Department of Information Engineering (DII)
M.Sc in Computer Engineering

Data Mining And Machine Learning Project: Laptop Price Prediction

Pietrangelo Manco

`https://github.com/PietrangeloManco/LaptopPricePrediction`

Academic Year 2023/2024

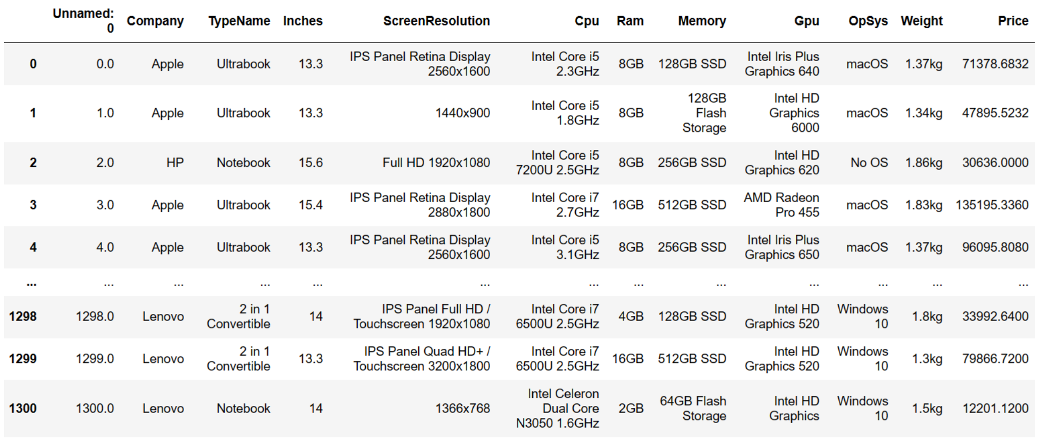
Contents

1 Introduction

The goal of the project is to build a Regression Model to predict the price of a given laptop configuration. The code was written in Anconda's Jupyter Notebook, using Python's scikit-learn library for most of the tasks. The work was carried out trying different methods for data pre processing, training and confrontation of the models, performance evaluation. The data used for the training of the models and their evaluation was a **`https://kaggle.com/`** dataset, originally scraped from a seller's website; details are discussed in the following section.

2 Dataset

The dataset used is available at the following **Link**. It's composed of uncleaned data, structured in 12 features and 1303 records. Each record represents a laptop model.



	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
0	0.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	71378.6832
1	1.0	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	47895.5232
2	2.0	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	30636.0000
3	3.0	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	135195.3360
4	4.0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	96095.8080
...	...	...	...	...	...	...	...	...	...	...	...	...
1298	1298.0	Lenovo	2 in 1 Convertible	14	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD	Intel HD Graphics 520	Windows 10	1.8kg	33992.6400
1299	1299.0	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD	Intel HD Graphics 520	Windows 10	1.3kg	79866.7200
1300	1300.0	Lenovo	Notebook	14	1366x768	Intel Celeron Dual Core N3050 1.6GHz	2GB	64GB Flash Storage	Intel HD Graphics	Windows 10	1.5kg	12201.1200

Figure 1: Overview of the Laptop Prices Dataset.

2.1 Features

There are 12 features, but one of them is an index that was immediately dropped. The remaining attributes were divided in 7 categorical and 4 numeric.

Categorical features are:

- Company
- TypeName
- ScreenResolution
- Cpu
- Memory
- Gpu
- OpSys

Numerical features are:

- Inches
- Ram
- Weight
- Price

Some of the listed categorical features were clearly more suitable for a numeric representation, like ScreenResolution and Memory, and the conversion was object of data pre processing. The Cpu attribute was splitted in CpuProducer, CpuModel and CpuClockSpeed, with the latter being numerical. Other features were subject of feature engineering, which will be discussed in details in the following sections.

3 Data Pre Processing

The first step of data pre processing consisted of spotting and managing eventual NULL values present in the dataset. There were 30 NULL values for each feature, corresponding to 30 NULL records, that were dropped.

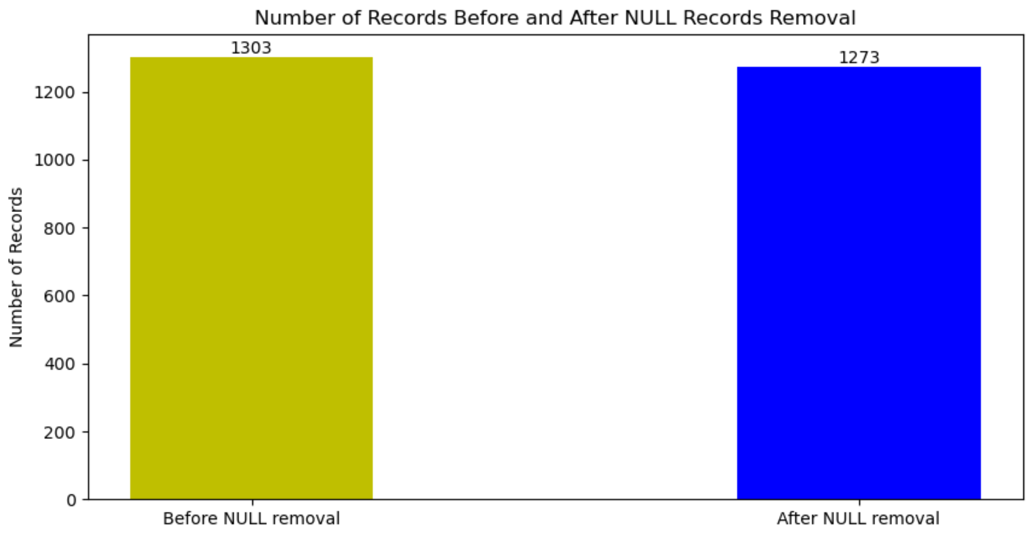


Figure 2: Data records before and after NULL removal.

3.1 Weight and Inches Features

Inspecting the 2 features, they both had a ’?’ value, whose records were dropped. The Weight feature had the unit of measure (kg) attached to each record, so it was removed. In addition to that, records whose weight was under a certain threshold (0.3 kg) were dropped, as they were most likely mistakes. Since these features were likely correlated, their correlation was computed to confirm that and they were joint in a new product attribute, called ’Weight_Inches’.

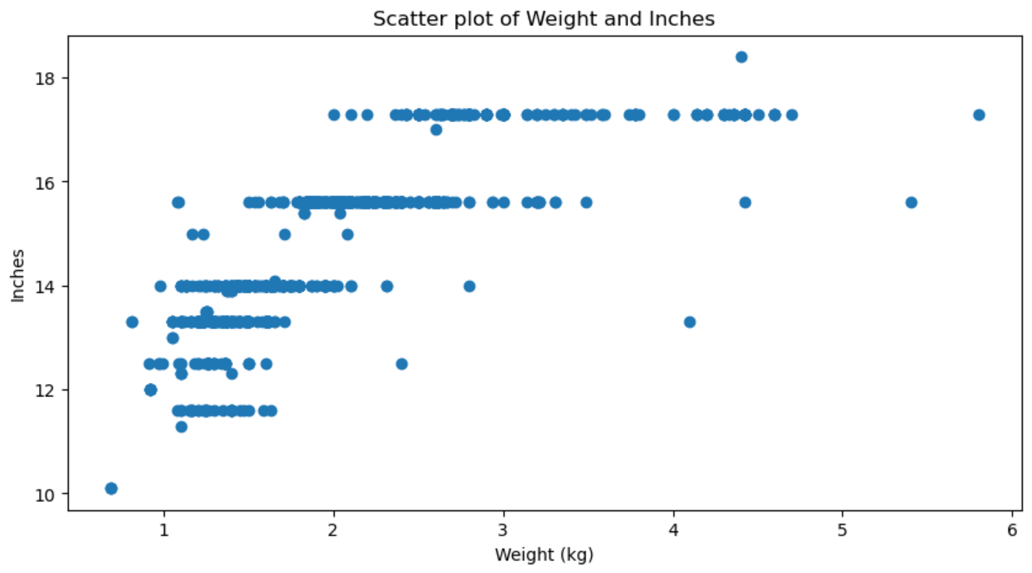


Figure 3: Scatter Plot of Weight and Inches attributes.

3.2 ScreenResolution Feature

The ScreenResolution feature had other information on screen technology attached to the classical 'a x b' value. To handle it, the value was extracted and it was converted in the actual product of the 2 numbers.

3.3 Cpu Feature

The Cpu feature was slightly more complex to address, as it's a categorical attribute composed of producer, model and clock speed, so it was split in that way. The CpuProducer was extracted as the first word of each value, and it only had 2 possible values: 'Intel' and 'AMD'. There was a 'Samsung' record, but it was dropped as Samsung isn't a Laptop Cpu producer, so it was probably a mistake. The CpuModel was more difficult to manage: 2 language processing functions were defined to standardize it as much as possible. In addition to that, CpuModel was also tricky to encode. The CpuClockSpeed, on the other hand, was easier to extract as the number before 'GHz'.

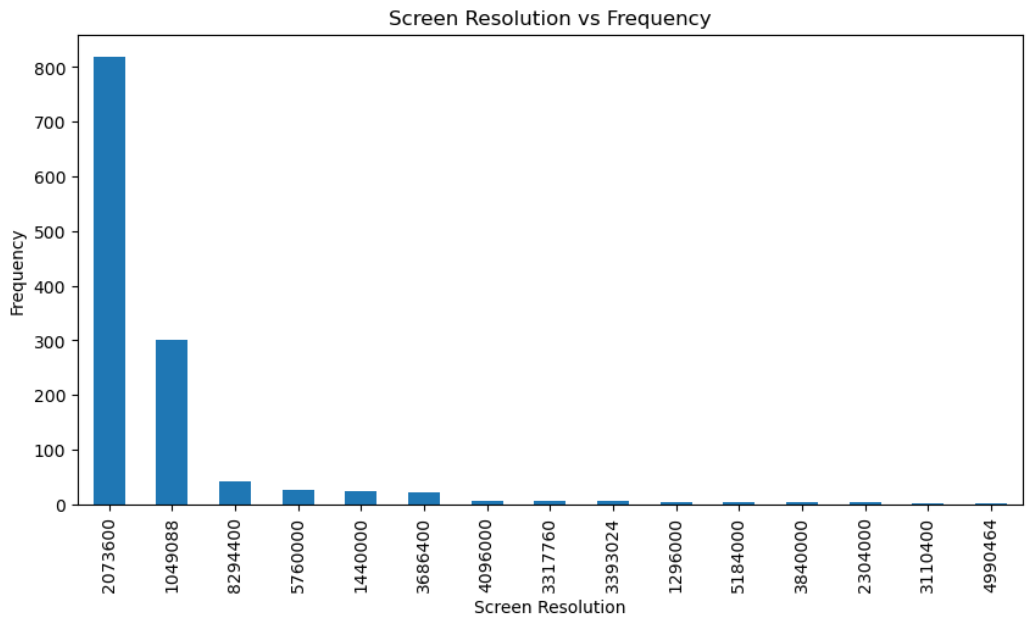
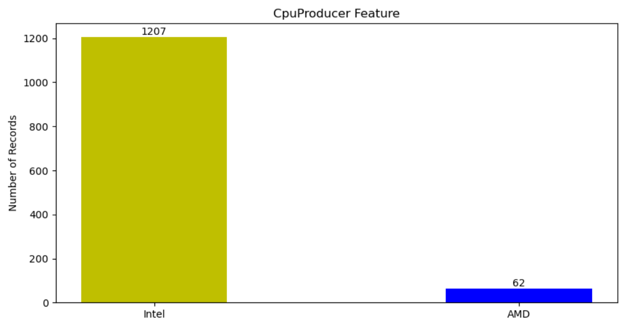


Figure 4: ScreenResolution instances after pre processing.

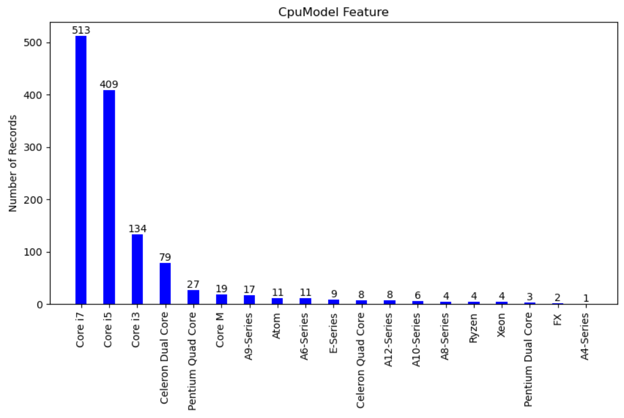
3.4 Ram and Memory Features

Ram feature was already clean enough, the only pre processing step performed was the unit of measure removal ('GB' string).

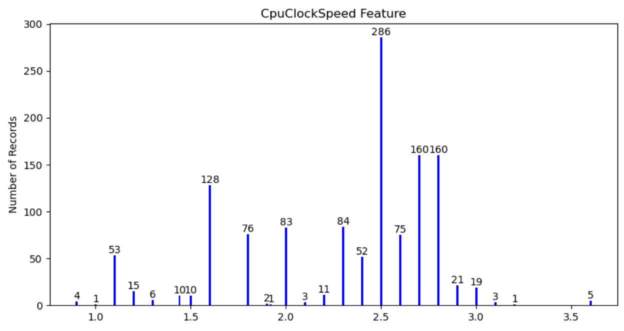
On the other hand, Memory feature was trickier: it was far less standardized, and it also had combinations of different types of memory (SSD, HDD). To manage it, 2 new attributes were created: SSD and HDD, and Natural Language Processing techniques were applied to extract such information from the initial feature, as well as removing units of measure. This division was important because SSD has definitely an higher impact of HDD on the target feature (Price).



(a)

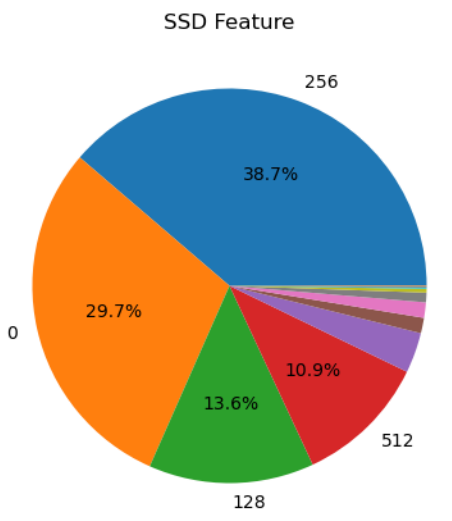


(b)

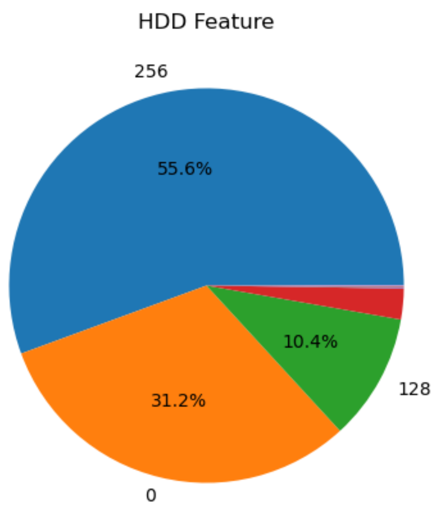


(c)

Figure 5: Cpu feature components after they were splitted and pre processed.



(a)



(b)

Figure 6: Pie charts for SSD and HDD features.

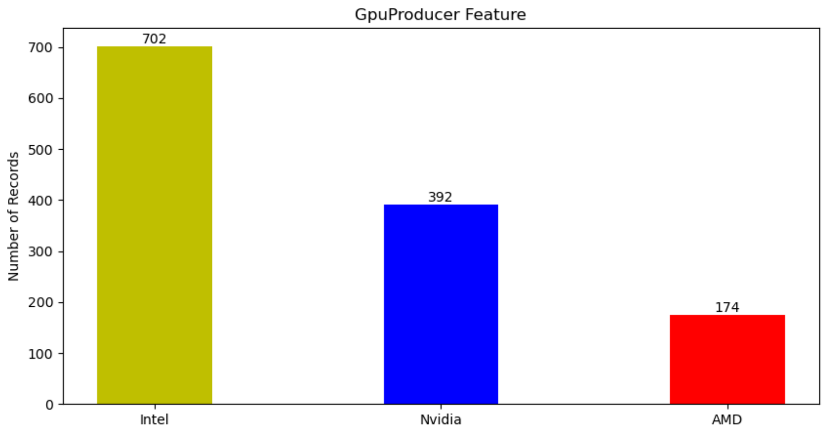
3.5 Gpu Feature

The Gpu Feature was managed in a similar way to the Cpu one: it was split in 2 different attributes, which are GpuProducer and GpuModel. NPL techniques were used again to extract such information, and some inconsistencies were manually solved.

Noticeably, GpuModel attribute had way too many value to establish a hierarchical order between them, as for CpuModel, so its encoding was even more problematic.

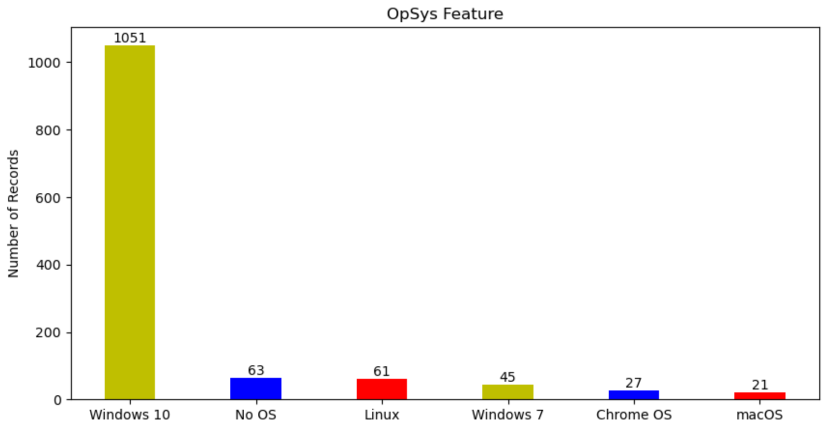
3.6 OpSys Feature

OpSys feature is about the operative system built in the laptop upon buying. There were some inconsistencies that were solved.



(a)

Figure 7: Histogram of GpuProducer feature frequencies.

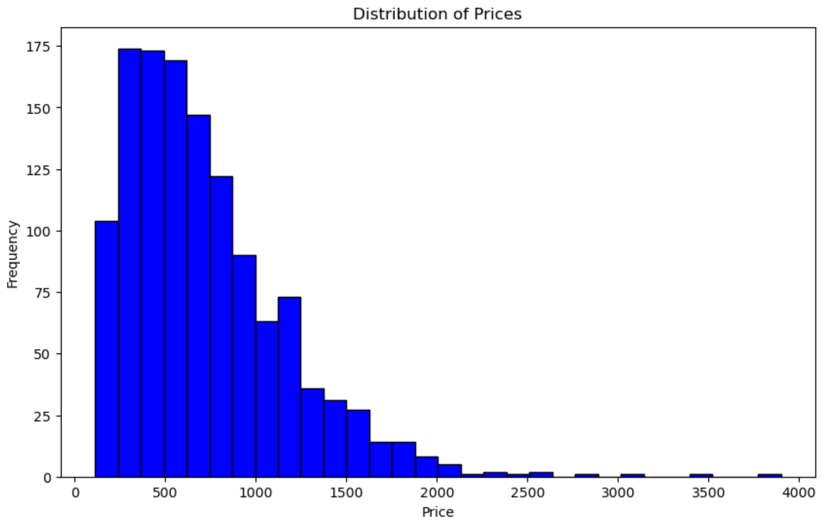


(a)

Figure 8: Histogram of OpSys feature frequencies.

3.7 Target Variable: Price

Price is the target variable of the regression task. It was initially expressed in Indian rupees, so it was converted to euros to better understand it. In addition to that, it was truncated to the second decimal. The feature distribution is right skewed, going from 111.25 to 3899.46, so it was later resampled and a logarithmic transformation was applied to it.



(a)

Figure 9: Initial distribution of the target variable.

4 Feature Encoding and Oversampling

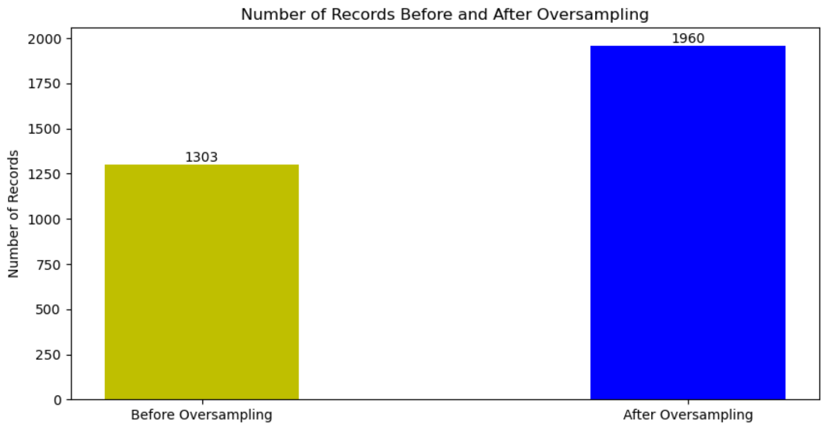
Since the task at hand is a Regression one, the features must be numerical to train the models. After the pre processing steps, there were 15 total features, 6 of which categorical: Company, TypeName, OpSys, CpuProducer, GpuProducer, CpuModel, GpuModel. The first 5 were encoded with a simple One Hot Encoding, since there wasn't a clear order of impact of the target variable within them. They were splitted in as many binary features as different values they could assume. On the other hand, Cpu and Gpu are key components in establishing a laptop's price, so they were handled differently.

- CpuModel: since there was a reasonable amount of different Cpu models in the dataset (18), their order was manually defined and the feature was encoded according to it.
- GpuModel: on the other hand, Gpu models were many more, so it wasn't feasible to order them all. Since Gpu is likely the component with the higher impact on any laptop configuration's price, it was encoded as the mean price of the records having the same Gpu. After the first operation, to prevent data leakage from the target variable, the attribute values were ordered and they were replaced with a score according to their position in the ordered list.

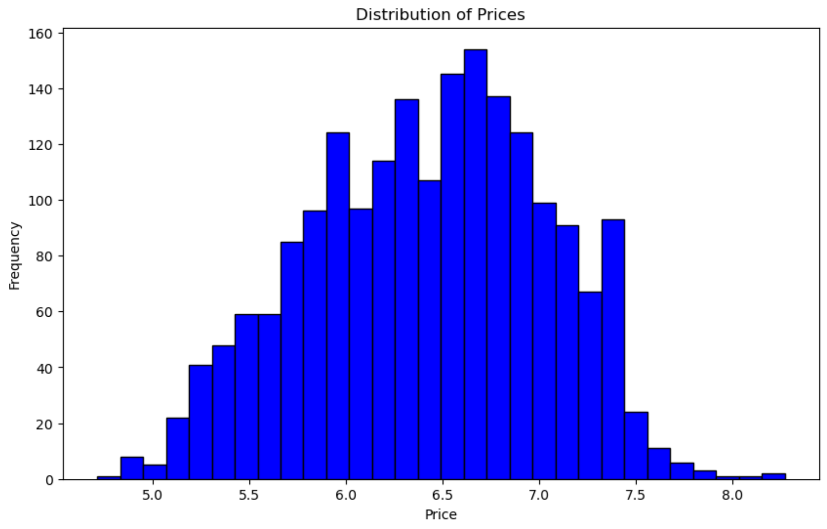
As for the target variable, since the distribution previously seen is skewed right, it was over sampled using a SMOTE technique adapted to regression tasks: the library used is called smogn, and it allows to introduce synthetic samples in the 'minority class', which in this case is the Price over a certain threshold, with Gaussian noise, to make them less correlated to the original ones. After the oversampling, a logarithmic transformation was applied to the target variable, to make the skewed nature of its distribution even less impactful in the training of the regression models.

5 Feature Selection

After the encoding, there were 44 features, so the feature selection technique was exploited to reduce them. It was implemented using the feature importance score of the RandomForestRegressor model in scikit-learn: after tuning its hyper parameters with a grid search, it was fitted on the dataset and used to compute feature importance. Features with an importance score under a threshold of 0.001 were pruned, and 25 features were kept. The RandomForestRegressor model was fitted only to extract the best number of features to use in the training of the models, but the method used for the actual feature selection was a Recursive Feature Elimination (RFE) that uses the tuned,



(a)

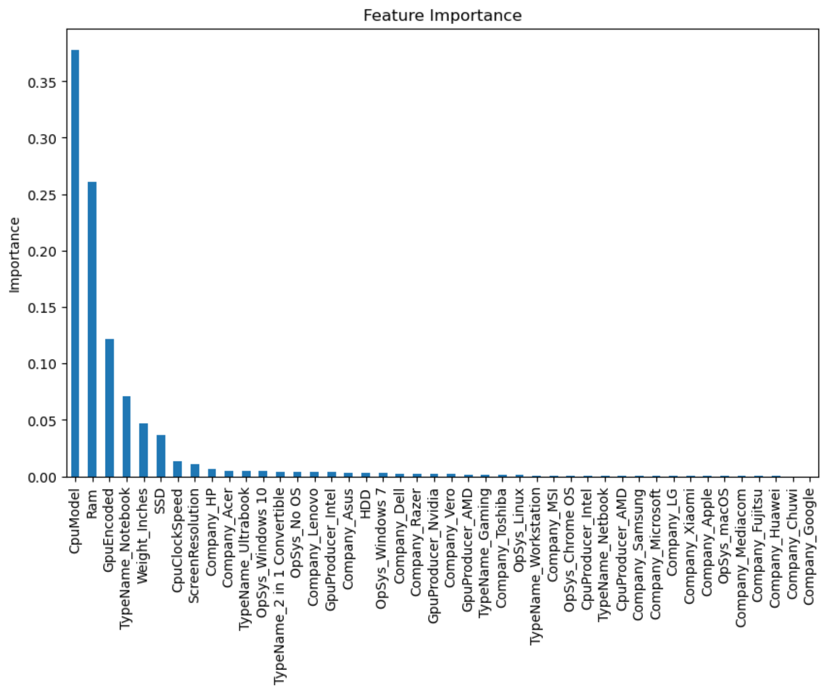


(b)

Figure 10: Oversampling and logarithmic transformation.

unfitted version of the model as estimator and the best number of features as a parameter.

As expected, the most important features are CpuModel, RAM, GpuModel, as they're usually the most expensive components of a laptop.



(a)

Figure 11: Feature importance scores.

6 Regression Models Training

Six different models were tuned, trained, evaluated and confronted. For each model but the Linear Regression, a grid search was performed to tune its hyper parameters. Input features were standardized for training using a Robust Scaler, which proved to perform better then other scaling method. Scaling function, feature selection model and regression model were put in a pipeline, to perform each operation with the right timing and avoid data leakage. The models and their hyper parameters are:

- Linear Regression: the most simple model, didn't need any hyper parameters tuning.
- Decision Tree: another simple model, the hyper parameters tuned were the maximum depth, the criterion to evaluate a split, and the minimum number of samples required to split a node.
- K-Nearest Neighbours: a lazy learning model, the hyper parameters tuned were the number of nearest neighbours, the weights and the metric to compute the distances.
- AdaBoost: an ensemble model that tries to give higher importance to errors in prediction, the hyper parameters tuned were the number of estimators, the learning rate and the loss function to use.
- RandomForest: another ensemble model that was also used for feature selection, the hyper parameters tuned were the number of estimators, the maximum depth and the minimum number of samples needed to split a node.
- GradientBoosting: ensemble method whose performance was the best among the models tried, the hyper parameters tuned were the number of estimators, the learning rate, the maximum depth and the maximum number of features to analyze to determine the best split.

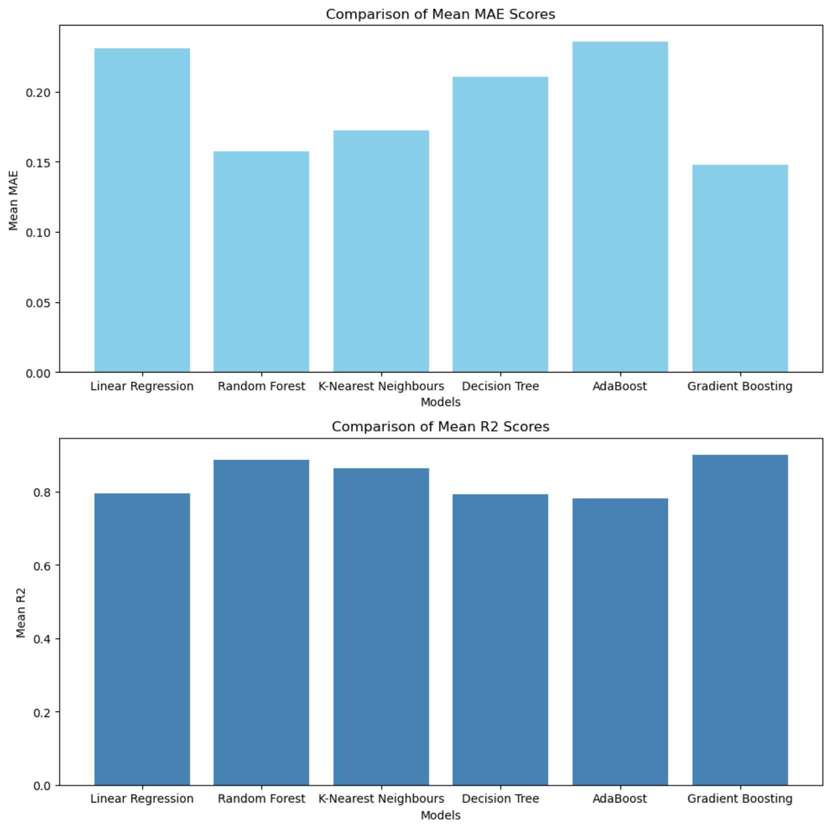
7 Models Evaluation

The metrics chosen for evaluation were Mean Absolute Error and R^2, and 2 custom functions were defined to compute them because the logarithmic transformation applied on the target variable had to be reversed first:

```
    def custom_mae(y_true, y_pred):
        y_pred_orig = np.exp(y_pred)
        y_true_orig = np.exp(y_true)
        return mean_absolute_error(y_true_orig, y_pred_orig)

    def custom_r2(y_true, y_pred):
        y_pred_orig = np.exp(y_pred)
        y_true_orig = np.exp(y_true)
        return r2_score(y_true_orig, y_pred_orig)
```

The evaluation was performed via a 10-fold cross validation for each model, and the final metrics are the average of the metrics from the 10 folds.



(a)

Figure 12: Evaluation metrics for the 6 models.

8 Models Comparison

Evaluation results showed that Random Forest, Gradient Boosting and K-Nearest Neighbours performed better than the other models. To establish if the difference is due to chance or it's actually relevant, the Wilcoxon Test was applied to the scores obtained in both metrics by the 3 models in each cross validation fold. Choosing a significance level of 0.05 for the p-value, the results of the test showed that Gradient Boosting is indeed better than the other 2 models in both metrics, while Random Forest is better than K-Nearest Neighbours according to Mean Absolute Error metric, but the difference seen in the R^2 metric between them is due to chance.

Table 1: Wilcoxon Test's results for the Mean Absolute Error between the 3 best models. The chosen significance value to reject the null hypothesis was p-value < 0.05.

Model	KNN	Random Forest	Gradient Boosting
KNN	-	0.027	0.006
Random Forest	0.027	-	0.006
Gradient Boosting	0.006	0.006	-

Table 2: Wilcoxon Test's results for the R^2 metric between the 3 best models. The chosen significance value to reject the null hypothesis was p-value < 0.05.

Model	KNN	Random Forest	Gradient Boosting
KNN	-	0.105	0.019
Random Forest	0.105	-	0.003
Gradient Boosting	0.019	0.003	-