Likhitha Varakala](https://northeastern.instructure.com/courses/131197/users/222521)

Manmitha Pantangi

Vrushali Ransubhe

Aybuke Korkan Yilmaz

**Class Number**: ALY6015

**Class Name**: Intermediate Analytics

**Class CRN**: 20419

**Email**: [varakala.l@northeastern.edu](mailto:varakala.l@northeastern.edu)

[pantangi.m@northeastern.edu](mailto:pantangi.m@northeastern.edu)

[ransubhe.v@northeastern.edu](mailto:ransubhe.v@northeastern.edu)

korkanyilmaz.a@northeastern.edu

**Introduction**

The dataset we chose for is “**Laptop Price Prediction**” and it is obtained from Kaggle. The data consists of 1303 rows and 12 variables.

The columns present in the dataset are,

|  |  |
| --- | --- |
| Column Name | Description |
| Laptop\_Id | Unique Id of the laptop |
| Company | The manufacturer of the laptop |
| TypeName | Type (Notebook, Ultrabook, Gaming, etc) |
| Inches | Screen Size |
| ScreenResolution | Screen Resolution of the laptop |
| Cpu | CPU brand, Cpu type and CPU Speed |
| Ram | Laptop RAM |
| Memory | Hard Disk / SSD Memory |
| Gpu | Graphics Processing Units (GPU) |
| OpSys | Operating System |
| Weight | Weight of the laptop in kg |
| Price | Price of the laptop in INR |

The questions we would like to examine are,

* What aspects affect the final cost of a laptop?
* Which model provides the most accurate laptop price prediction?

Methods performed,

* Multiple Linear Regression
* Ridge Regression
* Lasso Regression

**Multiple Linear Regression**

To determine the relationship between two or more independent variables and a dependent variable, utilize multiple linear regression. In this instance, we can examine how the columns are affecting the "Price" column. We can determine which factors contribute to a laptop's overall cost using this model. However, this model has limitations of its own, which is where regularized models come into play.

**Regularization** is a key technique that is employed to prevent overfitting of data, particularly when there is significant variability between the training and testing datasets. The approach involves introducing a "penalty" term into the best fit obtained from the training dataset, which helps to reduce variance with the testing dataset. Additionally, regularization restricts the impact of predictor variables on the output variable by compressing their coefficients.

**Ridge Regression**

Ridge regression adds a regularization factor to the common linear regression equation to handle multicollinearity. As a penalty for high regression coefficient values, this regularization term works to reduce the coefficients towards zero and avoid overfitting.

* It shrinks the parameters.
* It uses coefficient shrinking to lessen the complexity of the model.

**LASSO Regression**

Lasso regression is a method of regularization that is preferred over other regression techniques for more precise predictions. The approach involves shrinkage, whereby data values are pulled towards a central point, typically the mean. The lasso procedure is designed to favor simpler models that rely on fewer parameters.

The second question that we seek to answer is which of these three models will provide us with the accurate prediction.

**Exploratory Data Analysis**

In order to perform EDA effectively, we will have to clean the data.

**Data Cleaning**

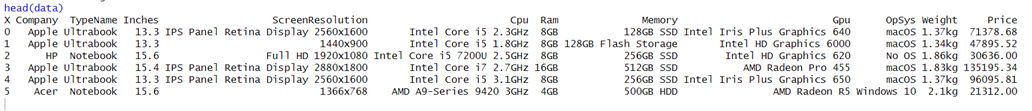
Installing Libraries and importing the dataset:

Before starting the code we installed necessary libraries and imported the data.

A picture containing text, screenshot, computer, indoor

Description automatically generated

The head() of the dataset,



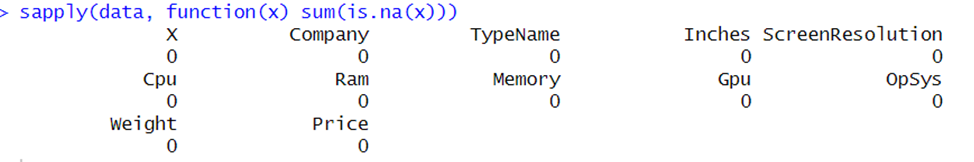
Summary before cleaning,

A screenshot of a computer

Description automatically generated

Using the summary() function, we can see that the variables Company, TypeName, ScreenResolution, Cpu, Ram, Memory, Gpu, OpSys and Weight are characters, and the others are numeric.

Checking for missing data in the dataset,



The output makes clear that there are no missing values or NAs in the data.

In-order to work with the data in full extent, we will have to clean up the data a bit by splitting and removing certain characters from the columns,

First, Splitting Gpu using the str\_split\_fixed() function into 2 different variables to get ‘Gpu Vendor’ and ‘Gpu Type’.

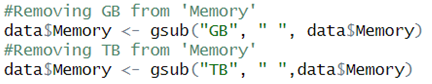


Splitting Memory into Memory and Storage Type,

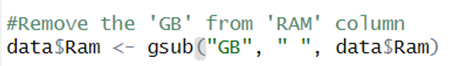


R uses the gsub() function for replacement operations. The function uses the input to replace the values that have been supplied. So, using the gsub() function

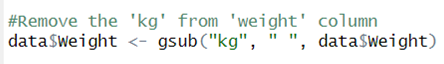
Removing ‘GB’ and ‘TB’ from Memory,



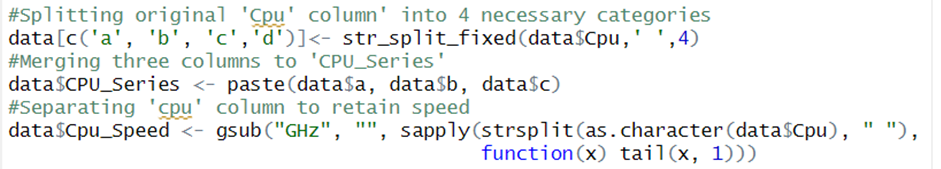
Removing ‘GB’ from RAM,



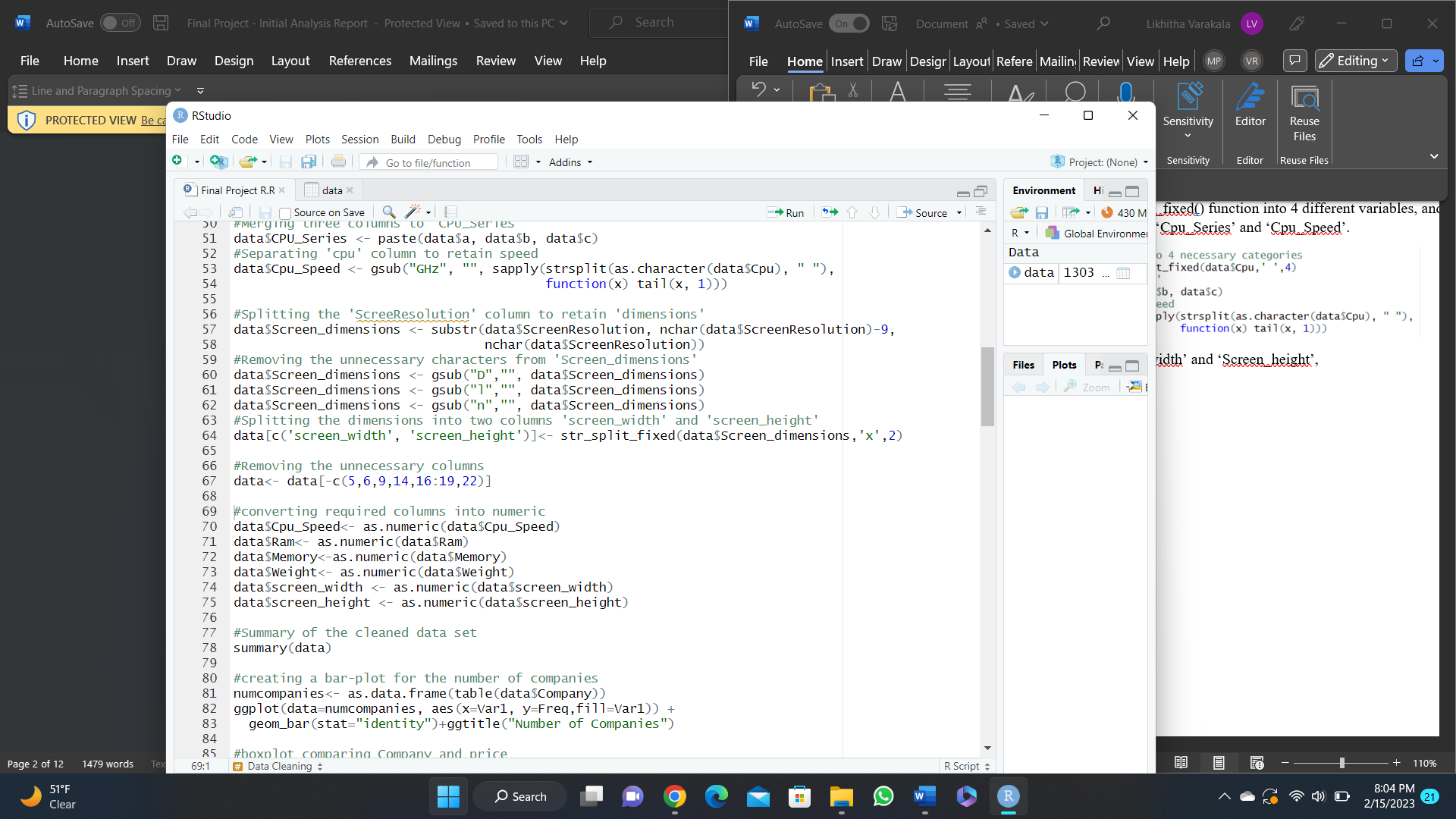
Removing ‘kg’ from Weight,



Splitting original ‘Cpu’ column using the str\_split\_fixed() function into 4 different variables, and then combining the first three variables together to get ‘Cpu\_Series’ and ‘Cpu\_Speed’.



Splitting ‘Screen\_dimensions’ to retain the dimensions ‘Screen\_width’ and ‘Screen\_height,’



Removing all the unnecessary columns,

A screenshot of a computer

Description automatically generated

Now, converting all the required variables to numeric,

A screenshot of a computer

Description automatically generated

This completes the process of cleaning the data, and the final summary of the data is as follows:

A screenshot of a computer

Description automatically generated with medium confidence

The final cleaned dataset consists of 1303 observations and 15 variables. The additional variables added to the dataset are ‘Gpu\_vendor’, ‘Storage\_type’, ‘CPU\_Series’, ‘Cpu\_Speed’, ‘screen\_width’ and ‘screen\_height’.

**Visualizations**

Bar plot for the number of companies in the dataset

Following code is used to generate the required bar graph,

A screenshot of a computer

Description automatically generated with medium confidence

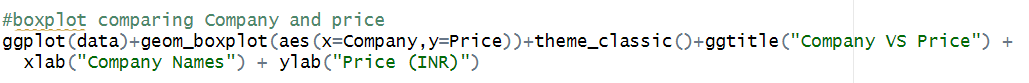
Chart, histogram

Description automatically generated

The dataset is represented graphically through a plot that shows the count of companies present, with the company names on the X-axis and the count of companies on the Y-axis. The graph provides information on the highest and lowest selling companies present in the dataset. From the graph, it can be concluded that Dell and Lenovo are the most popular brands among the other 19 companies. Conversely, Huawei is the least popular company in terms of sales.

**Boxplot to represent the mean prices with respect to the companies.**

Following code is used to generate the required boxplot using ggplot(),



Chart, box and whisker chart

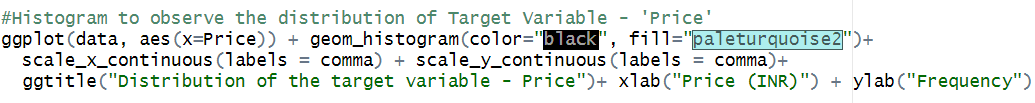
Description automatically generated

A boxplot was created to display the mean prices for each company. The plot allows for a visual comparison of the mean prices between the different companies.

Based on the boxplot, it can be observed that Razer laptops have the highest mean prices compared to other brands. Therefore, it can be concluded that laptops from the Razer brand are the most expensive among the others.

**Histogram to show the distribution of the target variable – ‘Price’**

To gain a clearer understanding of the distribution of prices, a histogram is created using ggplot(). The histogram provides a visual representation of how frequently different price ranges occur in the dataset.



Chart, histogram

Description automatically generated

The "price" variable's distribution is skewed, making it clear that cheaper laptops like those from Dell and Lenovo are sold and bought more frequently than name-brand models.

**Multiple Linear Regression**

Multiple linear regression is used to build a model that can help us understand how a set of independent variables influence a dependent variable. The model assumes a linear relationship between the independent variables and the dependent variable, meaning that a change in one independent variable is associated with a proportional change in the dependent variable.

Steps to perform Multiple Linear Regression

1. Plot a scatter plot between x and y, show the correlations.
2. Split train and test data.
3. Create a linear regression model.
4. Make predictions on the test dataset.
5. Finding Accuracy for test data.

Computing a correlation matrix using cor(),

Table

Description automatically generated

**Correlation shown by corrplot() function**

Chart, bubble chart

Description automatically generated

We will examine the Price variable—our dependent variable—and its interactions with other variables using the correlation chart as our guide. We can see that, as expected, screen height and screen width have a strong correlation. Price and RAM have a strong correlation. Inches and weight also have a strong relationship.

Before performing the linear regression, the dataset is to be split into train and test sets. The following code is used to split the ‘Laptops’ data into train and test sets. Before creating data partition, the *set.seed* function is used to retain the generated random values. The train data constitutes 80% of the rows and the test data constitutes the remaining 20% of the rows.

Graphical user interface, text

Description automatically generated

As the original data contains 1303 observations, the training set consists of 80% of the data i.e., 1044 rows and the testing set contains of 20% of the data i.e., 259 rows with 15 variables.

To utilize the train and test datasets for regularization, these sets are converted into matrices using *model.matrix* function. To remove the extra columns generated while conversion of train and test sets into matrices, *setdiff* function is used. The dependent variable ‘Price’ from train and test data is assigned to ‘train\_y’ and ‘test\_y’ variables respectively.

First, we are going to perform multiple linear regression with all the variables to determine which of the predictors are statistically significant,



Table

Description automatically generatedText, table

Description automatically generated with medium confidence  
By examining the model summary, we can see that the variables Cpu\_Speed, Ram, Memory, and Weight hold significant statistical importance, and they can be utilized for additional analysis. This conclusion is done using the p-values of those variables, as they are less than the significance level, alpha = 0.05. It's crucial to evaluate the multiple R-squared value when implementing a linear regression model, as it provides insight into the model's effectiveness. The multiple R-squared value indicates the degree to which the model fits the data. Here the multiple R-squared value is 0.8124 indicating that our model is 81.24% accurate while including all the variables.

Now, let’s include only statistically significant predictors into the model,

Text

Description automatically generated

By examining the model summary, we can see that the variables Cpu\_Speed, Ram, Memory, and Weight are statistically significant and impact the price of the laptop. This conclusion is done using the p-values of those variables, as they are less than the significance level, alpha = 0.05.

The regression equation for the model is,

*Price = 13702.35(Cpu\_Speed) +5050.68(Ram) + 30.84(Memory) + (-5229.18)(Weight) +*

*(-10530.05)*

For example, one unit increase in weight would decrease the price of the laptop by Rs. 5229.18. The multiple R-squared value indicates the degree to which the model fits the data. Here the multiple R-squared value is 0.6323 indicating that our model is 63.23% accurate while including all the variables.

**RMSE value**

Text

Description automatically generated

First, we predict the price of the laptop using the test data and we determine the RMSE value.

RMSE stands for Root Mean Squared Error, which is a commonly used metric to evaluate the accuracy of a statistical or machine learning model in predicting continuous numerical values. It measures the difference between the predicted values and the actual values, taking the square of the differences, averaging them, and then taking the square root of the result.

The RMSE value provides information about the average distance between the predicted values and the actual values, with a lower RMSE indicating better accuracy. In our model, the RMSE value for the test prediction is 23993.28, which is pretty high.

We will further perform Ridge and LASSO regression to predict the best model out of the three.

**Regularization**

Methods for calibrating machine learning models to lower the adjusted loss function and prevent overfitting or underfitting are referred to as "regularization." Constraining, regularizing, or reducing the coefficient estimates in the direction of zero is the process of regularization. Regularization comes in two flavors, L1 and L2.

The L1 regularization, commonly known as a LASSO regression, adds the "absolute value of magnitude" of the coefficient as a penalty term to the loss function. The "squared magnitude" of the coefficient, sometimes referred to as ridge regression or L2 regularization, is added to the loss function as a penalty term. The tuning parameter lambda can be used to modify the severity of the penalty term.

**Ridge Regression**

**Cross-validation:**

The *cv.glmnet* function from *glmnet* package is used to perform cross-validation for different lambda values by default 10 folds.

The *set.seed* function is used to retain the generated random values. For ridge regression, alpha is equal to zero. The cross-validation estimates the possible minimum and maximum lambda values. The obtained lambda values for lasso regression are,

Text

Description automatically generated

Text, letter

Description automatically generated

* Lambda minimum (lambda.min): 2871
* Maximum lambda within 1 standard error of minimum lambda (lambda.1se): 16818

These minimum and maximum lambda values are the two parameters which can be used to fit the actual ridge regression model. At these lambda values, it is expected to obtain smallest mean difference between actual and predicted variables in the model.

The log values of minimum and maximum lambda can be obtained by following code:

Graphical user interface, text, application

Description automatically generated

* Minimum log value of lambda – 7.96.
* Maximum log value of lambda – 9.73.

**Plot the results from cross-validation:**

The plot for k-fold cross-validation curve for lambda values for ridge is as shown below,

Chart

Description automatically generated

The x-axis shows the log lambda values, and the y-axis shows the mean-squared error, and the top values are for the number of nonzero coefficient estimates. The red dotted line from the plot is the cross-validation curve along with upper and lower standard deviation curves calculated for every kth fold (I have set the kth fold is set to 10). The two vertical dotted lines represent the log lambda.min and log lambda.1se. The dotted line for lambda tells that there are 52 non-zero coefficients.

**Fit the ridge regression model:**

A ridge regression model is fit against the training set by using *glmnet* function. For ridge regression, the alpha value is to be set to 0.

Since the obtained minimum lambda value is the least possible lambda, ridge regression model can be fit using the lambda.min value.

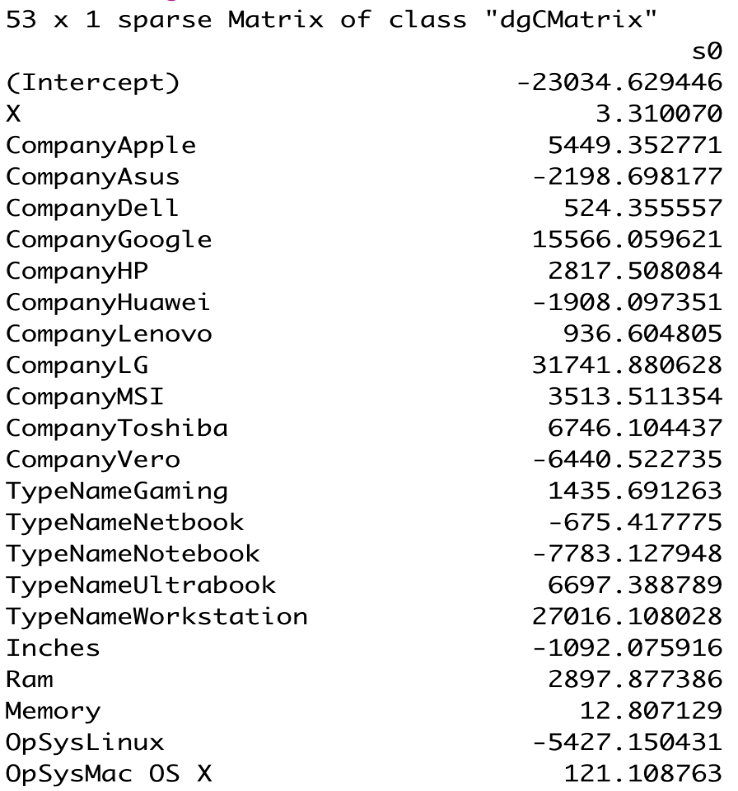
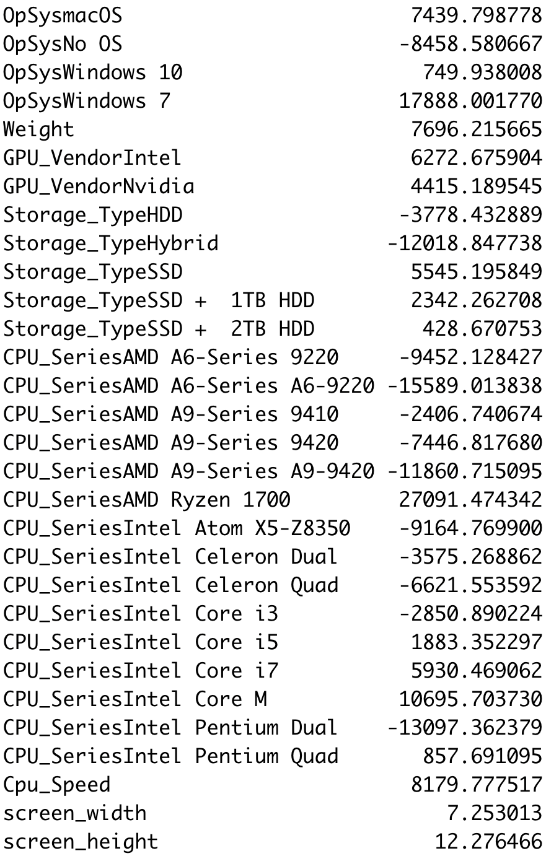
Ridge regression using lambda.min value:

Graphical user interface, text, application

Description automatically generated

Df value is 52 which is the number of nonzero coefficients, same value we have observed in the plot. The percent (of null) deviance explained (%dev) is 77.45 for the lambda.min.

The coefficients of the ridge model are as follows:

The coefficients of irrelevant variables are reduced using the Ridge regularization approach to a value that is almost zero but never equal to zero. Therefore, the ridge model keeps all 53 variables. As a result, we can see that all the variables have coefficients that are almost 0 but not exactly zero. According to ridge regression, the significance of each variable depends on their coefficients. If the coefficients are very small, the variable is not significant for the model.

Many of the coefficients for "CPU Series" in the ridge model shown above are negative, indicating that these variables will have a negative impact on the overall pricing (low) of the laptop.

**Plot for the coefficients of lasso model:**

Text

Description automatically generated

Chart

Description automatically generated

A plot is generated to visualize the coefficients of variables in the ridge model by taking log(lambda) on the X-axis and coefficients on the Y-axis. The plot describes the variation in coefficients with respect to increasing lambda value.

The plot clearly shows that there are 4 highest coefficients, with a value approximately between 20000 and 30000, when lambda value is between 12 and 14. For the ridge model with lambda.min value (lambda.min = 7.96), the highest coefficient is obtained for ‘Cpu\_speed’ and ‘Operating system’, which indicates that these variables explain greater variation in the dependent variable, ‘Price.’ This means that the price of the laptop is expected to be high based on average cpu\_speed and operating system of the laptop.

Additionally, it can be seen that the coefficients tend to reach zero as lambda value increases and are slightly higher than zero when lambda value is lower.

**RMSE and R2 values:**

The RMSE or root mean square error measures the average difference between predicted and actual variable values.

The R2 value is the coefficient of determination, which gives the percentage of the variation in dependent variable that is predictable by the independent variable.

The *predict* function is used to make predictions for the produced ridge model with minimum lambda values from train and test data sets.

Text

Description automatically generated

Text

Description automatically generated

The obtained values for train set are:

* RMSE value – 17953.03.
* R2 value – 0.774 i.e., 77%.

The obtained values for test set are:

* RMSE value – 17056.5.
* R2 value – 0.759 i.e., 75%.

We have observed the RMSE value 17056.5 and R2 value 0.7596 i.e., 75.96% for test set . The R2 value tells about the percentage of variance in the dependent variable that is predicted by the independent variable, in this case the 52 independent variables in the ridge model can account for about 76% of variations in the dependent variable "Price". The RMSE value for train set is 17953.03 which is greater than the RMSE value for test data 17056.5, is not much different and does not show overfitting of the model. Also, the *nfolds* parameter is set to 10 during cross-validation, which implies that it used 10 permutations to establish the minimum and maximum values for lambda, greatly lowering the probability that the model is overfit. Therefore, it can be stated that the model is not overfit.

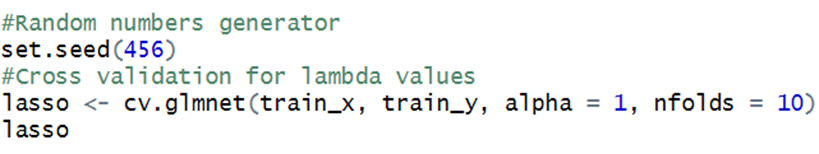
In comparison to the R2 value 63% of linear regression model we can see that the values are improved to about 76% in the Ridge regression model.

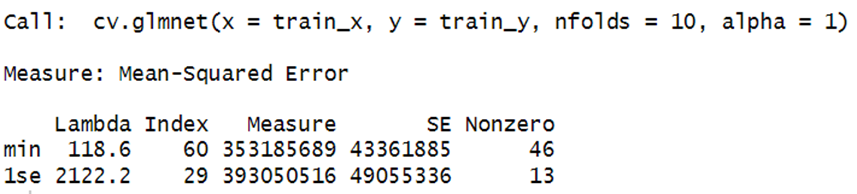
**LASSO Regularization**

**Cross-validation:**

The *cv.glmnet* function from *glmnet* package is used to perform cross-validation for different lambda values by default 10 folds.

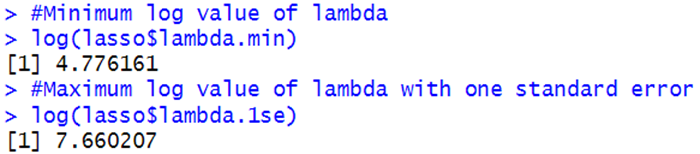
The *set.seed* function is used to retain the generated random values. For lasso regression, alpha is equal to one. The cross-validation estimates the possible minimum and maximum lambda values. The obtained lambda values for lasso regression are,





* Lambda minimum (lambda.min) – 118.6
* Maximum lambda within 1 standard error of minimum lambda (lambda.1se) – 2122.2

These minimum and maximum lambda values are the two parameters which can be used to fit the actual lasso regression model. At these lambda values, it is expected to obtain smallest mean difference between actual and predicted variables in the model.

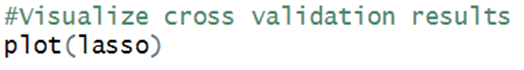


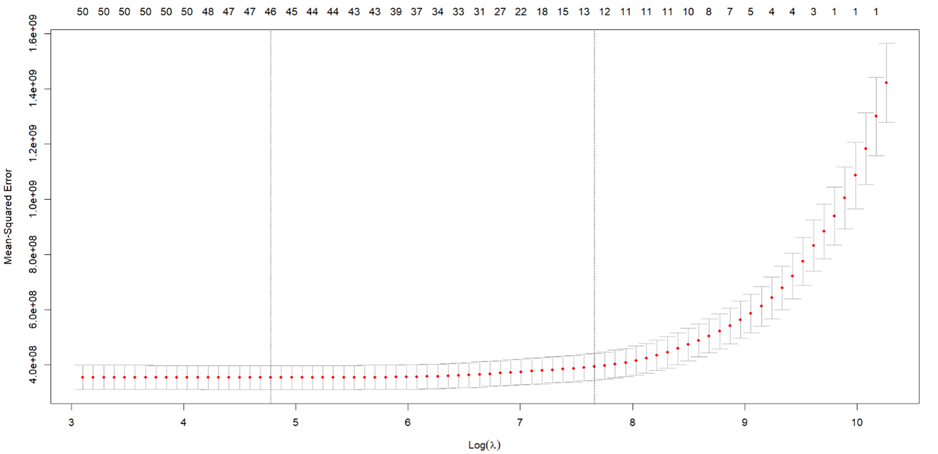
The log values of minimum and maximum lambda can be obtained by following code:

* Minimum log value of lambda – 4.776.
* Maximum log value of lambda – 7.660.

**Plot the results from cross-validation:**

The following code is used to generate plot for the obtained results from cross-validation.





The cross-validation curve is depicted in the above plot with mean-square error on the Y-axis and mean-square error as a function of lambda (log(lambda)) on the X-axis.

The minimum and maximum lambda values are represented in the plot by the two vertical dotted lines. The minimal lambda is shown on the first line, and the greatest lambda within one standard error of the minimum lambda is shown on the second line. The estimated errors of actual and predicted values are represented by the red dots on the curve.

When the lambda value is at its lowest, the number 46 at the top of the plot denotes the existence of 46 variables in the model; meanwhile, the number 13 denotes the presence of 13 variables in the model when the lambda value is maximum within one standard error of the minimum lambda.

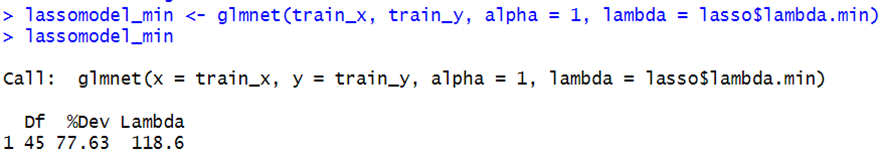
When log(lambda) is less than or equal to 8, the mean square error is similarly lower and flat; in this case, the coefficients are anticipated to be large, indicating that the model with minimum lambda value is likely to be the most effective. When log(lambda) is greater than or equal to 10, there is also a high mean square error, which suggests very small coefficient values.

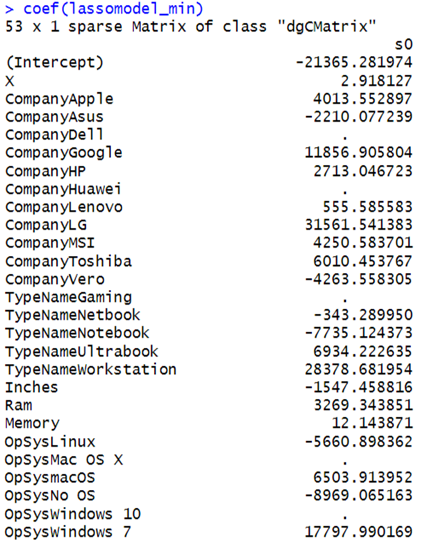
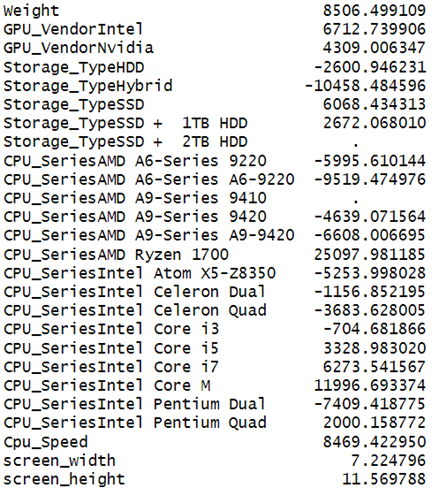
**Fit the LASSO regression model:**

A lasso regression model is fit against the training set by using *glmnet* function. For lasso regression, the alpha value is to be set to 1.

Since the obtained minimum lambda value is the least possible lambda, lasso regression model can be fit using the lambda.min value.

Lasso regression using lambda.min value:

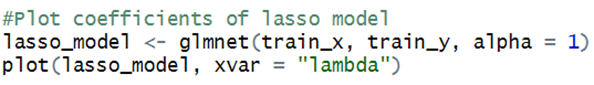


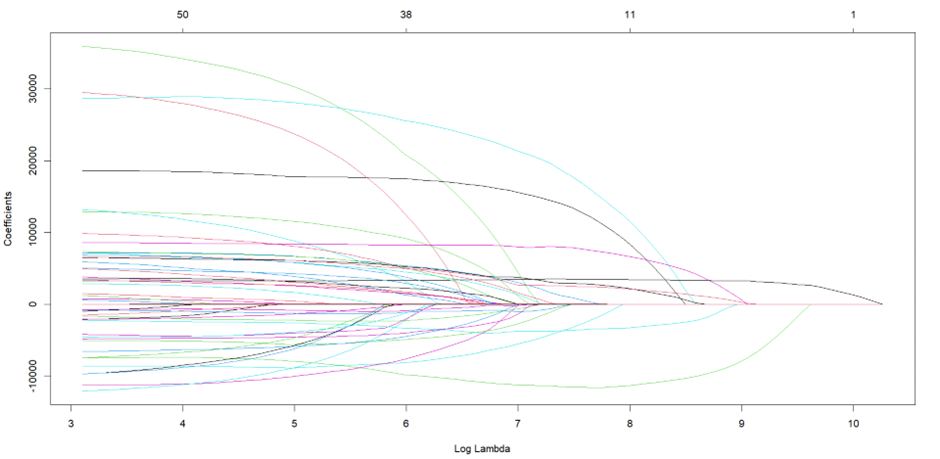
The coefficients of the above lasso model are as follows:

Using the Lasso regularization method, the coefficients of unimportant variables are minimized and equalized to zero. For the lasso model with lambda.min, the model consists of only 46 variables, the remaining unimportant 7 variables are set to zero.

The majority of the coefficients for "CPU Series" and "Operating system" in the lasso model shown above are negative, indicating that these variables will have a negative impact on the overall pricing (low) of the laptop. The lasso model has also excluded seven additional factors, such as the manufacturer and model of the laptop, which, while they do affect pricing, are outweighed by other factors like CPU series, operating system, RAM, and memory.

**Plot for the coefficients of lasso model:**





A plot is generated to visualise the coefficients of variables in the lasso model by taking log(lambda) on the X-axis and coefficients on the Y-axis. The plot describes the variation in coefficients with respect to increasing lambda value.

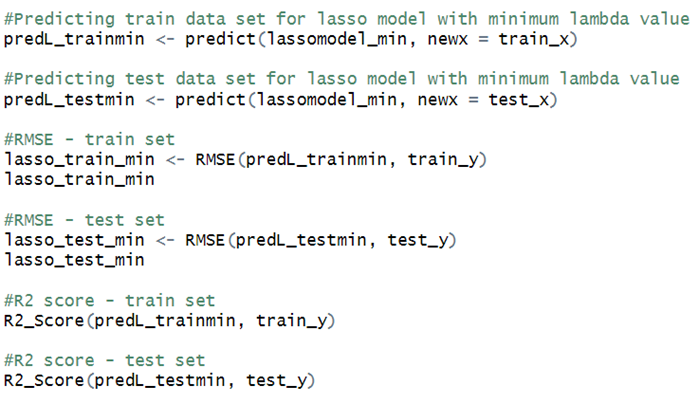
The plot clearly shows that there are 3 to 4 highest coefficients, with a value approximately between 20000 and 30000, when lambda value is between 7 and 8. For the lasso model with lambda.min value (lambda.min = 4.77), the highest coefficient is obtained for ‘Cpu\_speed’ and ‘Weight’, which indicates that these variables explain greater variation in the dependent variable, ‘Price.’ This means that the price of the laptop is expected to be high based on average cpu\_speed and weight of the laptop. Additionally, it can be seen that the coefficients tend to reach zero as lambda value increases and are slightly higher than zero when lambda value is lower.

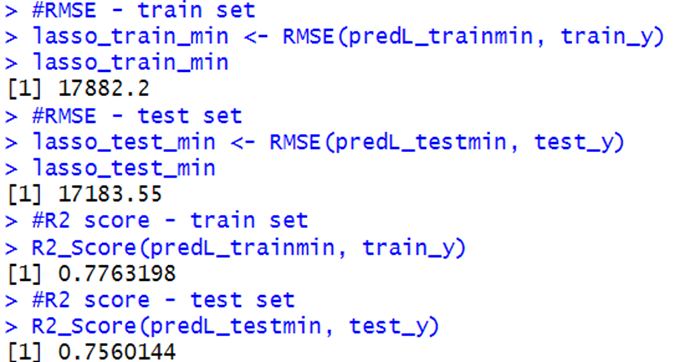
**RMSE and R2 values:**

The RMSE or root mean square error measures the average difference between predicted and actual variable values.

The R2 value is the coefficient of determination, which gives the percentage of the variation in dependent variable that is predictable by the independent variable.

The *predict* function is used to make predictions for the produced lasso models with minimum lambda values from train and test data sets.





The obtained values for train set are:

* RMSE value – 17882.2.
* R2 value – 0.776 i.e., 77%.

The obtained values for test set are:

* RMSE value – 17183.55.
* R2 value – 0.756 i.e., 75%.

The percentage of variation in the dependent variable that is predicted by the independent variables is indicated by the R2 value. According to the R2 values for the train and test sets, which are 77% and 75%, respectively, all 46 independent variables in the lasso model can account for 77% of variations in the dependent variable "Price" for the train set and 75% of variations for the test set.

It is evident that there is not a significant difference between the generated RMSE values for train and test data. The RMSE value for train set is greater than the RMSE value for test data, which implies that the model is not overfit. Additionally, the *nfolds* parameter is set to 10 during cross-validation, which implies that it used 10 permutations to establish the minimum and maximum values for lambda, greatly lowering the probability that the model is overfit. Therefore, it can be stated that the model is not overfit.

**Conclusion**

In this project we have leveraged RStudio a powerful language for data analysis using statistical methods for the laptop price prediction dataset. We have also learned about the regularization technique to get around the drawbacks of the linear regression model.

Answering our first question,

The factors that have a greater impact on the price of the laptop are ‘Ram’, ‘Cpu\_speed’, ‘Memory’, and ‘Weight’.

Below is a summary of how the models performed:

1. Linear Regression Model: Test set RMSE of 23993.28 and R-square of 63 percent.
2. Ridge Regression Model: Test set RMSE of 17056.5 and R-square of 76 percent.
3. Lasso Regression Model: Test set RMSE of 17183.55 and R-square of 75 percent.

Answering our second question,

Based on the performance of the models, it can be concluded that the Ridge regression model is the best to predict the laptop prices for this data.

**References**

*Laptop Price Dataset*. (2022, August 30). Kaggle. <https://www.kaggle.com/datasets/mohidabdulrehman/laptop-price-dataset>

*Split up a string into pieces — str\_split*. (n.d.). <https://stringr.tidyverse.org/reference/str_split.html>

Bevans, R. (2022a, November 15). *Multiple Linear Regression | A Quick Guide (Examples)*. Scribbr. <https://www.scribbr.com/statistics/multiple-linear-regression/#:~:text=Regression%20allows%20you%20to%20estimate,variables%20and%20one%20dependent%20variable>

Singh, D. (2019, November 12). *Linear, Lasso, and Ridge Regression with R | Pluralsight*. <https://www.pluralsight.com/guides/linear-lasso-and-ridge-regression-with-r>

*Cross-Validation Essentials in R - Articles - STHDA*. (2018, November 3). <http://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/>

D. (2020, May 17). *Simple Guide To Ridge Regression In R*. R-bloggers. <https://www.r-bloggers.com/2020/05/simple-guide-to-ridge-regression-in-r/>

Z. (2020b, November 13). *Lasso Regression in R (Step-by-Step)*. Statology. <https://www.statology.org/lasso-regression-in-r/>