DASC 5420 Final Project

Zijun Ma
MSc in Data Science
Department of Mathematics and Statistics
Faculty of Science
Thompson Rivers University
Kamloops, BC
maz22@mytru.ca

***Abstract*—This project compares different machine learning methods for analyzing a given dataset. Specifically, we evaluate the performance of the linear model, stepwise regression, Ridge regression, Lasso regression, Elastic Net, Neural Network, Bagging Tree, and Random Forests. We use statistical measures such as mean squared error to evaluate the performance of each model. Our analysis's results reveal each technique's strengths and weaknesses and show that the Random Forests method is best suited for this dataset.**

Keywords—Machine Learning

I. Introduction

The aim of this project is to identify the best machine learning model for predicting the price of used devices. We explore various techniques, including linear models, stepwise regression, Ridge regression, Lasso regression, Elastic Net, Neural Network, Bagging Tree, and Random Forests, to determine which method is most effective for this task. The findings of this study can inform future research and decision-making in the used device market, as well as contribute to the advancement of machine learning applications in the pricing domain.

II. Data

A. Dataset information

The dataset used in this project is called the "Used Phones & Tablets Pricing Dataset." It is acquired from the Kaggle website and contains 14 predictor variables and one response variable that provide information about used and refurbished devices. [10]

The variables included in the dataset are device_brand (the name of the manufacturing brand), os (the operating system on which the device runs), screen_size (the size of the screen in cm), 4g (whether 4G is available or not), 5g (whether 5G is available or not), front_camera_mp (the resolution of the rear camera in megapixels), rear_camera_mp (the resolution of the front camera in megapixels), internal_memory (the amount of internal memory or ROM in GB), ram (the amount of RAM in GB), battery (the energy capacity of the device battery in mAh), weight (the weight of the device in grams), release_year (the year when the device model was released), days_used (the number of days the used/refurbished device has been used), normalized_new_price (the normalized price of a new device of the same model), and normalized_used_price (the target variable, which represents the normalized price of the used/refurbished device). [10]

This dataset contains both numerical variables and categorical variables, and the target variable is normalized_used_price.

There are 11 numerical variables, which are screen_size, front_camera_mp, rear_camera_mp, internal_memory, ram, battery, weight, release_year, days_used, normalized_new_price, and normalized_used_price.

There are 4 categorical variables, which are device_brand, os, 4g, and 5g. The dataset's size is 278.31KB, and it contains 3,454 observations.

After careful consideration, the variable "normalized_new_price" will be removed from the dataset as it is not needed for the analysis. This variable is not relevant to the task of predicting the price of refurbished used handheld devices. By removing this variable, the dataset will have 13 predictor variables.

B. exploratory data analysis (EDA)

During the variable exploratory data analysis (EDA) process, a correlation plot is used to check for collinearity between each variable.

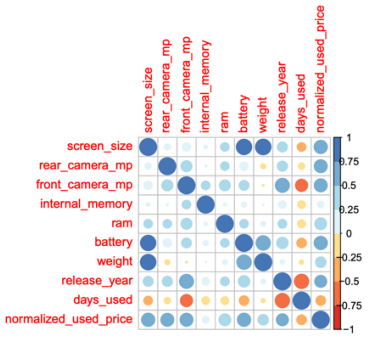

Fig.1. correlation plot

Fig.1 shows that there are high correlations between "battery" and "screen_size", "weight" and "screen_size", "release_year" and "days_used". After careful consideration, we decide to remove the variables "battery", "weight", and "release_year" from the dataset. After removing these variables, the dataset now has 10 predictor variables.

In addition to the correlation plot, a histogram was also used during variable EDA to analyze the distribution of the numerical variables.

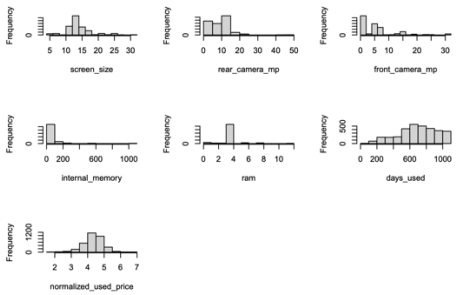

Fig.2. The distribution of the numerical variables.

Fig.2 shows that as the quality-related variables such as "screen_size," "rear_camera_mp," "front_camera_mp," "internal_memory," and "ram" increased, the count for phones decreased. This suggests that the number of used devices is less for newer or high-quality devices. It may be because people are unwilling to sell their device if the device is still new or has good quality.

Furthermore, the distribution of the target variable "normalized_used_price" followed a normal distribution.

During the variable EDA, a boxplot is used to analyze the distribution of all numerical variables.

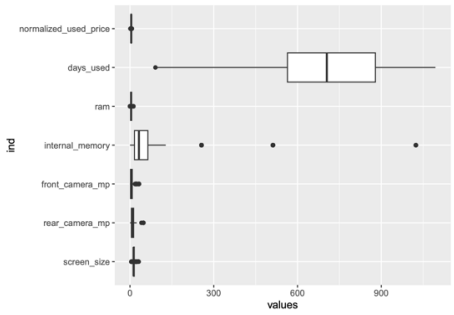

Fig.3. The distribution of all numerical variables.

Fig.3 shows the presence of outliers in some of the variables. Outliers can significantly affect the performance of certain machine learning models.

Bivariate analysis is also performed during the EDA phase by comparing the categorical variables(device_brand, os, X4g, X5g) with the target variable "normalized_used_price."

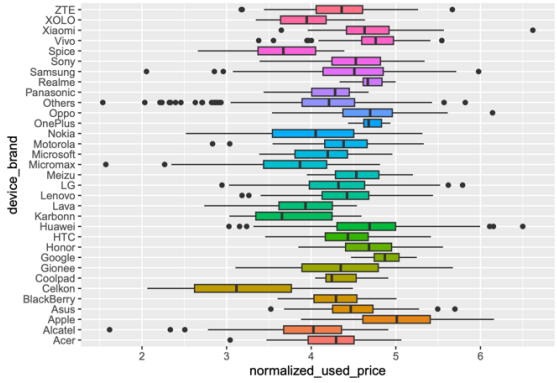

Fig.4. Compare the device_brand variable and the target variable.

Fig.4 compares the device_brand variable and the target variable, and it shows that certain brands, such as Apple, tend to have higher prices for used devices. This finding suggests that the brand of the device may be an important predictor for the price of a used device.

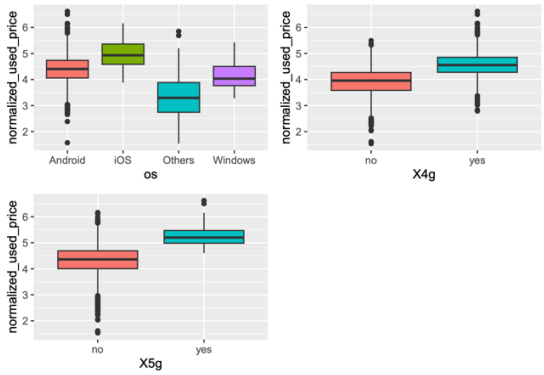

Fig.5. Compare os, X4g, X5g variables separately with the target variable.

Fig.5 shows the plot comparing os, X4g, X5g variables separately with the target variable. From the top left plot, we can conclude that devices with iOS are more expensive. From the other two plots, we can conclude that when devices with 4G/5G function, the price is more expensive.

At the same time, the numerical variables are also analyzed by comparing the variables (ram, screen_size, rear_camera_mp, front_camera_mp, internal_memory, days_used) with the target variable.

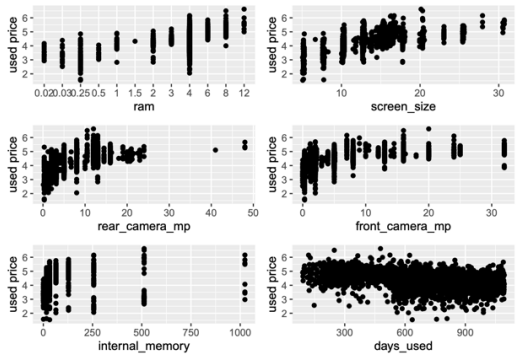

Fig.6. Compare the numerical variables with the target variable.

From Fig.6, we can see that for the variable (ram, screen_size, rear_camera_mp, front_camera_mp), as the x-axis value increases, the used price becomes higher, and for variable days_used, the older the devices, the lower the used prices. Only one variable is abnormal, which is internal_memory. From the previous correlation plot (Fig.1), we already know the correlation between internal_memory and normalized_used_price is very weak, thus from the result, we can see when the internal_memory value is from lower to higher, the price has the same variation.

C. Grouping device_brand variable

By checking the data set, we discover that the "device_brand" variable and "os" variable are highly collinear. This is because the "iOS" value within the "os" variable corresponds to the "Apple" value within the "device_brand" variable. As such, we can conclude that including both variables in the machine learning models would not be necessary and may lead to multicollinearity issues. At the same time, we find that the device_brand variable have too many categories (33), which can lead to overfitting when training the model. To address this issue, we decided to group the device brands based on each device brand's mean used_price values. We used the following rule to group the device brands:

If the mean used_price value for a brand was less than 3.7, we marked it as A.

If the mean used_price value for a brand was between 3.7 and 4, we marked it as B.

If the mean used_price value for a brand was between 4 and 4.3, we marked it as C.

If the mean used_price value for a brand was between 4.3 and 4.7, we marked it as D.

If the mean used_price value for a brand was greater than 4.7, we marked it as E.

By doing so, we are able to reduce the number of categories for the device_brand variable from 33 to 5, which should help to prevent overfitting during model training.

III. METHOD

A. Linear Regression Model

The linear regression model is a simple model that establishes a linear relationship between the input features and the target variable, which allows for the prediction of continuous values. The goal of linear regression is to find the best-fit line that minimizes the distance between the predicted values and the actual values. [1] We use mean square error (MSE) to represent the distance. Here is the formula to show the MSE.

$$MSE(\beta_0, \beta_1) = \sum_{i=1}^{n}[Y_i - (\beta_0 + \beta_1 X_i)]^2 = \sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i)^2. \quad (1)$$

B. Stepwise Rregression Model

According to Hayes [2], Stepwise regression is the iterative process of building a regression model step by step while choosing independent variables to be included in the final model. After each iteration, the potential explanatory factors are successively added or removed, and the statistical significance is tested.

There are three types of stepwise regression, which are forward stepwise, backward stepwise and two-way stepwise regression.

In a forward stepwise regression, the algorithm starts with an intercept-only model and adds one predictor variable at a time, based on which variable improves the model's fit the most, as determined by a chosen criterion such as the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).

In a backward stepwise regression, the algorithm starts with a full model containing all predictor variables and removes one variable at a time, based on which variable has the smallest impact on the model's fit, again as determined by the chosen criterion.

Two-way stepwise regression is method by iteratively adding and removing predictor variables. It is called "two-way" because it considers both forward and backward steps in the model-building process.

C. Ridge Regression Model

According to the bias-variance tradeoff, if the model fits the training dataset well, showing that the model's bias is very small, it may result in a large variance, corresponding to poor model prediction performance. The regularization method can solve this problem. By adding the penalty part to the loss function, we may efficiently minimize the bias of the model and hence raise the variance. The formulas of Ridge Regression model are shown below.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2 = |y - X\beta|_2^2 + \lambda|\beta|_2^2 \quad (2)$$

The penalty part of the Ridge Regression is the sum of the squared β, also known as the L2-norm, and the λ controls the amount to penalize. Using the cross-validation (CV) method, we can determine the best λ value that minimizes the MSE. After that, we apply the best λ value to the ridge regression model, perform model prediction on the test dataset, and calculate the MSE.

Here's how a CV works: there are two types of CVs: K-fold CVs and LOOCV (leave one out cross-validation). For the K-fold CV, we will divide the dataset into K pieces, selecting one piece as the testing dataset and the remaining K-1 pieces as the training dataset. And if we repeat this process K times, we can obtain an MSE value each time. We have K MSEs after K times, and calculate the mean of all these MSEs. LOOCV is a K-fold CV, where K is the sample size of the dataset.

D. Lasso Regression Model

For Lasso regression model, the formula of Lasso Regression model is shown below.

$$\sum_{i=1}^{n}\left(y_i - \beta_0\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j| = ||y - X\beta||_2^2 + ||\beta||_1,$$

From formula, we can see the penalty part is the sum of the absolute β, also known as L1-norm.

The idea behind applying the CV method to obtain the best λ value is similar to ridge regression. The differences between the two regressions are: ridge regression reduces the coefficients on all predictors, especially when all predictors have an equal impact on the response variable. However, Lasso Regression tends to select predictors with large coefficients and shrink the coefficients of the remaining predictors to zero. That is why we can use Lasso Regression to perform the model selection.

E. Elastic Net Regression Model

The formula for Elastic Net Regression is shown below,

$$\sum_{i=1}^{n}\left(y_i-\beta_0\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\left[(1-\alpha)\sum_{j=1}^{p}\beta_j^2+\alpha\sum_{j=1}^{p}|\beta_j|\right]=RSS+\lambda\left[(1-\alpha)||\beta||_2^2+\alpha||\beta||_1\right]$$

$$=|y-X\beta||_2^2+\lambda\left[(1-\alpha)||\beta||_2^2+\alpha||\beta||_1\right]$$

From above formula we can see, this method generates a regression model that is penalized with both the L1-norm (Lasso) and the L2-norm (Ridge). And the α value controls the mixture of the two penalties, or how much weight is assigned to each of the L1 and L2 penalties.

The α value is between 0 to 1. When α is 0, Elastic Net Regression assigns all weight to the L2-norm, resulting in ridge regression. When α is 1, Elastic Net Regression gives all weight to the L1-norm and transforms into lasso regression. We can determine the best α value using CV. Because there is no built-in function in R to find the best α, we write our CV function, first choosing a range of α values, then running cv.glmnet() function in R to get the MSE values corresponding to each α. We can find the best α value (corresponding to smallest MSE), and then we apply the best α value to cv.glmnet() function again to get the best λ value. In R, the α and λ is hyperparameter for function cv.glmnet().

F. Neural Network

According to Chen [3], A neural network is a machine learning algorithm that can be used for regression and classification tasks. It works similarly to the neural network of the human brain. A neural network contains multiple layers of interconnected nodes. The "neuron" in a neural network is a mathematical function, called an activation function, and they perform complex computations, including nonlinear computations. These activation functions introduce nonlinearity into the model, allowing it to learn complex relationships between input and target variables. The most used activation functions include the sigmoid function, the ReLU function, and the softmax function. Neural networks are often used in tasks that require a high degree of accuracy and the ability to learn and generalize from large amounts of data.
In R, we use neuralnet() function from neuralnet package. The hyperparameter we use is hidden, we set its value to (5, 3), which mean the model has two hidden layers, and first hidden layer has 5 nodes, second has 3 nodes.

G. Bagging Model

Bagging is a machine learning technique that uses a set of decision trees for prediction. Decision trees are non-parametric supervised learning methods used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from data features. It has a hierarchical tree-like structure composed of a root node, branches, internal nodes, and leaf nodes. Decision trees are high-variance models, meaning that slight changes in the training data can lead to a completely different model, and decision trees are often over-fitted.

To overcome this problem, ensemble techniques like bagging can be used. Bagging, short for bootstrap aggregating, refers to building different models using subsets of the samples and then aggregating their predictions to reduce variance. The idea behind bagging is that if we have a set of n independent observations, such as X1, X2....Xn, the variance of a single observation is σ2. The mean of all data points

will be (X1+X2+....+Xn)/n, and the variance of that mean will be σ2/n. Therefore, as we increase the number of data points, the variance of the mean will decrease. Similarly, if we train multiple decision trees on a given dataset and aggregate their predictions, the variance will decrease. Based on this idea, we create multiple subsets (bootstrapped sample) from the training data, where the data within each subset is randomly chosen with replacement. We can then build multiple trees on each bootstrapped sample. Now we have different bootstrapped samples and have built many decision trees for each bootstrap sample. The next step is to aggregate the output. For each data point in the test set, the target value is calculated from the average of all predictions from all regression trees. [4, 5]

The strategy we use to calculate the bagging error is called Out-of-Bag (OOB) error. In bagging, we create multiple subsets of the training data to train different models. Because the subset data are randomly and with replacement chosen from training data, there may exist some data point are not be chosen, these data are known as the out-of-bag (OOB) data. Then, we can apply these OOB data to evaluate the bagging model without needing to set aside a separate validation set. [5, 6]

In R, we use bagging() function from ipred package. The hyperparameter we use is nbagg (the number of bagged models to fit), nbagg.cv (cross-validation). [7]

We set a range of nbagg values, and by using for loop, we can find the best nbagg value with minimum OOB error value. Then we use this value to train our bagging model.

In R, we also use the varImp() function from caret package to visualize the importance of the predictor variables by calculating the total reduction in RSS (residual sum of squares) due to the split over a given predictor, averaged over all of the trees. The larger the value, the more important the predictor. [8]

H. Random Forest

Random forest is a type of ensemble learning method that builds multiple decision trees and aggregates their predictions. Similar to bagging, it uses bootstrap sampling to create subsets of data from the training set. However, in random forest, each decision tree is built using a randomly selected subset of features from the original set of features. This introduces randomness in the feature selection process and reduces the correlation between individual trees, leading to a more diverse set of trees. [9]

In R, we use ranger() function from ranger package. The hyperparameter we use is mtry (the number of variables to be selected randomly at each split), min.node.size (the minimum size of terminal nodes. Nodes with fewer observations than this value will not be split further), replace (determines whether or not sampling is done with replacement), sample.fraction (the fraction of the training data to be used for growing each tree).

We set mtry (1, 2, 3), because for regression problem, the mtry value is often choose p/3, p is the number of predictor variables. For our dataset, p is 9.

And set min.node.size is 10 or 20.

Setting replace is TRUE or FALSE, because our data set has categorical features that are highly imbalanced. For example, the categorical variable os has much more devices number for Android compare to other os, which is high imbalanced. And sampling without replacement likely improves performance.

We make a combination of all possible hyperparameter value, by using for loop to train the model with each combination. For each combination, a OOB error will be produce too. Then we find top 10 combination base on the OOB error values' increasing order.

IV. RESULT

A. *Linear Regression Model*

```
##
## Call:
## lm(formula = normalized_used_price ~ ., data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.36397 -0.16610 -0.00773  0.15076  1.29135
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.254769   0.054470  78.113  < 2e-16 ***
## osiOS             0.219217   0.087778   2.497 0.012615 *
## osOthers         -0.103436   0.070333  -1.471 0.141584
## osWindows         0.026632   0.037806   0.704 0.481253
## X4gyes            0.060997   0.016459   3.706 0.000218 ***
## device_brandB     0.039882   0.048997   0.814 0.415796
## device_brandC     0.079948   0.048500   1.648 0.099471 .
## device_brandD     0.124532   0.047999   2.594 0.009564 **
## device_brandE     0.135564   0.067967   1.995 0.046267 *
## screen_size       0.225357   0.013359  16.870  < 2e-16 ***
## rear_camera_mp    0.226308   0.010485  21.584  < 2e-16 ***
## front_camera_mp   0.087396   0.019237   4.543 5.97e-06 ***
## internal_memory   0.310321   0.071192   4.359 1.39e-05 ***
## ram               0.106661   0.016408   6.501 1.08e-10 ***
## days_used         0.033853   0.008542   3.963 7.74e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2509 on 1543 degrees of freedom
## Multiple R-squared:  0.5885, Adjusted R-squared:  0.5847
## F-statistic: 157.6 on 14 and 1543 DF,  p-value: < 2.2e-16
```

Fig.7. The summary of linear regression model.

From fig.7, we can noticed that not all the variables are significant important. However, all the numerical variables are significant important. And the MSE is 0.0649533, and adjust R square error is 0.5847265.

4.2 stepwise regression model

```
##
## Call:
## lm(formula = .outcome ~ osiOS + osOthers + X4gyes + device_brandC +
##     device_brandD + device_brandE + screen_size + rear_camera_mp +
##     front_camera_mp + internal_memory + ram + days_used, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.36448 -0.16702 -0.00788  0.15146  1.28840
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.293247   0.031323 137.066  < 2e-16 ***
## osiOS             0.218893   0.087754   2.494 0.012721 *
## osOthers         -0.104804   0.070253  -1.492 0.135952
## X4gyes            0.061780   0.016428   3.761 0.000176 ***
## device_brandC     0.046670   0.020547   2.271 0.023260 *
## device_brandD     0.088459   0.019005   4.655 3.52e-06 ***
## device_brandE     0.099722   0.051636   1.931 0.053634 .
## screen_size       0.224824   0.013295  16.910  < 2e-16 ***
## rear_camera_mp    0.226798   0.010472  21.657  < 2e-16 ***
## front_camera_mp   0.085899   0.019172   4.480 8.00e-06 ***
## internal_memory   0.320903   0.070277   4.566 5.36e-06 ***
## ram               0.105857   0.016386   6.460 1.40e-10 ***
## days_used         0.034009   0.008534   3.985 7.06e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2508 on 1545 degrees of freedom
## Multiple R-squared:  0.5881, Adjusted R-squared:  0.5849
## F-statistic: 183.9 on 12 and 1545 DF,  p-value: < 2.2e-16
```

Fig.8. The summary of stepwise regression model

From fig.8, we can noticed that stepwise regression remove the osiOS, osWindows, device_brandB variable from the model compare to the linear model, and those variables in linear model are not significant important. And the MSE is 0.0650784, adjust R square error is 0.5849493.

4.3 Ridge regression

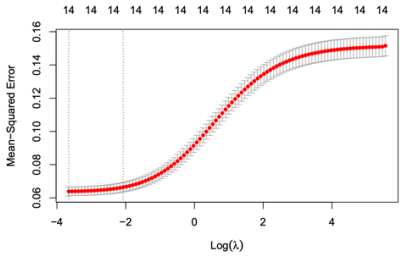

Fig.9.

From Fig.9, we can find best λ value (0.02584945). Later, we use the train data set to get the MSE, which is 0.06479813.

4.4 Lasso regression

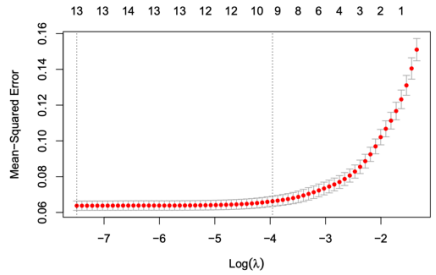

Fig.10.

From Fig.10, we can find best λ value (0.0005569095). With similar process, we get the MSE is 0.06498742

4.5 Elastic Net regression

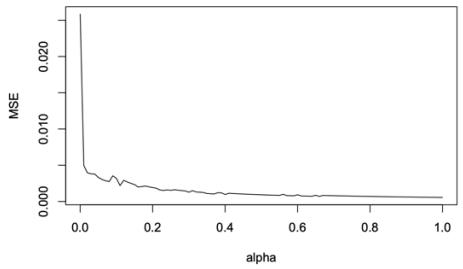

Fig.11.

From Fig.11, we can find the best α value (which is 1), it make the model become the Lasso regression. And the MSE is same as lasso MSE, which is 0.06498742

4.6 Neural Network

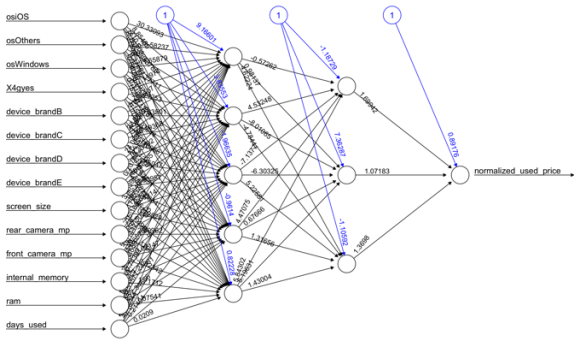

Fig.12.

The above figure shows the weights and bias between each nodes. And the MSE is 0.06498484.

4.7 Bagging

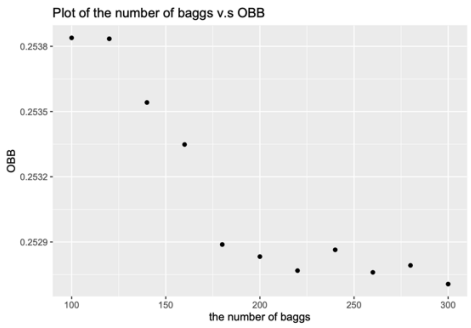

Fig.13. the error curve for bagging

Fig.13 shows the error curve for bagging 100-300 deep, unpruned decision trees. The benefit of bagging is optimized at 300 trees although the majority of error reduction occurred within the first 220 trees.

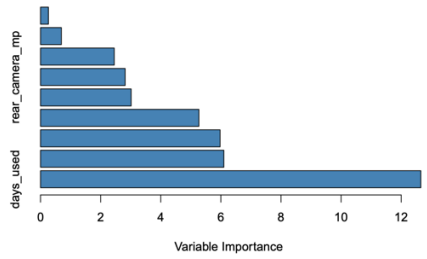

Fig.14.

From fig.14, we can see that days_used is the most importance predictor variable in the model while os is the least important.

And the MSE is 0.06292119.

4.8 Random Forest

```
##    mtry min.node.size replace sample.fraction       OOB
## 1     3            20    TRUE            0.50 0.2474806
## 2     3            20   FALSE            0.50 0.2482197
## 3     3            20    TRUE            1.00 0.2485383
## 4     3            10    TRUE            0.50 0.2485693
## 5     3            20    TRUE            0.63 0.2488084
## 6     3            10   FALSE            0.50 0.2488370
## 7     3            20   FALSE            0.63 0.2490910
## 8     3            20    TRUE            0.80 0.2491526
## 9     3            10    TRUE            0.63 0.2494640
## 10    3            10    TRUE            0.80 0.2495774
```

Fig.15.

From above figure, we can see that the model has smallest OOB error value, with mtry value is 3, min.node.size is 20, replace is TRUE, sample.fraction is 0.5. After we apply these value to model, and apply this model to testing data set, we get MSE is 0.05887985.

4.9 Comparing MSE results among all models

TABLE I. MSE VALUES AMONG ALL METHODS

Method	**MSE**
Linear Regression Model	0.0649533
Stepwise Regression	0.0650784
Ridge Regression	0.0647981
Lasso Regression	0.0649874

Method	MSE
Ekastic Net Regression	0.0649874
Neural Network	0.0649848
Bagging	0.0630723
Random Forest	0.0588799

From above table, we notice that the Random Forest give us the smallest MSE.

V. Discussion and Conclusion

The purpose of this project is to determine the most effective machine learning model to predict the price of used devices. We explored a variety of techniques including linear models, stepwise regression, Ridge regression, Lasso regression, Elastic Net regression, Neural Network, Bagging trees, and random forests to determine which method was best suited for this task. After running and comparing the performance of each model, we found that Random Forest performed the best, producing the lowest mean square error which indicates that Random Forest is the most effective model for predicting used equipment prices.

References

[1] Greenwell BB & B. Chapter 4 Linear Regression | Hands-On Machine Learning with R [Internet]. bradleyboehmke.github.io. [cited 2023 Apr 16]. Available from: https://bradleyboehmke.github.io/HOML/linear-regression.html

[2] Stepwise Regression [Internet]. Investopedia. 2019. Available from: https://www.investopedia.com/terms/s/stepwise-regression.asp

[3] What Is a Neural Network? [Internet]. Investopedia. [cited 2023 Apr 16]. Available from: https://www.investopedia.com/terms/n/neuralnetwork.asp#toc-understanding-neural-networks

[4] scikit learn. 1.10. Decision Trees — scikit-learn 0.22 documentation [Internet]. Scikit-learn.org. 2009. Available from: https://scikit-learn.org/stable/modules/tree.html

[5] Chelliah I. Bagging Decision Trees — Clearly Explained [Internet]. Medium. 2021. Available from: https://towardsdatascience.com/bagging-decision-trees-clearly-explained-57d4d19ed2d3

[6] Greenwell BB & B. Chapter 10 Bagging | Hands-On Machine Learning with R [Internet]. bradleyboehmke.github.io. Available from: https://bradleyboehmke.github.io/HOML/bagging.html

[7] bagging function - RDocumentation [Internet]. www.rdocumentation.org. [cited 2023 Apr 16]. Available from: https://www.rdocumentation.org/packages/ipred/versions/0.4-0/topics/bagging

[8] Zach. How to Perform Bagging in R (Step-by-Step) [Internet]. Statology. 2020. Available from: https://www.statology.org/bagging-in-r/

[9] Boehmke B. Chapter 11 Random Forests | Hands-on Machine Learning with R [Internet]. Github.io. 2019. Available from: https://bradleyboehmke.github.io/HOML/random-forest.html

[10] Used Phones & Tablets Pricing Dataset [Internet]. www.kaggle.com. Available from: https://www.kaggle.com/datasets/ahsan81/used-handheld-device-data