Report

Laptop Price Prediction

Mini-Project
Course Data Science 2022.1

Bui Thanh Tung
Dao Van Tung
Tran Vuong Quoc Dat
Mai Ha Dat
Lecturer: Prof. Than Quang Khoat

ABSTRACT

There is a wide range of prices for different computers and for non-tech savvy people, it is easy to get overwhelmed and can be an intimidating experience to validate the price of a computer and purchase one. Therefore, we aim to predict the prices of laptops by analyzing data crawled from two websites. The findings can be useful for both consumers looking to purchase laptops and companies looking to understand the market and make informed decisions. The methodology addresses the phases of Data Preparation, Exploratory Data Analysis, Feature Engineering, Model Training and Otimization, Model Evaluation and Selection, and finally Model Interpretation. The results of this research demonstrate the effectiveness of using XGBoost for predicting laptops' prices, achieving an R2_score of 0.83 on the testing set. The feature importance analysis showed that factors such as battery capacity, RAM capacity and SSD capacity have the greatest impact on the predicted price..

Keywords. Laptop, Data Crawling, Data Preprocessing, Exploratory Data Analysis, Feature Engineering, Regression, Model Interpretation

ACKNOWLEDGEMENT

We would like to express our deep gratitude to Prof. Than Quang Khoat, who gives us this golden opportunity to work on this wonderful project. Without his tremendous support and instruction, we cannot complete this project.

Contents

1 Introduction

The laptop market is constantly evolving, with new models being released regularly and older models becoming obsolete. As a result, understanding the factors that influence laptop prices is important for both consumers and companies. Consumers can use this information to make informed purchasing decisions, while companies can use it to stay competitive in the market.

In this research paper, we aim to predict the prices of laptops by analyzing data crawled from two websites. The data was preprocessed to convert features into a standard format, and create new features through feature engineering. Exploratory Data Analysis was conducted to understand the relationships between the features and the target variable, price. After that, feature engineering was performed to create new features and transform existing ones to improve the performance of the machine learning models

Five different machine learning models were trained and evaluated to predict the laptop prices, including Decision Tree [1], Random Forest [2], XGBoost [3]. The best performing model was selected based on its evaluation metric, R2_score. The best model was further interpreted using SHapley Additive exPlanations (SHAP) [4, 5] to understand the contribution of each feature to the predicted price.

The findings of this research will provide insights into the factors that influence laptop prices and the effectiveness of using XGBoost for price prediction. The results of this research can be useful for both consumers looking to purchase laptops and companies looking to understand the market and make informed decisions. The rest of the paper is organized as follows: Section 2 describes the whole pipeline, including Data Crawling, Data Preprocessing, Exploratory Data Analysis, Feature Engineering and Modeling, Section 3 presents the experiment process, provides the interpretation of the best model (XGBoost) using SHAP and finally, Section 4 concludes the paper and provides future work.

2 Methodology

2.1 Data Crawling

Data crawling is a process of automatically extracting information from websites by making use of computer programs.

In this project, we crawl data using both HTTP requests and the Selenium web driver. HTTP requests were utilized to retrieve and extract structured data from websites, such as text and images, without the need for an interactive browser. Selenium, on the hand, was used to automate the interaction with a web browser and retrieve data from dynamic websites that rely on JavaScript to load content. We use HTTP requests for low-security, easy-to-access, and captcha-free websites, for example, laptop88.vn and thinkpro.vn here. For sites with illegal access detection, such as Userbenchmark.com, we use Selenium to ”pretend” real user behavior so that we can crawl data undetected.

In order to predict the laptop price like the original problem, we crawled all possible information related to the laptop's selling price, including 10 features: store name, laptop name, CPU name, GPU name, ram information, display information, memory information, battery capacity, connection modes, weight and price of the laptop. This data is crawled from 2 websites laptop88.vn and thinkpro.vn.

In there:

- **Display information**: contains information about monitor size, monitor resolution x, monitor resolution y, monitor panel, monitor display
- **Ram information**: contains information about ram capacity, ram type and ram's speed

In addition, we crawl more CPU and GPU benchmarks for each laptop's Desktop, Gaming, and Workstation functions for better prediction results. This data is crawled from the website userbenchmark.com.

The results after crawling are saved in 2 csv files with the following format:

- **Laptop information data**:

	A	B	C	D	E	F	G	H	I	J	K
1		name	cpu	ram	monitor	gpu	memory	battery	connectio	weight	price
2	1725	Sáº¡c Mac	Intel Core	8GB, DDR4	14", 1920	Intel Iris X	SSD 512GE	39.3WHr	1 x Type-C	1.4 kg	3590000
3	986	Ghám¿ CÃ´	Intel Core	16GB, LPD	14.4", 240	Intel Iris X	SSD 512GE	58WHr	2 x Type-C	1.743 kg	3790000
4	1386	MÃ n hÃ¬n	AMD Ryze	8GB, DDR4	15.6", 192	AMD Rade	SSD 512GE	39.3WHr	1 x Type-C	1.7 kg	7790000
5	2101	Asus Vivo	Intel Cele	4GB, DDR4	15.6", 136	Intel UHD	SSD 256GE	37WHr	1 x Type-C	0	7990000
6	3	ASUS Vivc	Intel Core	8GB, 3200	15.6", 136	Intel UHD	SSD 256GE	37WHr	1 x Type-C	1.8 kg	8990000
7	2074	Lenovo Id	Intel Core	4GB, DDR4	14", 1920x	Intel UHD	SSD 128GE	35WHr	3 x USB-A	1.6 kg	8990000
8	13	Surface Pr	Intel Core	8GB, LPDD	12.3", 273	Intel UHD	SSD 128GE	45WHr	1 x USB-A	0.77 kg	9990000
9	12	Lenovo Th	Intel Core	8GB, DDR4	14", 1920	Intel UHD	SSD 256GE	48WHr	2 x Type-C	1.6 kg	9990000
10	11	HP ZBook	Intel Xeor	16GB, DDF	15.6", 192	Intel UHD	SSD 512GE	64WHr	2 x Type-C	2 kg	9990000
11	10	Dell Inspi	Intel Core	8GB, 2666	15.6", 109	Intel UHD	SSD 256GE	42WHr	3 x USB-A	1.96 kg	9990000
12	2080	Dell Latitu	Intel Core	8GB, DDR4	14", 1920	Intel UHD	SSD 256GE	42WHr	1 x Type-C	1.52 kg	9990000
13	2081	Lenovo Id	Intel Core	8GB, DDR4	14", 1920	Intel UHD	SSD 256GE	44.5WHr	1 x Type-C	1.39 kg	9990000
14	8	HP 14s No	Intel Core	8GB, DDR4	14", 1920	Intel UHD	SSD 256GE	42WHr	1 x Type-C	1.46 kg	9990000
15	7	Asus Vivo	Intel Core	4GB, LPDD	14", 1920x	Intel UHD	SSD 512GE	42WHr	1 x Type-C	1.5 kg	9990000
16	6	Dell Vostr	Intel Core	8GB, DDR4	15.6", 192	Intel UHD	SSD 256GE	42WHr	3 x USB-A	1.96 kg	9990000

- **CPU, GPU information data**:

	A	B	C	D	E	F	G	H
1		cpu_name	number_of_core	number_of_thread	avg_bench	gamming	desktop	workstati
2	57	AMD Ryzen	4	8	0	0	0	0
3	58	Intel Core i5	4	8	69.7	70	78	57
4	59	Intel Core i5	4	8	73	73	80	62
5	60	Intel Core i3	4	8	65.5	65	72	55
6	61	Intel Core i3	0	0	68.5	68	78	59
7	62	AMD Ryzen	4	4	68	68	70	54
8	63	Intel Core i3	2	4	0	0	0	0
9	64	AMD Ryzen	6	12	70.3	70	73	64
10	65	AMD Ryzen	6	12	69.5	69	71	63
11	66	Intel Core i7	4	8	63.2	63	71	52
12	67	Intel Core i5	4	8	69.7	70	78	57
13	68	AMD Ryzen	4	8	49.4	49	53	41
14	69	Intel Core i5	2	4	66.4	66	73	57

2.2 Data Pre-processing

Data Preprocessing is a technique that transforms raw data into an understandable format. Real-world data (raw data) is always incomplete and that data cannot be sent through models as it would cause certain errors. That is why we need to clean data before sending it through a model to yield the best results. In this research, the preprocessing stage involved several steps to prepare the laptop data for analysis and modeling.

1. **Merging of datasets**: The 2 laptop files had different formats, and thus, they were preprocessed separately into the same format and then merged together. This step helped to ensure that the data was consistent and could be used for further analysis.

2. **Removal of NaN values and duplicates**: Instances with NaN values for the price feature or with more than 4 missing features were dropped. This was done to ensure that the data was clean and free of errors. Additionally, duplicate instances were also removed to further reduce the possibility of errors in the data.

3. **Standardization in the format of features**: For the features monitor, battery, weight, and price, the data was standardized into a common format for all the data. This step helped to ensure that the data was consistent and could be used for further analysis.

4. **Splitting of features**: The feature `ram` was split into `ram_capacity` and `ram_type`, and the feature `memory` was split into `HDD_capacity` and `SSD_capacity` to indicate the storage capacity of the HDD and SSD in the laptop. This step helped to make the data more readable and easier to analyze.

5. **Scoring of CPU and GPU**: The feature `gpu` was replaced by `gpu_score` to indicate the performance of the GPU, and the feature `cpu` was replaced by scores for 4 different categories: the average benchmark and how effective a CPU is for workstation, gaming, and desktop purposes. This step helped to make the data more meaningful and easier to analyze.

6. **Conversion of data types**: All features were converted to the appropriate data types for the following analysis and modeling steps. This step helped to ensure that the data was consistent and could be used for further analysis.

During the whole process, 333 instances are discarded from the sample, leaving a final sample that consisted of 872 different instances, which is 72.37% of the initial sample. Table 1 presents the 18 features constructed from the data obtained for this project.

These preprocessing steps helped in cleaning and transforming the raw data into a usable format. The preprocessing performed in this research helped in preparing the data for the prediction of laptop prices and ensured that the data was clean, complete, and free of errors. The preprocessed data was then used for the following analysis and modeling steps, including feature engineering, data normalization, and feature selection.

Feature	Data Type	Description
website	Categorical	Website that the laptop's sold at
battery	Numerical	Battery capacity (mWh)
weight	Numerical	Weight of the laptop (kg)
laptop_brand	Categorical	Brand of the laptop
monitor_size	Numerical	Size of the monitor (inch)
ram_capacity	Numerical	Memory storage space (GB)
ram_type	Categorical	Ram type that laptop's using
SSD_capacity	Numerical	Storage space of SSD (0 if laptop doesn't have SSD)
cpu_brand	Categorical	Storage space of HDD (0 if laptop doesn't have HDD)
cpu_core_num	Numerical	Number of cores of the laptop's CPU
cpu_thread_num	Numerical	Number of threads of the laptop's CPU
avg_bench	Numerical	Average score of the CPU in laptop in *userbenchmark.com*
gamming	Numerical	How good the CPU in laptop for Gamming in *userbenchmark.com*
desktop	Numerical	How good the CPU in laptop for Desktop in *userbenchmark.com*
workstation	Numerical	How good the CPU in laptop for Workstation in *userbenchmark.com*
gpu_score	Numerical	Average score of the GPU in laptop in *userbenchmark.com*
unknown_gpu	Numerical	Specify if the GPU score is missing (1 if missing, otherwise 0)
unknown_cpu	Numerical	Specify if the CPU score is missing (1 if missing, otherwise 0)

Table 1: Set of features

2.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyze data, identify the underlying pattern and bring fourth the key observation about the data. It comprises of several steps: visualizing data, detecting outliers, and summarizing the charasteristics of the data through descriptive statistic. The key objectives of EDA are detecting anomaly and identifying relationships between variables, both of which we can done via plotting, finding correlation matrix, or other types of visualization. EDA is often the first steps in any data analysis process. The insights acquires from EDA can be very beneficial to the modelling as well as informative to the subsequent experimental design steps.

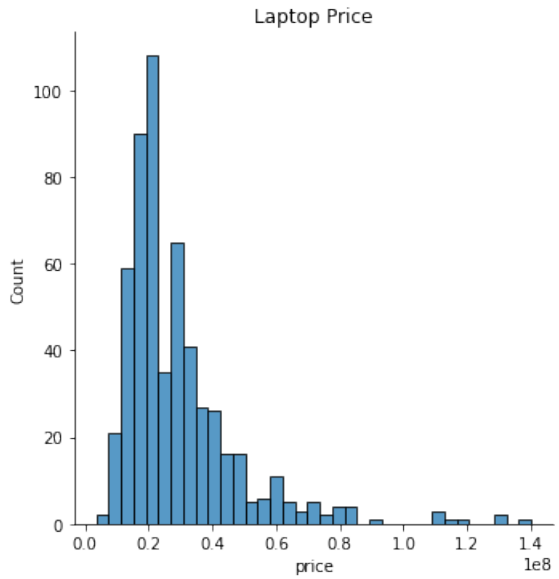

Figure 1: The target column (price) distribution

From figure 1, we can see that our data consists of laptop with price goes from 6 milion upto 140 milion. However, most of them lie in the range 15 to 30 milion. We deduce that with the exception of some workstatinn or high-end gaming laptop, computer in the above price range are suitable for most people.

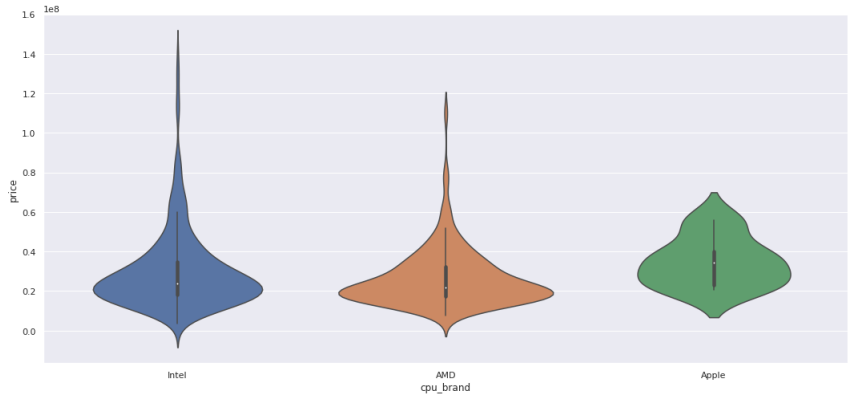

Figure 2: The violin plot of the laptop price with each cpu brand

The figure 2 demonstrates that most cpu brand have quite similar laptop price distribution. Laptops with Intel and AMD cpu have price around 20 milion while that of laptops with Apple cpu is a little higer, at 25 milion. However, Intel and AMD cpu laptops' price distribution has higher variance because expensive high-end gaming laptop, worksation as well as moderate work laptop are all use these two cpu brands. We think that Intel and AMD produce a wide range of cpu with many purpose while Apple only foucs on a small section of the market.

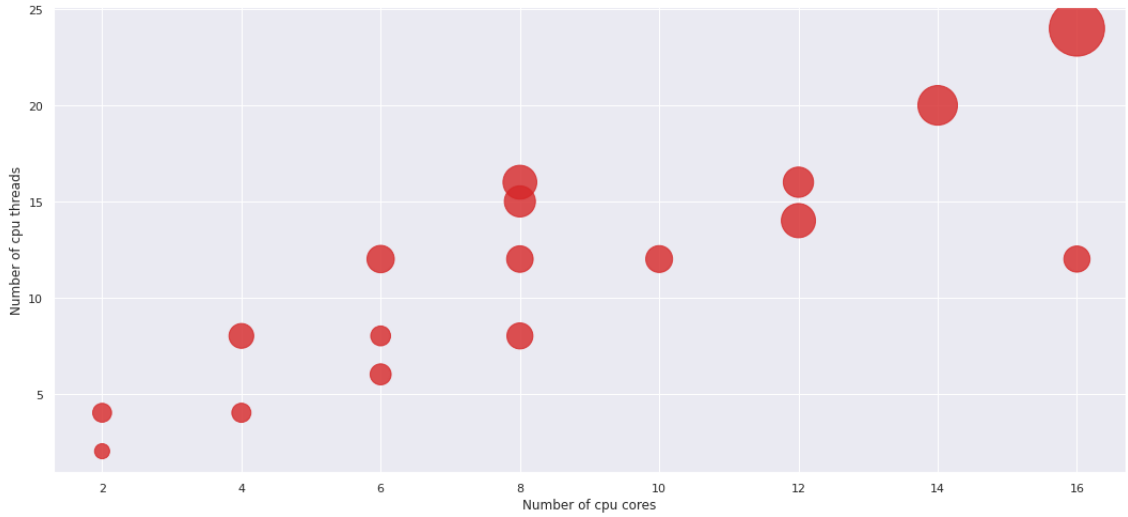

Figure 3: Average laptop price based on cpu threads and cpu cores

Beside cpu brand, the number of thread and the number of cores have strong positive correlation with laptop price. From the figure 3, it is evident that the more cpu cores and threads a laptop has, the higer its price. For example, laptop with 16 cores and 14 threads has the have price.

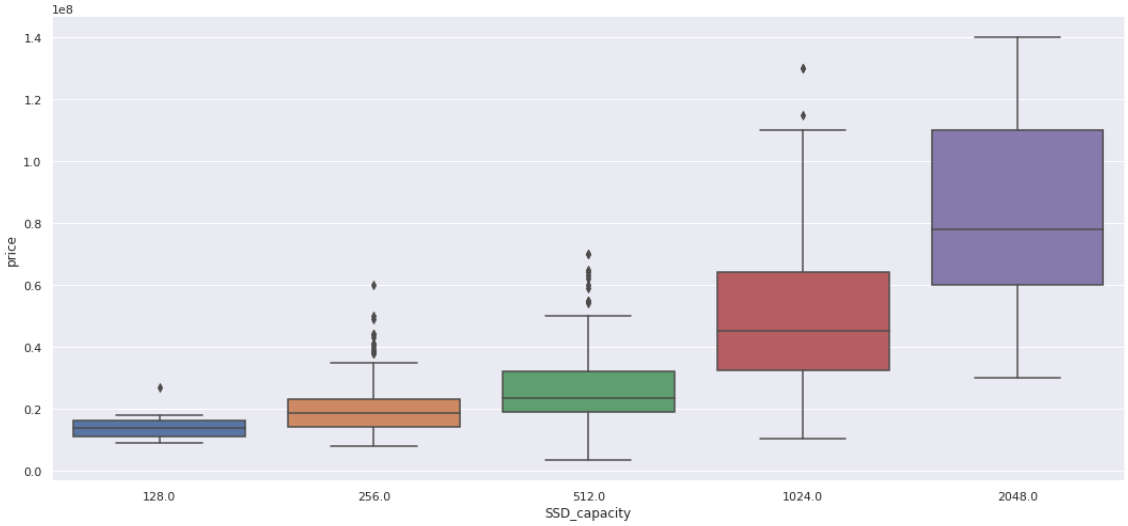

Figure 4: The candlestick plot of SSD capacity and laptop price

Another important feature that strongly influence the price of a laptop is the SSD capacity. From figure 9, we see that the higher the SSD capacity, the higher the laptop price. Markedly, laptops with 2048 MB of SSD are significantly more expensive than the other, with the highest price being 140 milion. However, the price distribution of laptop with large SSD capacity tends have higher variance. We think that this phenomenon is caused by the varied in price of different SSD brands.

2.4 Feature Engineering

Feature engineering is an essential step in the pre-processing of data for any machine learning model. It involves transforming and creating new features from the raw data to improve the performance of the model. In this research, the following steps have been performed to engineer the features of the dataset:

1. **Handling of missing values**: After cleaning the data, 434 rows contained missing values in 12 features. To handle the missing values, the `ram_type` feature was replaced by the most frequent ram type, which accounted for 76% of all laptops. This step helped to ensure that the data was clean and free of errors. Two columns were added to specify if a row did not contain values for GPU score and CPU score. For the remaining missing numerical values, KNNImputer was used, where all numerical values were taken into consideration. The missing values were imputed using the k-Nearest Neighbors approach, where a Euclidean distance was used to find the nearest neighbors. This step helped to ensure that the data was complete and could be used for further analysis.

2. **Outlier removal**: Outliers are data points that lie significantly away from the majority of data points. These points can affect the performance of the machine learning model, and thus it is essential to remove them. In this research, outliers were detected using IQR (interquartile range). Once detected, the outliers were removed from the dataset, which helped in improving the performance of the machine learning model.

3. **Transformation of unused values**: Some values in the dataset may not be used or have a very low frequency. In such cases, these values can be transformed into the more frequent ones to reduce the number of distinct values and simplify the dataset. This helps in reducing the complexity of the dataset and makes it easier for the machine learning model to learn and make predictions.

4. **One-hot encoding**: Categorical features are features that contain non-numerical values. These features need to be transformed into numerical values to be used in machine learning models. In this research, one-hot encoding was used to transform the categorical features into numerical ones. One-hot encoding creates a new feature for each possible value of the categorical feature, and each instance is then represented by a binary vector. This helps in transforming categorical features into numerical ones, making it easier for the machine learning model to learn and make predictions.

5. **Min-max scaling**: Normalization is an important step in feature engineering as it helps in bringing the data to a common scale. In this research, min-max scaling was used to normalize the data. Min-max scaling transforms the data such that all features lie in the range of [0, 1]. This helps in improving the performance of the machine learning model as well as making it easier for the model to learn and make predictions.

6. **Feature selection**: Feature selection is a process of identifying the most important features in the dataset. In this research, the importance value was calculated using random forest, and the features with an importance value greater than 0.0025 were selected for further analysis. This helped in reducing the complexity of the dataset and improving the performance of the machine learning model. By selecting the most important features, the model can learn and make

predictions more efficiently, reducing the chances of overfitting and improving the generalization ability of the model.

These steps helped in transforming the raw data into a clean, transformed, and normalized dataset, which was then used for further analysis and modeling. The feature engineering steps performed in this research helped in improving the performance of the machine learning model and reducing the complexity of the dataset.

2.5 Modeling

2.5.1 Decision Tree

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.The core algorithm for building decision trees called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction.

The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest standard deviation reduction (i.e., the most homogeneous branches).

Step 1: The standard deviation of the target is calculated.
Step 2: The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

$$SDR(T, X) = S(T) - S(T, X)$$

$T : Target$

$X : Attribute$

$SDR(T, X) : Standard\ Deviation\ Reduction$

$S(T) : Standard\ Deviation\ of\ the\ target$

$S(T, X) : Standard\ Deviation\ of\ the\ target\ on\ the\ attribute\ after\ splitting$

Step 3: The attribute with the largest standard deviation reduction is chosen for the decision node.
Step 4: The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed. In practice, we need some termination criteria.

2.5.2 Random Forest

Random Forest Regression is a supervised machine learning algorithm that is used for regression problems. It is a type of ensemble learning that combines multiple decision trees to make predictions. The idea behind Random Forest Regression is to create many decision trees on different subsets of the data and combine their predictions to produce a final prediction.

Random Forest Regression works by creating multiple decision trees on different subsets of the data, also known as bootstrapped samples. For each decision tree, a random subset of the features is selected to split each node in the tree. The final prediction is made by averaging the predictions of all the decision trees in the forest. This process is repeated many times to reduce the variance of the model and to improve its generalization performance.

The prediction made by a Random Forest Regression model can be expressed mathematically as follows:

$$f(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{1}$$

Where $f(x)$ is the prediction made by the Random Forest Regression model, T is the number of decision trees in the forest, and $f_t(x)$ is the prediction made by the t-th decision tree in the forest.

2.5.3 XGBoost

XGBoost (eXtreme Gradient Boosting) is an efficient and scalable machine learning algorithm that is widely used for various regression and classification problems. It is an ensemble learning method that combines multiple weak decision trees, called base learners, to create a strong and robust model. The algorithm is known for its accuracy, scalability, and computational efficiency, making it a popular choice for data scientists and machine learning practitioners.

In XGBoost, the base learners are trained in a stagewise manner, where each new tree is trained on the residuals of the previous trees. The residuals are calculated as the difference between the predicted values of the previous trees and the actual target values. By focusing on the residuals, the algorithm can correct its predictions and improve the overall accuracy of the model. The boosting process continues until the desired number of trees is reached or the performance of the model no longer improves.

XGBoost is a powerful and efficient algorithm for predictive modeling that can be applied to a wide range of problems, including regression and classification tasks. With its ability to handle large datasets and its ability to fine-tune its performance, XGBoost is a popular choice for data scientists and machine learning practitioners.

2.5.4 Neural Network

It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. ANNs are used widely in both regression and classification tasks, can handle large data, with a lots of variances for many specific problems like Computer Vision, Natural Language Processing,...

3 Experiments

3.1 Performance Measure

In order to evaluate the performance of the machine learning models used for predicting laptops' prices, we need to use appropriate performance measures. One commonly used performance measure for regression problems is the R2 score, also known as the coefficient of determination.

The R2 score measures the proportion of the variance in the target variable that is explained by the predictors. It ranges from 0 to 1, with a score of 1 indicating a perfect fit between the predictions and the true values, and a score of 0 indicating a poor fit between the predictions and the true values. In other words, the R2 score measures how well the model captures the variation in the target variable.

In a data set has n values marked $y_1, y_2, ..., y_n$ (collectively known as y_i or as a vector $\mathbf{y} = [y_1, y_2, ..., y_n]^T$, each associated with a fitted (or modeled, or predicted) value $f_1, f_2..., f_n$ (known as f_i, or sometimes $\hat{y}_i$, as a vector f). Define the residuals as $e_i = y_i - f_i$ (forming a vector e).

If $\bar{a}$ is the mean of the observed data: $\bar{a} = \frac{1}{n}\sum_{i=1}^{n} y_i$ then the variability of the data set can be measured with two sums of squares formulas:

- The sum of squares of residuals, also called the residual sum of squares:

$$SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i (e_i)^2$$

- The total sum of squares (proportional to the variance of the data):

$$SS_{tot} = \sum_i (y_i - \bar{y})^2$$

- The most general definition of the coefficient of determination is

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

We used the R2 score as the main performance measure to evaluate the accuracy of the predictions made by the machine learning models for predicting laptops' prices. The R2 score provides a single metric that summarizes the overall goodness-of-fit of the model. By comparing the R2 scores of different models, we can determine which model provides the best fit to the data, and therefore make the most accurate predictions of laptops' prices.

3.2 Training and Testing setup

In order to evaluate the performance of the machine learning models for predicting laptops' prices, it is important to split the data into two separate sets: a training set and a testing set. The training set is used to train the machine learning models, while the testing set is used to evaluate the performance of the trained models.

In our project, we split the data into a training set and a testing set, with the testing set accounting for 25% of the total data. This split is commonly used in machine learning, and provides a good balance between having enough data to train the models and enough data to evaluate their performance.

The data was randomly split into training and testing sets, ensuring that both sets are representative of the overall data distribution. This random split is important to ensure that the results of the machine learning models are not biased by any underlying patterns in the data.

As the baseline model, we chose linear regression. This model has been widely used in regression tasks and has proven to be effective in many cases. After training and testing the linear regression model on our laptop price prediction task, we obtained an R2 score of 0.634. This shows that there is still room for improvement and other more advanced models might perform better on this task.

3.3 Hyperparameters tuning

Hyperparameter tuning refers to the process of adjusting the hyperparameters of a machine learning model in order to optimize its performance. Hyperparameters are parameters that are set before training a model and cannot be learned from the data. They control various aspects of the model such as the learning rate, the number of hidden layers in a neural network, the number of trees in a random forest, etc.

The goal of hyperparameter tuning is to find the best combination of hyperparameters that result in the best performance for a given task. This is done through a process of trial and error, where different combinations of hyperparameters are tried and their performance is evaluated. The combination of hyperparameters that result in the highest performance are then selected as the final hyperparameters for the model. In our project, we apply two different tuning methods:

- **Grid Search Cross-Validation**: Grid Search CV is a brute-force method that tests all possible combinations of hyperparameters in a given grid and selects the best combination based on the performance metric. Cross-validation is used to evaluate the performance of each model and to prevent overfitting. Cross-validation involves dividing the data into training and validation sets, training the model on the training data and evaluating its performance on the validation data. This process is repeated multiple (k) times to get a more robust estimate of the model's performance. This method is simple and straightforward, but it has some limitations. For example, it may not find the optimal combination of hyperparameters if the grid is too small or if the optimal combination falls between the grid points.
- **Bayesian Optimization**: Bayesian optimization [6] is a probabilistic method that uses Bayesian inference and Gaussian processes to model the relationship between the hyperparameters and the performance metric. The method iteratively updates the probabilistic model and selects the next

set of hyperparameters to test based on the model's predictions. This results in a more efficient search of the hyperparameter space and a better chance of finding the optimal combination of hyperparameters.

3.3.1 Decision Tree

Decision Tree models can greatly benefit from hyperparameter tuning. By tuning these hyperparameters, the model can be made more robust to overfitting or underfitting, which can improve its accuracy on unseen data, obtain optimal results and make the most of the model's capabilities.

1. **Grid Search Cross-Validation** (k = 5): 0.680
2. **Bayesian optimization**: 0.695

3.3.2 Random Forest

Despite the poor result of Decision Tree model, Random Forest's model performance is very good. It's our the second best model, not very far behind XGBoost.

1. **Grid Search Cross-Validation** (k = 5): 0.830
2. **Bayesian optimization**: 0.823

3.3.3 Neural Network

The ANN model has many hyperparameters for tunning so it can get better result, we tunning it with leaning rate, optimizers, number of epochs, and the activation functions for the neurons. Due to the fact that our data is very small in both number of data points and number of attributes, the ANN model don't have good performance and it have to train with a large number of epochs to get a usable result.

1. **Grid Search Cross-Validation** (k = 5): 0.656

3.3.4 XGBoost

The XGBoost model is a powerful machine learning algorithm that can handle complex datasets and deliver accurate predictions. However, the model's performance depends heavily on the values of the hyperparameters. In order to obtain the best results, these hyperparameters need to be fine-tuned for the specific problem at hand.

1. The first step in the tuning process was to evaluate the performance of the XGBoost model using the **default hyperparameters**, gaining a R2 score of 0.798. This provided a baseline performance that could be used to measure the improvement of the model after tuning the hyperparameters.
2. **Grid Search Cross-Validation** (k = 5): 0.791
3. **Bayesian optimization**: 0.833

The results of our experiments showed that Bayesian optimization delivered far better results compared to the default hyperparameters and Grid Search CV. The model's performance improved significantly, achieving a higher R2 score and providing more accurate predictions.

3.4 Results

We evaluated the performance of four different machine learning models for the task of predicting laptops' prices. These models include Random Forest, Decision Tree, XGBoost, and Artificial Neural Network.

Model	R2 (Train Dataset)	R2 (Test Dataset)
Linear Regression (baseline)	0.729	0.634
Decision Tree	0.694	0.695
Random Forest	0.922	0.830
XGBoost	**0.963**	**0.833**
ANN	0.689	0.656

Table 2: Performance of our models

The results of our experiments showed that the XGBoost model performed the best with an R2 score of 0.833, followed closely by the Random Forest model with an R2 score of 0.830. On the other hand, the Decision Tree model had an R2 score of 0.695, while the Artificial Neural Network model had the R2 score of 0.656

It is worth mentioning that the XGBoost model was able to achieve the best results due to its ability to handle small datasets and its ability to perform well in non-linear regression problems. Moreover, the Bayesian optimization method was applied to tune the hyperparameters of the XGBoost model, which further improved its performance.

3.5 Model interpretation

Interpretability is an important aspect of machine learning models, particularly in applications where the results have a significant impact, such as predicting laptops' prices. One way to interpret the results of a machine learning model is by using SHAP (SHapley Additive exPlanations) values. As the XGBoost model with hyperparameters tuned by Bayesian optimization has the best performance among all the regression models, we will use this model to interpret our results with the help of SHAP.

SHAP (SHapley Additive exPlanations) is a model-agnostic, post-hoc method. It is a local interpretation method, but we could apply it to all the data to get a global understanding. It is based on the cooperative game theory which uses Shapley Additive Value (named in honour of Lloyd Shapley, 1951) to allocate the unequal contributions of all the players in a team. It belongs to the class of additive feature attribution methods. Additive feature attribution means a marginal contribution of each feature. It basically compares the differences with and without that player/feature. The SHAP values for a particular instance can be positive or negative, indicating whether the feature increased or decreased the prediction, respectively. Armed with this new approach we return to the task of interpreting our XGBoost model:

3.5.1 Local interpretability: explaining individual predictions

Force plots the most complete display of a single prediction. In Figure 5, 2 force plots explains the underlying contributions of each feature to the prediction for the median-priced house in the dataset.

The force plot structure emphasises the additive nature of positive and negative contributors, and how they build on the base value to yield the model's prediction, f(x). In the first example, we could clearly see that the high score for all categories regarding to the CPU, 1GB SSD, 20-thread CPU makes up for the high price. Whereas, a combination of low battery capacity, low score for GPU and CPU significantly decrease the price of the second laptop.

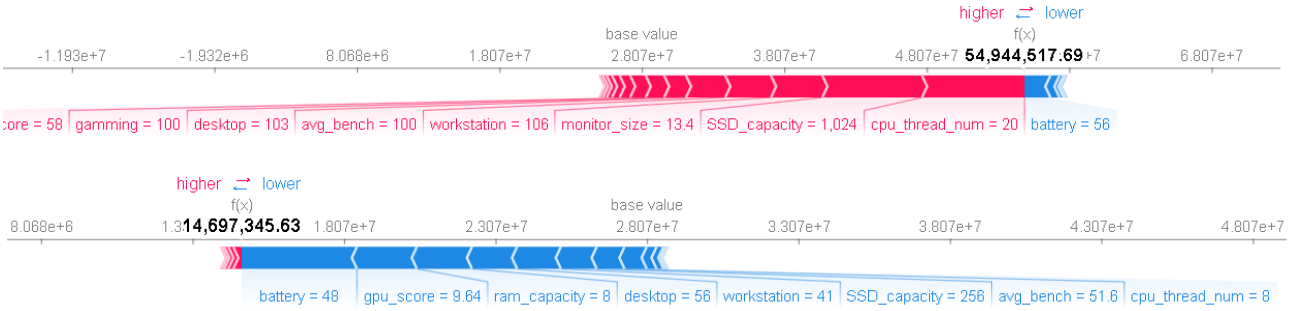

Figure 5: 2 force plots show the effects of features on the laptops' prices

3.5.2 Global interpretability: understanding drivers of predictions across the population

The goal of global interpretation methods is to describe the expected behaviour of a machine learning model with respect to the whole distribution of values for its input variables. With SHAP, this is achieved by aggregating the SHAP values for individual instances across the entire population.

Bar chart of mean importance

The simplest starting point for global interpretation with SHAP is to examine the mean absolute SHAP value for each feature across all of the data. This quantifies, on average, the magnitude (positive or negative) of each feature's contribution towards the predicted house prices. Features with higher mean absolute SHAP values are more influential. Mean absolute SHAP values are essentially a drop-in replacement for more traditional feature importance measures but have two key advantages:

- Mean absolute SHAP values are more theoretically rigorous, and relate to which features impact predictions most (which is usually what we're interested in). Conventional feature importances are measured in more abstract and algorithm-specific ways, and are determined by how much each feature improves the model's predictive performance.
- Mean absolute SHAP values have intuitive units - for this example, they are quantified in VNDs, like the target variable. Feature importances are often expressed in counterintuitive units based on complex concepts such as tree algorithm node impurities.

Figure 6 shows absolute Shapley values per feature across the data sorted in decreasing order. Suprisingly, the battery feature is actually the most important, followed by the `SSD_capacity` and `ram_capacity` feature. The features that we expect to have large contribution to the prediction of prices, i.e. `gpu_score` and `avg_bench` only come in sixth and elenventh place.

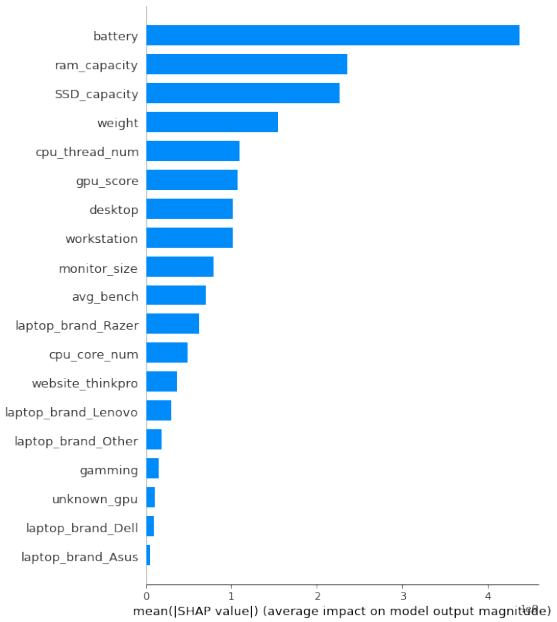

Figure 6: Mean SHAP value magnitude

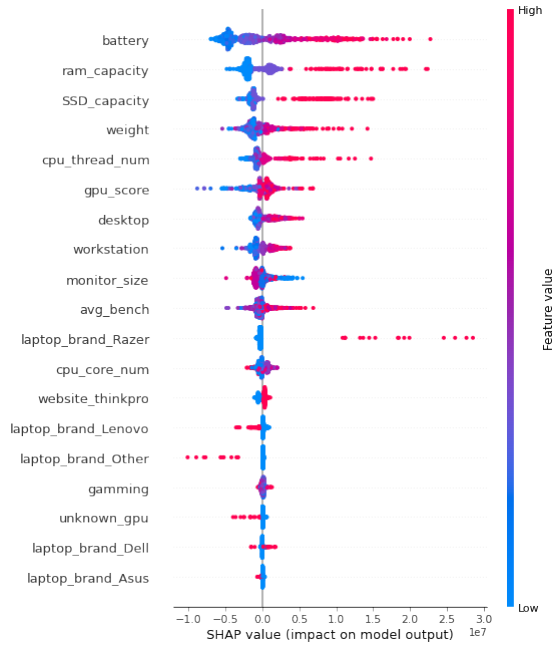

Figure 7: SHAP value impact on model output

Summary plot

The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The color represents the value of the feature from low to high. Overlapping points are piled up in y-axis direction, so we get a sense of the distribution of the Shapley values per feature. The features are ordered according to their importance. Whereas before the bar plot told us nothing about how the underlying values of each feature relate to the model's predictions, now we can examine these relationships. One interesting note is that the `gamming` feature, which specifies how good a laptop is for gamming, doesn't contribute much to the prediction of the XGBoost model

Figure 7 shows the beeswarm plot for our laptop prices prediction problem. It is interesting to note that the battery feature has more total model impact than the `gpu_score`, `avg_bench` features. While `SSD_capacity` is not the most important feature globally, it is by far the most important feature for a subset of customers. The coloring by feature value shows us patterns such as how large monitor decrease the price. If it is a Razer laptop have higher ram capacity, there is a high chance that's an expensive one. We can also see that if a laptop brand is "other", or in other words, not belongs to a popular brand, the price will decrease quite significantly

Dependence plot and Interaction plot

SHAP dependence plots show the effect of a single feature across the whole dataset. They plot a feature's value vs. the SHAP value of that feature across many samples. SHAP dependence plots are similar to partial dependence plots, but account for the interaction effects present in the features, and are only defined in regions of the input space supported by data. The vertical dispersion of SHAP values at a single feature value is driven by interaction effects, and another feature is chosen for coloring to highlight possible interactions.

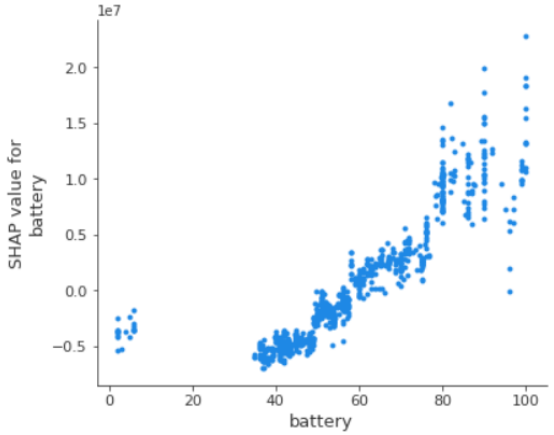

Figure 8: Dependence plot for battery's capacity

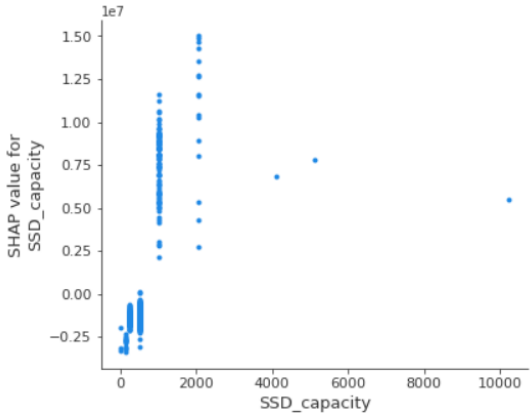

Figure 9: Dependence plot for SSD's capacity

Figure 8 and Figure 9 shows dependence plots for the top 2 feature: battery and SSD_capacity, respectively. Here we see the clear impact of battery's and SSD capacity on laptops' prices as captured by the XGBoost model. The raw variable value at which the distribution of SHAP values cross the y=0 line tells the threshold at which the model switches from predicting lower to higher laptop prices. For battery's capacity, this is approximately 60mWh, while the value for SSD's capacity is about 1GB.

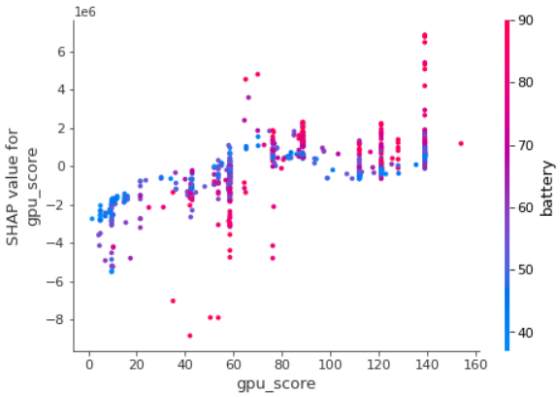

Figure 10: Interaction plot for GPU score coloured by battery capacity

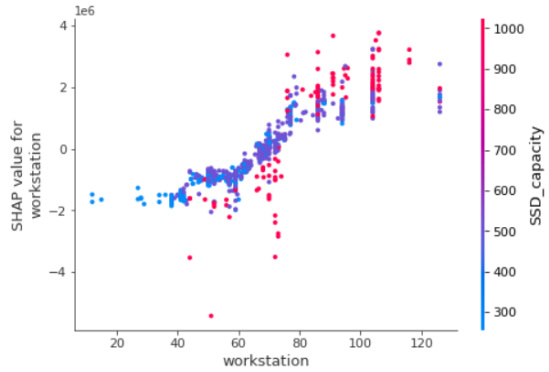

Figure 11: Interaction plot for workstation score coloured by SSD capacity

Figure 10 shows interaction plots for GPU score. We could a high value of battery lowers the effect of GPU score if the score is below 80, but raises it if it is above 80. Same effects could be witnessed in Figure 11, where high storage space of SSD lowers the effect of workstation score if the score is below 80, but raises it if it is above 80.

4 Conclusion and future work

In conclusion, this project aimed to predict the prices of laptops using various machine learning models. The data was crawled from two websites and preprocessed to handle missing values and convert the

features into a standard format. Exploratory Data Analysis (EDA) was performed to understand the relationship between different features and the target variable. Feature engineering was carried out to generate new features and improve the performance of the models. Four different machine learning models were trained and tested, including Random Forest, Decision Tree, XGBoost, and Artificial Neural Network.

The results showed that XGBoost outperformed other models, achieving a R2 score of 0.83. Hyperparameter tuning was carried out using Grid Search CV and Bayesian Optimization, and the results from the latter showed improved performance compared to the former. SHAP values were used to interpret the results of the XGBoost model, and the most important features were found to be battery, SSD storage capacity, and RAM storage capacity.

There are several directions that can be pursued to improve the performance of the model. Firstly, incorporating additional data sources, such as customer reviews or specifications from the manufacturers, may provide additional insight into the prices of laptops. Furthermore, we can explore other feature engineering techniques and incorporate more data to increase the size of the training set. It would also be interesting to compare the results with other state-of-the-art models, such as LightGBM, CatBoost to see if there is any improvement in performance. Additionally, we can also try to predict the prices of laptops from multiple countries to get a more comprehensive picture of the global laptop market.

References

[1] L. Breiman et al. “Classification and Regression Trees”. In: 1984.

[2] Leo Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[3] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[4] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[5] Scott M Lundberg et al. “Explainable machine-learning predictions for the prevention of hypoxaemia during surgery”. In: *Nature Biomedical Engineering* 2.10 (2018), p. 749.

[6] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.