Machine Learning Project Milestone 2: Laptop Price Estimation

Rana Dbouk
rana.dbouk01@lau.edu

Mohammad Khalifeh
mohammad.khalifeh03@lau.edu

Omar Al Itani
omar.alitani01@lau.edu

December 4, 2022

Abstract

This study aims at finding a machine-learning model that estimates laptop prices based on specs and features. The reason behind choosing this topic is strongly related to our majors. As computer science students, we are constantly looking for laptops with certain specs that meet our requirements, whereas people often don't have the expertise to find a laptop within their budgets. Therefore, we aimed to build a machine learning model that estimates a laptop's price based on given specs. Specs taken into consideration in our model are carefully selected based on different approaches to eliminate features that are not relevant to our output or causing multicollinearity with other features. During our experiments, we build different regression models to better understand the underlying trends in our dataset. Therefore, we implemented Linear Regression, Lasso Regression, K-Nearest Neighbor, Support Vector Machine, and Random Forest models. After fitting our models, linear regression and support vector machine using a linear kernel provided accurate results, thus, validating our initial assumption that linearity exists between some features and our output. Our best performing model was Random Forest with an R^2 score value equals 0.81.

1 Introduction

Predicting laptop prices is a crucial and significant endeavor, particularly when the laptop demand increased after the nationwide lockdown. Additionally, the huge enthusiasm for laptops that we witnessed in 2020 to facilitate distant work and learning also increased the worldwide demand on laptops. Since prices typically depend on a variety of distinctive qualities and factors that are updated after each release, price prediction requires specialist knowledge of these specs. Particularly, features that have the most significant effect on a laptop's price are brand and model, RAM, ROM, GPU, CPU, etc. Therefore, estimating a laptop's price given the specs can be a crucial task, taking into consideration that not everyone is interested or knowledgeable in laptop features; Therefore, choosing a new laptop may be tricky for people on a budget and searching for specs, and can be harder for those with no prior knowledge of laptop prices. The motivation behind this study as computer science students having laptops as a necessity, is to provide a machine learning model that will help anyone estimate the budget they must have to purchase a computer with the specs they want.

2 Related Work

Laptops Price prediction has been studied in various research using different approaches. Vasishali et al. achieved 81% prediction precision using a supervised machine learning approach which is C4.5 (used as a Decision Tree Classifier algorithm) that lists the features as tree elements from root till branches based on their Entropy and Gain in order to complete its calculation.

Another research was done by Andy Guozhen who tested 4 different models that are Random Forest Regressor (with different number of nodes), Decision Tree Regressor, Linear Regression Model, and XGBoost Model (XGBRegressor). The models' performance was estimated by the R-squared value and the Mean Absolute Error (MAE) in which the goal is to achieve a high R-squared and a low MAE. Based on the result of the models, the XGBoost model achieved the highest R-squared

and lowest MAE score among all models where it was selected as the web app final machine learning model.

3 Proposed Method

Multiple regression models were explored and implemented.

3.1 Multiple Linear Regression

Multiple linear regression is a parametric regression algorithm that assumes a linear relationship between multiple independent variables. The model will weigh the effect of each variable on the output while minimizing its cost function. $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ The cost function represents the difference between the real output and the predicted output squared, and the minimization of this equation can derive the equation for the betas of the model:
$\beta = (\mathbf{X^T X})^{-1}.\mathbf{X}^T Y$

3.2 Lasso Regression

Lasso regression is a variation of linear regression while regularizing the cost function. Linear regression can generate large betas and cause overfitting on the training data, this issue is resolved in Lasso regression. The difference in the new cost function is that lasso regression minimizes the summation of error added to it the summation of the absolute value of betas multiplied by a certain constant lambda. This means that betas will be controlled based on the value of the lambda while minimizing the squared error. It is also known as L1 regularization since the betas can reach zero, hence removing the feature and applying feature selection on the variables.
$Loss = Error(Y - \widehat{Y}) + \lambda \sum_1^n |\beta|$

3.3 KNN: K-Nearest Neighbor

The loss function of a k-NN cannot be minimized during training. Actually, this algorithm has never been trained and doesn't have a loss function. For k-NN, the only "training" that takes place is memorizing the data (making a local copy) so that it can find the average values of the k-neighbors during the prediction of a new point. Technically, there is no optimization because no function is fitted to the data (it cannot be trained using gradient descent).

3.4 SVR: Support Vector Regression

When using SVR, the goal is to essentially take into account the points that are inside the decision boundary line. The hyperplane with the most points is the one that has the best fit line. Decision boundaries can be located at any distance from the hyperplane, let's say 'a'. Thus, decision boundaries are lines that are '+a' and '-a' from the hyperplane. The primary goal in this case is to choose a decision boundary that is located at a distance from the initial hyperplane such that the data points that are closest to the hyperplane or the support vectors fall within that boundary.

3.5 Random Forest

Random Forest grows multiple decision trees that are merged. It uses bagging (choosing a subset of the training data) and feature randomness when building each tree to find the uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. The loss function can be explicitly specified, such as the mean squared error (MSE) or R^2 score.

4 Experiments

4.1 Dataset

We obtained our dataset from the following Kaggle project. Initially, the dataset file was composed of about 900 samples, all of which have 23 features. However, with data preprocessing, these numbers will change.

4.1.1 Removal of Irrelevant Features

The first thing we did was remove some features that were irrelevant to our study. These features were the old price, discount, reviews, star ratings, and ratings.

4.1.2 Fixing the Output Feature

Then we reordered the columns to have our output feature the latest price as the last column of the table. And we also converted the price values from Indian Rupee to the US dollar.

4.1.3 Mistake in Pandas Data Types

Another step we did was converting the data types of some of the columns since they were incorrectly inferred as object types instead of numerical types by the "pandas" library.

4.1.4 Treatment of Missing Values

We had three features in our dataset that had some missing values. And for every one of those columns, there was a different solution. The "mode" column's missing values were marked as null. We considered them to be a new categorical value for that feature. The "display size" column's missing values were replaced by the average display size value. And for the "processor generation" column, we manually filled in these values so that they logically assess the processor's power.

4.1.5 One-Hot-Encoding and Elimination of Outlier Data

We then performed one hot encoding to transform all categorical features into numerical data. After finishing this step, we observed a massive increase in the number of columns of our data. We realized that we had many categories that had very low frequencies. Thus, we decided to drop the few rows that contain the rarely occurring categorical values. This solution reduced the number of columns and helped us eliminate outlier data.

4.1.6 Feature Scaling

We also observed in our dataset that features had very different ranges for their values. Thus we decided to scale our input features using Python's standard scalar, which converts all columns to have a standard deviation of one and a mean of zero.

4.1.7 Backward Elimination

And for the last step of data preprocessing, we performed Backward-Elimination to reduce the number of features in our dataset. Backward-Elimination is used in regression to eliminate the input variables that do not express any correlation with the output feature.

4.1.8 Dataset After Preprocessing

At the end of data preprocessing, we only lost about 50 rows, which is good since our dataset wasn't large enough at the start. And we got our number of features down to 18, which is good since we removed a lot of features and also had to convert the categorical features to multiple additional columns. Our final list of features after data preprocessing is the following:

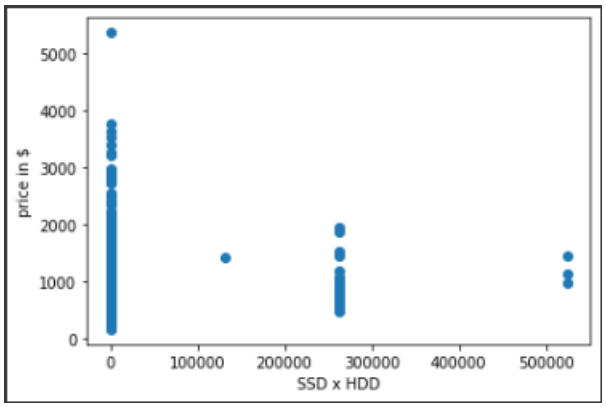
- HDD
- SSD
- RAM GBs
- Graphic Card GBs
- processor name
- processor generation
- processor brand Intel
- processor brand M1
- display size
- weight gaming
- weight thin and light
- touchscreen
- OS bit
- OS Windows
- OS MAC
- MSOffice
- warranty
- latest price (output feature)

4.2 Discovering relationships between features

Some experiments were made to discover some trends in our data, and the most important findings will be presented.

4.2.1 SSD and HDD Interaction

We suspected that laptops with both SSD and HDD might have higher prices on average, however, the results of the scatter plot in figure 1, of the interaction term (ssd*hdd) vs the prices, show no pattern whatsoever.



4.2.2 HDD to Price Correlation

- 0 HDD: count: 665.000000, mean 983.221083, std 559.082912
- 512 HDD: count 55.000000, mean 853.039818, std 720.198500
- 1025 HDD: count 164.000000, mean 645.743476, std 238.101467

Based on the above statistics, there can be a negative relationship between the laptop's expected price and the size of the HDD, since newer laptops with higher specs mainly do not have HDD and rather provide faster SSD.

4.2.3 RAM to Price Correlation

- 4 RAM: count=254, mean=705.372, std=421.822
- 8 RAM: count=448, mean=807.654, std=335.256
- 16 RAM: count=180, mean=1445.912, std=700.719
- 32 RAM: count=3, mean=2053.723, std=1720.534

As ram sizes increase, the average price increases. Therefore, there is a clear positive relationship between the ram size and a laptop's price.

4.2.4 Touchscreen's effect on price

- no touchscreen count=782, mean price=874.773$
- touchscreen count=103, mean price:=1197.532$

Most laptops don't come with touchscreens. Laptops with touchscreens on average are about 300$ (big jump in price) more expensive than laptops without a touchscreen.

4.2.5 CPU Cores' effect on price:

Data and the market are dominated by Intel and AMD processor brands, and M1 processors are also used for some Apple laptops. Figure 2 shows the most frequent and important processor models for the 3 main processor brands (Intel, AMD, and Apple). It's clear that with an increase in the number of cores in the processors the price of laptops increases. (Increase in the number of cores can be deduced by an increase in the numbers in the processor name)

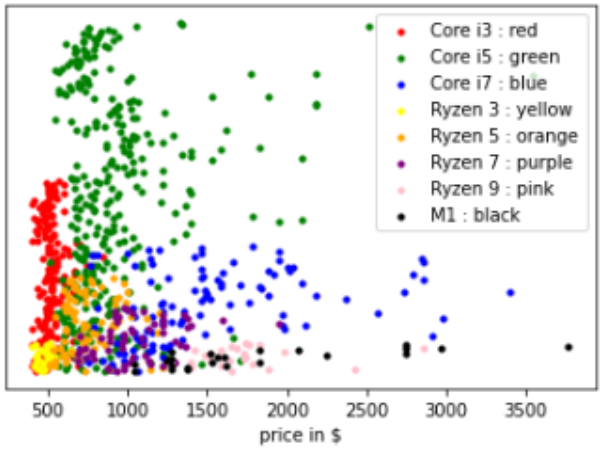


Figure 2: Cores vs. Price in $

- Core i3 intel processors are the cheapest intel laptops. Core i7 intel processors are the most expensive intel laptops. Core i5 intel processors are in the middle range of prices.

- Ryzen3 AMD processors are the cheapest Ryzen laptops. Ryzen9 AMD processors are the most expensive Ryzen laptops. Ryzen 5-7 AMD processors are in the middle range of prices.

- Core i5 seems to be similar in price to Ryzen 5-7. Same can be said between Core i3 and Ryzen 3. As well as for Core i7 and Ryzen 9.

- M1 laptops seem to be relatively more expensive than intel and AMD laptops.

4.3 Testing feature relevance with Backward-Elimination

When trying to implement backward elimination to check the highest p-values until all p-values are below 0.05, our model seemed to be completely overfitting and therefore had false p-values for our features. This might be due to many reasons, one of which is that the model may not be linear, or we must use other algorithms that better fit our model. Obviously, this issue was later resolved when discovering that our huge number of features (almost 40) caused our model to completely overfit. Therefore, we performed feature selection and carefully chose features that are directly related to our model and will affect our output. Another issue was that some categorical features created many new variables when applying one hot encoding, and drastically increased our number of features. Some of these could be converted into numerical features as some categories can be less significant than others. This helped us in reducing the number of features to 18.

4.4 Software

Python3 and google colab.
Libraries used: numpy, pandas, matplotlib, scikit-learn, model stats.api.model

5 Results and Discussion

After fitting our different models, this section will present and discuss our findings.

	Linear Regression	Lasso Regression	K-NN (with k=4)	SVR (linear kernel)	Random Forest
MSE	101015	101015	79524	85729	57529
RMSE	317.8	317.8	282	293	239
R2	0.73	0.73	0.74	0.66	0.81

Table 1: Models' regression scores on a train-test split

Linear Regression	Lasso Regression	K-NN (with k=5)	SVR (linear kernel)	Random Forest
0.69	0.69	0.67	0.59	0.76

Table 2: Models' average R2 scores on 10-fold cross-validation

Based on the above tables showing our results, the following sections will analyze each model's performance.

5.1 SVR

Support vector regression had the highest RMSE scores and lowest R2 scores in train-test splits and 10-fold cross-validation. It was our worst-performing model. We tried four different kernels for SVR:

- linear
- polynomial
- sigmoid
- radial-based function

The best performer was the linear kernel. This result is logical since we observed linear relationships between some features and the output.

5.2 Linear Regression

According to table 1, our linear regression model had an acceptable R2 score when using train-test-split, with value equals 0.73. This further proves our first assumption of linearity between different features and the predicted price. As shown in table 2, the R2 score slightly

decreased when using 10-Fold cross validation technique, and this change can be explained by our limited instances on some laptops. For example, laptops with RAM = 32 GB or laptop brands of type "Samsung" are rare in our data, and predictions on these laptops will significantly affect our model's performance in general.

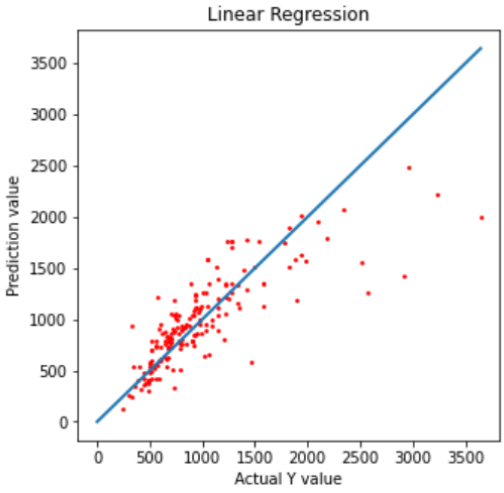


Figure 1: Actual vs. Predicted Values in Linear Regression model

Figure 1 is a plot of the real prices versus the predicted ones by our linear regression model. The line drawn represents the perfect model, which is Y= x. Of course, our aim is to have our points as close as possible to this line, and this figure shows that prices between 300$ and 1500$ are closer to the line than the prices ranging from 2000$ till 3500$. This is an important observation, since we previously mentioned that our dataset is limited, and laptops which prices above 200$ compose a small portion of our dataset, thus explaining the huge difference in the figure above.

5.3 Lasso Regression

Lasso regression model is expected to have similar results to the linear regression, as it will never achieve a higher r2 score when removing features.

5.4 KNN

KNN was tested on both Train Test Split and Cross Validation in order to find the optimal K value and R2.

Starting with Train Test Split, we tested it on 28 different K values ranging from 1 to 28, which at the end it found K=4 as the optimal K with R2=0.739.

Moving on to Cross Validation that we have prepared it earlier, it gave a different K which is 5, however, it had a lower R2 with 0.67.

It is important to note that KNN is mostly used in classification problems, however, we wanted to test whether it would provide a good estimation on regression problems. Hence, our expectations were low. Additionally, KNN is known for its curse of dimensionality, which can be resembled in our model. Our dataset has 18 features which can negatively affect KNN's .

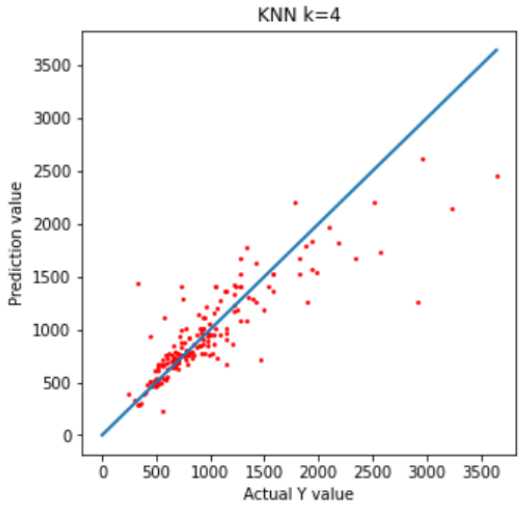


Figure 2: Actual vs. Predicted Values in KNN

Figure 2 displays the performance of KNN when K=4 on estimating the price of the laptop. One important observation is that KNN accurately predicted the prices of laptops with prices ranging from 300$ till 1000$ with a noticeably better performance than that of Linear regression, random forest, or support vector machine. However, when with prices above 1000$, its performance started decreasing and lagging behind our random forest model.

5.5 Random Forest

Random Forest has the lowest RMSE and highest R2-scores. It was tested on both Train Test Split and Cross Validation. The random forest regressor had 100 estimators on a random state of 42.

Starting with Train Test Split, it had a high result compared to the previous models with R2=0.812.

Moving on to Cross Validation, it had the highest result among all models that were used with R2=0.83.

This model has the best performance since it resembles our approach to solve a problem that has multiple conditions that are the laptop specs in our case. To clarify, we ask questions concerning the specs and based on the answer, we can shape our direction towards the laptop's estimated price.

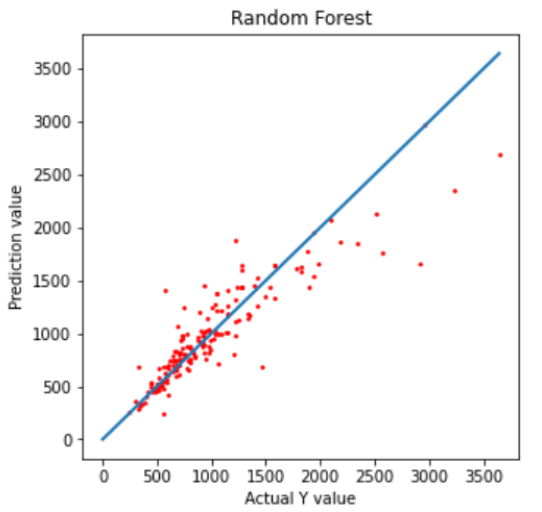


Figure 3: Actual vs. Predicted Values in Random Forest

Figure 3 displays the performance of Random Forest. Comparing it to Linear Regression and KNN figures, we can notice that the plotted data are closer to the perfect model. This observation is logical when considering referring to our results tables, as random forest had the highest R2 score and lowest error. Hence, we can conclude that the Random Forest model is the optimal model that best captured the trends in our dataset.

6 Conclusion

In conclusion, we discovered the power of Machine Learning in solving problems that require complex mathematical techniques quicker than us humans. Technology in modern days has been integrated into almost all fields, and people are constantly relying on it, especially when it comes to laptops. Therefore, our project proved to be helpful for people who are confused when buying a laptop, by saving them time spent on searching and money paid for extra specs that they may not need. The results of our models achieved acceptable performances considering the size of our dataset, and this explains the difference between our models and what previous work has achieved. Our project does not end here, where future improvement can be achieved by gaining access to datasets with more combinations of laptop models, processors, and graphics cards that can help increase the accuracy of the model and address its limitations. It is guaranteed that future laptops will integrate high specs that we lacked in our dataset, thus, improvement of our model in the near future is achievable.

7 Contributions

- Omar: Related Work, Data cleaning, Random Forest model fitting, and Conclusion.
- Mohamad: Data Transformation, Feature selection, Data Visualization, SVR model fitting.
- Rana: Introduction, Abstract, Data Visualization, Linear & Lasso Regression, and KNN model fitting.

Everyone evenly contributed in the Discussion & Results section.

References

[1] V. Surjuse, S. Lohakare, A. Barapatre, and A. Chapke. Laptop price prediction using machine learning. 2022.

[2] Guozhen, Any. (2021). Predicting Laptop Prices using ML. Retrieved from: https: //medium.com/analytics-vidhya/predicting-laptop-prices-using-ml-e60a0315b45a