Ind Eng 142A Final Project - Laptop Price Prediction

Khoa Nguyen, Yusuf Ahmad, Melad Mayaar, Safwan Masood, Yuki Kuwahara, TM Carillo

Motivation and Impact

In today's world, we are constantly surrounded by technology. Whether it be smartphones, laptops, desktop computers, tablets, smart watches, etc, it is everywhere. And as students at a university, laptops are a crucial part of our academic career, as it allows us to take notes, complete assignments, and even communicate with peers and professors. For us laptop users, although it can be sometimes challenging, we can simply look at a laptop from its brand, specifications, and its pricing to make a decision. But the harder challenges come from the manufacturers side, especially the new manufacturers and brands who are trying to break into the laptop market to come up with a reasonable price and to be profitable.

The easiest choice those manufacturers can make is to simply slap on a random price and try to sell it to customers. But as mentioned, it needs to be profitable, but also a price that is reasonable and competitive enough with the respective specifications that it would make a customer believe they are making a good financial decision to make a purchase. Thus, our project aims to help these new manufacturers who are trying to break into the laptop market to be able to evaluate and decide on a price for their laptop models based on their specifications. We will build and evaluate several models, ranging from simple linear regression models to decision tree regressor models to see which models are the best suited for this task. We will also be trying to improve our model with various techniques to get a desirable model we will be satisfied with.

Our hope for this project is to make a positive impact on the new manufacturers who are trying to break into the laptop market to be able to be profitable as well as make good sales. In the world of laptops, where there are already big names such as Apple, Dell, and Acer, it can be quite difficult to navigate and get a share of the market, and even come up with a reasonable price tag. But we hope that our project can shed light on these manufacturers to make a good product for the general consumers at a good price.

Dataset

We have used two data sources to create the final data frame that was used to build our models. One of the data sources is from Kaggle, simply called the "Laptop Price", created by Muhammet Varli. The description of this dataset simply says "Laptop Company Price List for Regression". Not much is stated, thus making us believe that the use case for this dataset would mainly be used for very basic data analysis and linear regression models. Though we are on the same line, our goal, as stated above, is to create a more complex but better model that will actually be

usable in the real-world, especially with the additional data that we will add, which we will discuss right after.

This dataset had most of the important specifications of a laptop, such as the screen size, storage capacity, CPU and GPU model, and screen resolution, alongside with the price of the laptops, which made us decide to pursue this dataset. However, we felt that there was one important thing that was missing, which was the number of cores that the CPU has. This is important because the number of cores in a CPU is very crucial for the laptop's performance, and prices are heavily influenced by this in the market. Thus, we decided to find data that had the specifications of CPUs and add the number of cores based on the model to the data we originally found. The dataset we ended up using was halaalfaris' "Intel_CPUs.csv" dataset. This was perfect because we were able to extract the CPU model (such as i7-7700HQ) using regex from the original data and join the data together to match the core counts (and most of the dataset had Intel CPU laptops, as Intel holds the biggest share in terms of Laptop CPUs).

Additionally, we have also modified our dataset by manipulating some of the variables, or columns. For example, we were able to scrape the resolutions of the screens of the laptops as resolution height and resolution width by using helper functions that included regex. For example, "IPS Panel Retina Display 2880x1880" became 2880 in the resolution height variable and 1880 as the resolution width (screen type was not extracted due to extremely variable screen types based on manufacturers and inconsistency with the text). We were also able to extract CPU speeds from the CPU columns, such as "Intel Core i5 8250U 1.6 GHz" to "1.6" (GHz), which should make our modeling process a little more efficient.

Lastly, we also dropped columns that we deemed unnecessary. This is not the same as dropping columns based on VIF values when creating a linear regression model. For example, columns such as laptop_ID, Company, Product (Name), did not make sense to be included as our goal is to figure out the prices from specifications of the laptop. We did discuss whether or not to keep the type of the laptops (such as ultrabook, notebook, or gaming laptop), but we have decided to keep it as different laptop types can yield to different prices. We also converted the prices of the laptop from Euros to USD, and weight of the laptops from kilograms to pounds as it is the standard metrics that we use here in the United States.

Analytics Models

Since laptop price is a continuous variable, we opted to use various forms of regression models to predict it. Prices for laptops in this dataset, specifically, have a fairly large range with a minimum of $316 to a maximum of $3,083. We also initially started with 11 independent variables to input into our models. The 3 categorical independent variables were TypeName, OpSys, and GPUBrand. The 8 numerical independent variables were Inches, Speed (GHz),

Storage (GB), Ram (GB), Weight (lbs), Cores, ResolutionWidth, and ResolutionHeight. We effectively sought to predict the vector 'Price_usd'.

Multiple Linear Regression

Our initial strategy for modeling laptop prices was to employ a multiple linear regression model due to its simplicity. The first step in this process was to visualize any correlation between the numerical, continuous variables (Inches, Speed(GHz), Storage(GB), Ram(GB), Weight(lbs), Cores, ResolutionWidth, ResolutionHeight, and Price (USD)) using a heatmap. From this, we found that Price and Ram had a fairly strong positive correlation, and Price and Weight had a fairly strong negative correlation. This tells us that Ram and Weight are most likely important factors. Although we could not make any definitive conclusions from this, it gives us good insight into the factors that might determine price.

For the first pass at linear regression, we used a very basic model with all 11 variables. We attained an OSR2 of .68. To improve this we tried to use the statsmodel version of linear regression. We first inputted all or a combination of the categorical variables (TypeName, OpSys, and GPUBrand). However, doing this ended up leaving us with R2 and OSR2 scores between .57 and .61 – which was not high enough.

In efforts to increase these metrics, we stuck with only using the 8 continuous variables. We built Model0, the first model, with all 8 continuous variables and attained an R2 of .70 and OSR2 of .67. We then tried to cut certain variables using Variance Inflation Factor (VIF) to see if there was high multicollinearity. By doing VIF on all 8 continuous variables, we found that there was very high multicollinearity between ResolutionHeight and ResolutionWidth. This would make sense as both are interrelated. We ended up scrapping Resolution Height. We then built Model1 with only 7 of the variables and landed an R2 of .70 and an OSR2 of .69 – which were very good up to this point.

We continued the process of running VIF and decided to drop the Cores feature for Model2, which ended up leading to slightly lower accuracies of .695 (R2) and .67 (OSR2). Running VIF again, we dropped Inches and Weigh for Model3. This led us to even lower scores of .598 (R2) and .60 (OSR2). After this, we started to use various combinations of the variables to build model4 and attained higher R2 and OSR2 scores of .696 and .675. We concluded the linear regression here as our results were becoming stagnant, with Model1 being the best performing. In the future, to improve these scores, we could try to expand on our feature engineering through scaling or doing some form of regularization and cross-validation.

Decision Tree Regressor

To deepen our analysis, we chose to apply a Decision Tree Regressor Model in hopes of improving our overall prediction performance. We first imported the DecisionTreeRegressor from sklearn and fit our training set to it. Our hyperparameters were 5 leaf samples, a ccp alpha

of 0.01, and a random state of 88. We ran the model and received a tree with 89 nodes. This model gave us an OSR2 of .64 – which was not too bad.

To improve this Decision Tree model, we leveraged Principal Component Analysis (PCA) to find the optimal amount of features. From this, we found that the optimal number of features is 5. And using these 5 features we attained at an OSR2 of .54. This strategy ended up not working out as well as we thought. We attained a better OSR2 by just using a plain Decision Tree Regressor. In the future, to improve this accuracy, we can tweak the hyperparameters like max_depth, ccp_alpha, min_samples_split, and max_leaf_nodes. Another thing we could possibly do is doing more extensive feature engineering.

Random Forest

To build upon our decision tree regressor strategy, we leveraged a Random Forest Regressor in hopes to improve the OSR2. The reason why we chose this is because it is an ensemble method – which uses decision trees in a collective way to improve the predictive performance and reduce the risk of overfitting. To apply this model, we used the hyperparameters of 5 max_features, 5 min_samples_leaf, 300 n_estimators, and a random_state of 88. By doing this, we attained a higher OSR2 value of .722 – leading us to have our best performance so far.

To improve this further, we could use some form of bootstrapping or tweaking the hyperparameters to find the optimal model. Another way we can improve it is using advanced feature engineering or transformations like log, square, and square root features that do not conform to normality – helping us catch non-linear relationships.

Gradient Boosting

To cap off our modeling portion, we chose to apply gradient boosting. This is when multiple decision tree models are trained sequentially to improve model performance. This seemed like a very good option to improve our basic decision tree model since it is very flexible and good with handling various data types. And in our case, specifically, we have both numerical and categorical data within our training and test sets.

For our gradient boosting model, we used a range of n_estimators from 1000 to 5000 with a learning rate of .0001, max_leaf_nodes of 3, max_depth of 15, 10 min_samples_leaf, a random_state of 88, and a verbosity of 1.

When applying this model, we obtained our best performance with an OSR2 of .74 across 4000 boosting stages. Out of all the models that we used in this project, this gave us the best outcome most likely due to the fact that it handles various types of data very well.

Another step that we can take in the future is to tweak the hyperparameters and combine some other form of ensemble methods like stacking, while being careful with overfitting.