A Random Forest and Gradient Boost Methods to Predict Camera Prices

Marcos García García, Carlos Gual Zaragoza

Radboud University, Netherlands

Abstract— **We have analyzed a camera dataset that contains more than a thousand cameras specs witch has been released in the last twenty years. The goal of this paper was to develop regression models capable to predict the price of a new camera. This information can be used, for example, to compare whether the price of a new camera is proportional to the performance it can achieve, or it's just this extra price the cost of the branding. To achieve it, two different algorithms have been trained: Random Forest and Gradient Boosting. Both models outperform similarly so an extra analysis was needed. The overall performance of this analysis is positive, giving consistent predictions with a tolerant error rate.**

***Keywords*— Camera, Price, Random Forest, Gradient Boosting, Data Mining, Regression.**

I. Introduction

In an increasingly globalised economy, competition between companies is inevitable. The price of a product, in most cases, is the main factor that makes a power client decline for your product. For this reason, the choice of a fair price according to the service a company offers is very relevant to ensure a good economic performance of the product.

Following this philosophy, we have decided to determine whether data mining is capable of helping with this problem. Developing a model that can output an accurate price for a given camera's specifications. This tool can be also implemented in different market niches like real estate or car production.

At the same time, this approach has also implications in the customers. Having this kind of information can lead to a better decision when purchasing a new product and they have to choose in a bast market with hundreds of options.

II. Related Works

This project has been inspired by two different works we found online, both quite similar to our project: the possibility of developing a model capable of predicting certain values given an appropriate dataset.

The first project was developed using a dataset uploaded by the user Ogres Leka in 2019 and titled: "Used cars database"[1]. The objective of this project was to predict the price of a second-hand car [2]. Although the implementation of this project is not finished, we got an overall idea of how to work with data.

Ultimately we found a second project titled: Predicting House Price, developed in 2018. Using a collection of houses containing characteristics such us: area in a square meter, price, conditions represented as a number from 1 to 10, etc he was successfully capable of training a regression model to predict the price of the house[3].

With those two examples, we decided that the idea of creating a model caple to predict the price of a camera by using an adequate dataset was feasible.

III. Dataset and Preprocessing Methods

The dataset used in this project is from the web page where we found the related works mentioned in the section above: Kaggle [4]. It's a huge database with thousands of datasets of all kind of topics. It also has a very big community where people around the world implements and share their own approaches using the available data.

We have used a dataset called: *1000 Cameras Dataset*, by Chris Crawford [5]. It contains around one thousand rows representing each camera and thirteen attributes one per column such as Model-Brand, Released date, Resolution, Price, etc.

To begin working with the data, first we had to analyze it. The distribution the price was very even in most of the dataset, although some value was out of the average they can not be considered outliers because the range of the price of the cameras can vary a lot depending on their features and the amount of professional cameras released on the market is lower than semi-professional or lower-budget and this is reflected in the dataset.

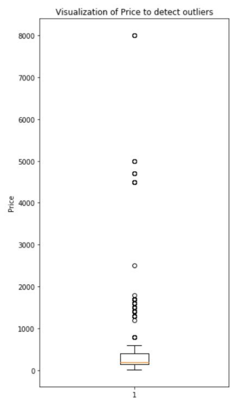

Fig. 1 Distribution of the price on the vertical axis

Figure 1 is an illustration of the distribution of the price along with the whole dataset. From it, we have extracted that above 774 dollars, maximum price inside the box, they are only 144 cameras out of 1036 in the complete dataset.

Another import factor that could influence the price is the brand of the product. To find out if this statement is correct, we plotted a whole correlation matrix in which the correlation is computed for all the attributes. First, we had to convert the Model, formed by the actual model of the camera and the brand, into a numeric code, one for each brand. In total there are 12 different brands like Canon, Casio, Contax, Epson, Fujifilm, etc. Once the brands have been converted into numbers the correlation matrix can be computed. Figure 2 shows the correlation between brand and price is so low with an actual value of 0.022 out of 1.

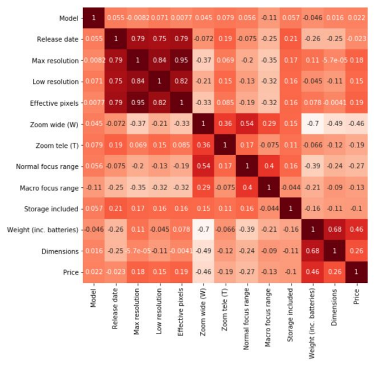

Fig. 2 Correction Matrix along all attributes

We have determined that the low correlation is produced because the majority of the brands analysed in this dataset aim to a big range of budgets. This affirmation can be shown in Figure 3.

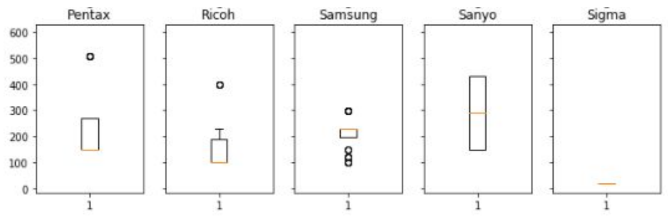

Fig. 3 Distribution of the price in five brands

To begin working in the regression models, a few preprocessing steps were needed. Fist the dataset was shuffled randomly. As we said before the data contained some extreme prices that had to be spread out randomly. In the last step, the dataset was split into two subsets randomly. One consisting of 75% of the original set to train the model. And the second one, the resting 25% of the data for testing purposes.

The idea behind this process was that in order to test if the trained moles perform properly it is necessary to give them as input new data. Otherwise, we could never know if the model has just memorized the data or it has leaned to generalize.

IV. Regression methods

This study proposes tree-based ensemble methods to predict the price of cameras by considering all relevant variables derived from the historical camera market. The ensemble-based algorithms consist of multiple base models (such as decision trees, neural networks), and each base model provides an alternative solution to the problem, whose predictions are averaged to produce the final output of the model. This approach often produces a more stable and accurate prediction than one produced by any of the individual base models included in the ensemble, because combining the outputs of each model reduces the total error of the prediction.

A. Decision Tree

Decision trees are the building block for the regression algorithms implemented for this project. They can be seen as a series of yes/or no question asked about our data eventually leading to a prediction class or in this case a continuous value in the use for regression[6].

The internal structure of a decision tree each internal node represents a "test" or decision on an attribute of the data set. In or case, each node is one of the columns of the data, e.i: Model, Resolution, Dimension, etc. The bach connects the nodes, and they represent the decision, normally we are referring to binary nodes in which each one has two branches, therefore two decision. The last nodes are named leaf node, they output the class label after computing all the decisions of the tree.

Although decision trees have great advantages like they are easy to implement and understand, It is important to be aware of their weakness. Overfitting may occur when the tree is too deep and the model begins to memorize the data instead of generalizing. Another important disadvantage is the low prediction accuracy for a given dataset. The next sections will explain in more detail how tree-based algorithms are implemented and try to solve the disadvantages related to single decision trees.

B. Random Forest

A Random Forest is a collection of decision trees. This implementation combines the simplicity of decision trees with the flexibility resulting in a vast improvement in accuracy. The essential idea is to average many noises but approximately models, and hence reducing the variance of the prediction. "*A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models*" [7].

The low correlation between the models is the key. To achieve this, Random Forest begins creating n trees or estimator, for each tree a random sample from the training data with replacement, so the same camera could appear more than once. This process is known as bagging.

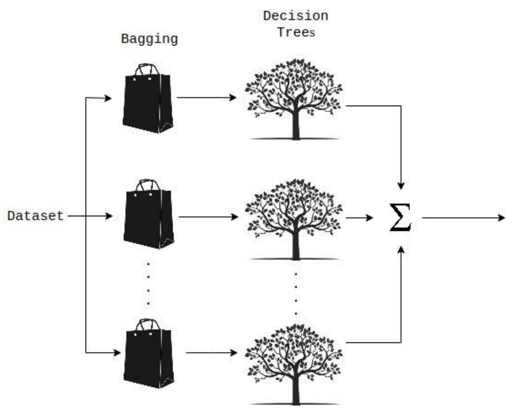

Fig. 4 Illustration of Random Forest bagging the train data, feeding it into the individual decision trees and averaging the prediction of the "forest".

C. Gradient Boosting

The second regression algorithm implemented in the project is Gradient Boosting. In this approach, the predictions are made sequentially instead of independently as shown in the previous with Random Forest. This technique employs the logic in with the subsequent predictions, or models, lear for the mistakes of the previous predictor. In the boosting process, examples that are difficult to estimate using the previous base models appear more often in the training data than the ones that are correctly estimated. Each additional base model is aimed to correct the mistakes made by its previous base models. “Therefore, the observations have an unequal probability of appearing in subsequent models and ones with the highest error appear most” [8].

The goal of Gradient Boosting, as any supervised algorithm, is to define a Loss Function and minimize it. It works as input along with the training set. In our project, this function is the Least Squares Regression (LS). We also tried different versions, such as the Least Absolute Deviation (LAD), Huber, or Quantile but all of them with pour prediction rate. As mentioned before in each iteration the new tree learns the errors from the previous. This value is denoted as residual, and it is computed subtracting the predicted value with the actual value.

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

Fig. 5 Initial constant for the model $F_0(x)$ where $L(y_i, \gamma)$ is the Loss Function

Gradient Boosting begins initializing the model with a constant value. Following the formula shown in Figure 5, we can compute the constant by the summations of each Loss Function over each observed value (in our project refers to the price of each camera contained in the training set), looking for the predicted value that minimizes the sum resulting in a more accurate model. $F_0(x)$ results in the average price of the whole training set. Once we have the initial prediction of the model we can compute the residual by subtracting the actual value with the prediction $F_0(x)$.

The second stem of the algorithm is a lop. It will repeat the following statements *M* times, passed as a hyperparameter, and represents the number of trees. So for each tree:

A. Compute the residual for each sample by subtracting the actual value the previous prediction.
B. Fit a regression tree that predict the residual of each sample.
C. The output of each leave in the new tree is determined by the Loss Function
D. New prediction for each sample.

V. Results

From the previous explanation of the algorithms implemented in the project, we extracted that selecting the correct parameters will have a big impact on the performance of the prediction. In the project, we used GridSearchCV so get the optimal values of those parameters. In the traditional way, in order to best score of your model, it is necessary to compare the for different value of the parameters the score of the regression model, plot them as a function over each iteration as analyse it.

Beginning with the implementation of Random Forest, the main parameters that could produce a significant impact on the score are the number of trees in the forest denoted as ‘n_estimator’ and the maximum depth of each tree as ‘max_depth’. With the intention of getting the optimal values, we used the library mentioned above. We gave as inputs an array of [100,200,300] for ‘n_estimator’, one of [25,50,100,125] for ‘max_depth’, and use 5 cross-validation. After fitting the GridSearchCV we got the optimal parameters are 200 trees with a maximum depth of 50. By fitting the Random Tree with those values we obtained an R^2 error of 0.95 for the training data and an R^2 error of 0.73 for the testing.

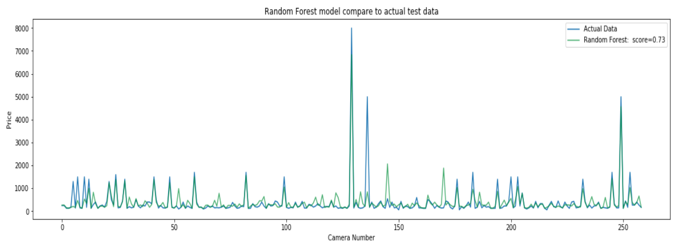

Fig. 6 Plot of the actual price compare to the prediction of Random Forest model.

For the last algorithm, Gradient Boosting was implemented following a similar approach. First, we detected that the main parameters are the number of trees ‘n_estimator’, the maximum depth of each tree ‘max_depth’, and in contrast with Random Forest, the learning rate ‘learning_rate’. To obtained the most suitable values we fitted the GridSearchCV with: 'n_estimators': [300,400,500], 'learning_rate': [0.1, 0.01], 'max_depth': [2,3,4], and cross-validation of 5 folds.

For the Gradient Boosting parameters, we obtained 'n_estimators' of 500, 'learning_rate' of 0.1, 'max_depth' of 3. By fitting the model with those parameter results in an R^2 error of 0.97 for training and an R^2 error of 0.75 for testing. Gradient Boosting outperforms slightly better than Random Forest as Figueres 7 shows.

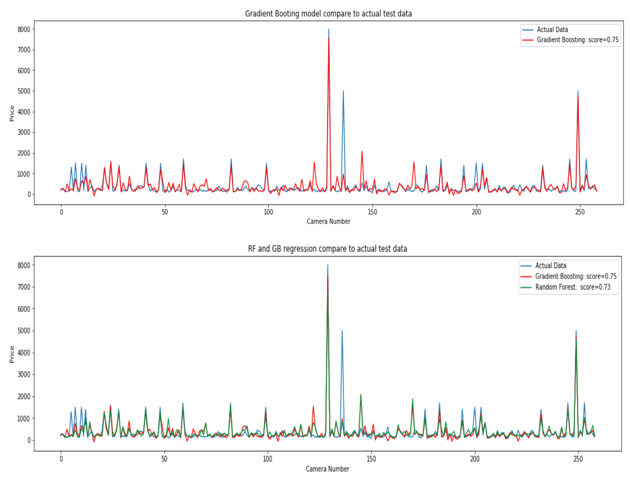

Fig. 7 On the top plot, the actual price is compared to the predicted price from Gradient Boosting. The bottom plot shows the Random Forest and Gradient Boosting prediction over the actual data on blue.

VI. Conclusion And Improvements

Ensemble methods based in single decision trees, like Random Forest and Gradient Boosting, are capable to reduce the variance more than the use of a single tree through averaging, in the case of Random Forest, or growing trees sequentially by adjusting the weight of the training data to minimize certain loss function.

From the results section, the proposed Gradient Boosting overcomes the Random Forest with a 2%~3% prediction rate. Although, the models show to be so dependent on the variation of the data used for training altering the difference in the prediction score. Taking it into context, both models seems to work with sufficient accuracy.

To conclude this application, it can be taken as a proof of concept. In order to extrapolate the idea and the work exposed newer data needs to used due the cameras' release date varies between 1994 and 2007. This even before the smartphones conquer the digital camera market. Producing phone models should also take into account in order to create the new data set.

The use of different algorithms should also be considered. Especially Machine Learning algorithms could produce a significant improvement in the analysis of the market, finding new patterns that Random Forest or Gradient Boosting could have just ignored.

References

[1] https://www.kaggle.com/orgesleka/used-cars-database/kernels
[2] https://www.kaggle.com/ddmngml/predict-car-value
[3] https://www.kaggle.com/predicting-house-price
[4] https://www.kaggle.com/
[5] https://www.kaggle.com/crawford/1000-cameras-dataset
[6] https://towardsdatascience.com/decisionTrees
[7] https://towardsdatascience.com/RandomForest
[8] https://medium.com/Gradient-Boosting
[9] The Elements of Statistical Learning, Random Forest
https://why-boosting-method-is-sensitive-to-outliers