iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** V **Month of publication:** May 2021

DOI: https://doi.org/10.22214/ijraset.2021.34259

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

Price Prediction using Machine Learning Methods

Mr. John Kenny[1], Mr. Sagar Mandal[2], Mr. Rushabh Mehta[3], Mr. Hrushikesh Malpekar[4]
[1]*Assistant Professor,* [2, 3, 4]*Computer Engineering Department, Universal College of Engineering, Mumbai University, Mumbai, India*

Abstract: ***Exponential growth is seen in e-commerce due to Covid-19. It has become extremely difficult for e-merchants to set an ideal price point for all products on the platform. Increased demand leads to higher prices and even many sellers on these portals keep prices a little high for better profit. Pricing as per how active the customer is on the interface, giving them discounts as per their purchases or/and rate of purchase. We built a price optimization system using algorithms such as SVR, Linear Regression (L-BFGS), Boosted Trees, Random-Forest LightGBM and at the end comparing results for which model has the least error. Predicting the optimal price point for every product effectively is the key to boosting the demand for the product. To get this optimal price point price prediction model has been used. The main intention of our model is to achieve an optimal price that would grow the demand for the product and be profitable to the retailer.***

Keywords: Price Prediction, Linear Regression (L-BFGS), Support Vector Machine, Boosted Trees, LightGBM, Random Forest Regressor.

I. INTRODUCTION

[1] The web Shopping (E-commerce) story began back four decades ago and is been rapidly growing since then steadily together with technologies and innovations. [2] The E-commerce industry is growing at an rapid rate. The pioneer of e-commerce was Amazon, a corporation that started selling books in the early 1990s. In comparison to traditional days, the retailer used to price the products manually having all the features of the product in mind. nowadays using the advancement of the web as retailers can track and analyze customers' private information, like their behavior and preferences.

Specifically, e-commerce is very sensitive towards pricing because its pricing strategy changes with a small impact of discounts. In fashion E-Commerce starting from product discovery, discounts, and the optimal price point play a very crucial role in driving the demand. The main objective of our model is to achieve an optimal price range with a sweet discount rate that would profit the user and increase the demand for the product that would effectively be profitable to the retailer. The model's features include all the Product details such as Product embedding rating, rank, engineered features a detailed brief could be seen in the data processing. The output label is the price value at which the product must be sold to increase the revenue, which is continuous. Hence it's a regression problem. For the Price prediction model, we have tried out several regression models like Linear Regression, Random Forest Regressor, XGBoost, SVR, LightGBM and performed various tuning parameters on every model to get the best out of every model. We have applied different solvers for the model to boost them efficiently. The Solvers are newton: Newton-Raphson, l-bfgs: limited memory BFGS, fista: accelerated gradient descent. A comprehensive explanation of the given models is discussed in the Model section. The significant next challenge is to predict the price for new products, it's a cold start problem. To overcome this problem, we've used a Price prediction model to learn the Product embedding rating, rank, Product description, Category, brands, etc. Succeeding in the Result and Analysis section using Quality metrics like Bias and Variance trade-off, RMSE, MSE we have assessed the best model among the given models.

II. RELATED WORKS

We've done some literature surveys during this section associated with price optimization. Firstly, during this paper [2], they proposed a machine learning and optimization technique within which to search out the optimal price at the individual product level, demand prediction model predicts the subsequent day demand for each product at a specific discount percentage, next to the worth elasticity of demand to urge the multiple demand values by varying the discount percentage. So multiple price demand pairs for every product are seen Typically, fashion e-commerce has immeasurable products, so there are many permutations. This setup is different from a traditional fashion e-commerce setup where we want to decide on a precise price point for all the products, whereas, within the former case, it is a range of price. Methodology stated in [2] for deriving the optimal price point. Typically, it states that as price increases, demand goes down, and vice-versa. But there is a contradiction in the case of Giffen/Veblen goods. Giffen/Veblen goods are a type of product, for which the demand increases when there is an increase in the price. E.g., The Branded Products is a type of Veblen good as its demand increases with an increase in price because it is used more as a status symbol.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

To deal with this the demand of each product is calculated based on the historical price-demand pairs for that product. Rather For new products, the elasticity of similar products determined by product embedding was used this is quite a hectic and unconventional way to deal with Cold-Start Problem. A simple and efficient way could have been emphasizing the importance of product description, rank, product-embedding rating, brand, sub-category i.e., the product details which we have proposed in the paper.

In paper [1], the authors proposed due to the increase in online shopping, sellers try to scam the majority of people for profit by hiking prices before or during sales and applying huge discounts on credit cards and other related coupons. Their system offers price prediction to the users for the next day so that the user or the customer can decide to buy the product at a given price. The user doesn't buy the products because the products are not worth the price or could find a better value elsewhere according to the user's perspective. At last, when demand sellers have to drop the price rather initially the seller could have significantly discounted the products. Our competitive pricing model, suggests that the seller the effective price to their product depending on their objectives and decisions.

The product description is usually represented in the form of a product model. [3] Consumers are more motivated to engage in all kinds of psychological activities related to the product, including purchase given that the quality of description provided is superior. Higher the quality of description, better the feelings towards the product. This concept amongst others motivated us to use the product description to complete our given task the authors also added that a great product description means a great product involvement i.e. how the user interacts with the product both emotionally and physically.

III. METHODOLOGY

A. Data Processing

To estimate the dynamic price for all products in the platform extensive feature engineering is done. The features are broadly categorized:

1) *Observed Features:* These features are behavioral features of Products they include brand, title, description, rank, also_view, similar_item, also_buy, category, etc. From these features, the model can learn the aspects of the products.
2) *Engineered Features:* To capture the trends of the products, we've performed data processing on relevant columns of the dataset. Firstly, the rank columns included data (e.g. 3,725,957 in Clothing, Shoes, Jewellery, etc.). We've extracted the Subcategory into different columns and rank into different Column. Further, we extracted the Subcategory and create segregated columns for each category. i.e. if the product is of category Clothing, Shoes, Jewellery. Then Category column Clothing, shoes, and jewelry will have 1 binary value rest category column will be assigned 0 binary value. We penalized and extracted the relevant and important words by performing TF-IDF on the description. Label Binarizer was an efficient way to deal with brand columns to encoded the categorical data. Count vectorized was performed on Title to effectively characterize the data. Normalizing the price played a crucial role. Initially, when we trained our model on unnormalized price value the model was persisting heavy loss. Since our price values were highly fluctuating and the model was not able to understand the relationship. So, we decided to normalized the price which significantly reduced the model loss. Removed the features with document frequency <=1. So only relevant features can be used for training the model.
3) *Rank:* The search ranking of a product is very highly correlated with the quantity sold. Products in the top search result have a greater tendency to getting sold first.
4) *Product Embedding:* The user-product interaction column. The value of an element in this column signifies the implicit score (rating) of a user's interaction with the product, which captures the hidden attributes of the products.

B. Price Prediction Model

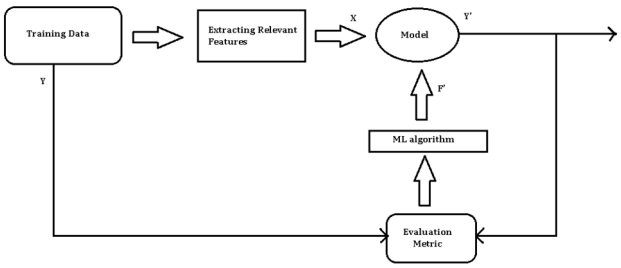


Figure 1: Architecture Diagram

©IJRASET: All Rights are Reserved

1) *Linear Regression using solver (lbfgs: limited memory BFGS):* Firstly, we analyzed the Linear Regressor. We prefer linear regression because it works on formulating a relationship with the given features and the target value. The processed dataset was split into 80-20 %. Only relevant columns were chosen as features so the model would fit the data efficiently. We selected a solver capable of training data with enormous features and fewer sample data. Limited memory BFGS (lbfgs) is a robust solver for wide datasets (i.e. datasets with many coefficients). We set L2_penalty to 2800 to avoid overfitting, which provides a prominent result. Max_iteration was set to 100 iterations further increasing the iterations has no relevant result. lbfgs_memory_level iterations were set to 100 to storage the requirement for every one of those gradients is that the num_coefficients within the problem. Lastly, a 5% validation set was set to evaluate the model while it's training. After the model was trained and evaluated on a 20% test set the RMSE was around 0.55. The below figure 2 describes the training progress.

Training Root-Mean-Square Error	Validation Root-Mean-Square Error
2.005578	1.988750
1.503848	1.596512
0.998482	1.206962
0.748798	1.025236
0.572335	0.958629
0.273750	0.897334
0.168272	0.872062
0.141226	0.732057
0.139813	0.575546
0.130325	0.563083
0.129463	0.561816
0.129511	0.561622
0.129524	0.561647
0.129525	0.561629
0.129525	0.561628

Figure 2: Training Progress of Linear Regressor

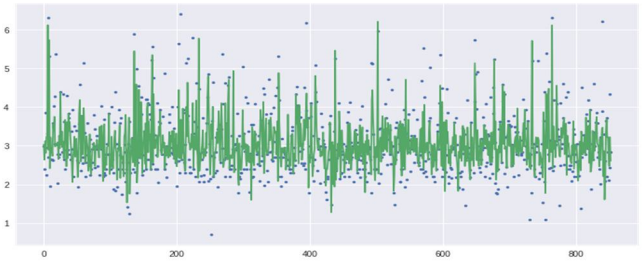


Figure 3: Prediction Scatter Plot of Linear Regressor

Has you could observe in above figure 3 the model is incorporating the region of the original price range very well.

2) *LightGBM(Light Gradient Boosting Machine):* Another Model was LightGBM, we picked this method due to its boosting frameworks such as GBDT, DART, and GOSS and on the opposite side, LightGBM splits the tree leaf-wise. We stacked all the Processed Data and wrapped it into one dataset, and transformed it into CSR to house a sparse matrix. Now 80-20% split was used. The Boosting Parameter we preferred was the traditional GBDT due to its better efficiency, lower memory usage, and vast features of learning capability. To avoid overfitting, we used the GridSearchcv feature to tune our hyperparameters like num_leaves, regularization parameter reg_alpha, reg_lambda to avoid overfitting. Eventually, after training and evaluating the model we reached a set of parameters ('learning rate': 0.01, 'num leaves': 800, 'max depth': 20, 'reg alpha': 0.55, 'reg lambda': 0.65). The num_boost_round was set to 3200 further increasing this parameter was persisting the same loss. To come up with max_depth and num_leaves we used the technique num_leaves =< 2^(max_depth). After evaluating the model on test data, the rmse appeared up to be 0.574. This was the best RMSE we could drive out of the LightGBM model.

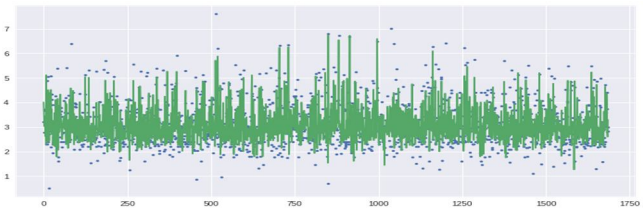


Figure 4: Prediction Scatter Plot of LightGBM

After observing the Scatter Plot, one could analyze the performance of LightGBM

©IJRASET: All Rights are Reserved

3) *Support Vector Regressor:* We've used a support vector machine for regression to solve our given problem, mainly due to its feature of tolerance, and capping the error also known as maximum error, or the epsilon. In the case of SVR, we split the dataset into different sections i.e. 1,50,100, 300, 500, and so on. We trained our model on every section and evaluate it on the validation dataset this way we were able to understand the learning Curve of the model. We used the GridSearchcv feature to tune our hyperparameters such as epsilon, regularization parameter C to avoid overfitting. Gamma was set to "scale" which uses 1 / (n_features * X.var()). The kernel we opted for was the radial basis function because it's relatively easy to calibrate. After training and evaluating the model on the test set we reached a set a parameter (C=40, cache_size=200, degree=3, epsilon=0.1, gamma='scale', kernel='rbf' ,max_iter=-1, shrinking=True, tol=0.001) and RSME was reached 0.545 which was best with respect to other models.

	Mean training scores		Mean validation scores
1	-0.000000	1	0.845899
50	0.009724	50	0.823069
100	0.011759	100	0.696207
300	0.029931	300	0.618795
500	0.033195	500	0.563548
1000	0.047903	1000	0.511695
2000	0.073539	2000	0.447007
3000	0.083834	3000	0.400866
4000	0.091936	4000	0.372872
5416	0.100802	5416	0.341022
dtype:	float64	dtype:	float64

Figure 5: Training Progress of SVR Model.

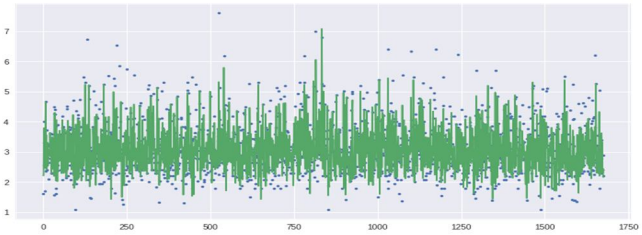


Figure 6: Prediction Scatter Plot of SVR

Observing the Scatter Plot the model seem a bit complex and performance of the model for seems effective.

4) *Boosted Decision Tree:* [4] We chose XGBoost, Due to its ensemble approach, bagging, and pruning techniques. The same 80-20 % approach was carried out on processed data. We picked features from the dataset that we thought could be fitted to increase the accuracy. Hyper-parameter min_child_weight and min_loss_reduction was set optimally to deal with overfitting and prune the unnecessary node. Setting the max_iterations = 100 was found effective in further enhancing the parameter the model was revolving around the convergence point. max_depth was assigned a higher range since we have an enormous number of features. After various training and evaluation, we came up with a set of hyper-parameters (min_loss_reduction=1.3) which subsequently seems efficient. The RMSE was stated around 0.565. The further figure shows the training progress.

Training Root-Mean-Square Error	Validation Root-Mean-Square Error
1.972795	1.997658
1.442229	1.483047
1.073337	1.137023
0.821246	0.918963
0.647216	0.779134
0.351564	0.597268
0.314721	0.589966
0.309675	0.589026
0.309663	0.589127
0.309663	0.589145
0.309663	0.589148
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149
0.309663	0.589149

Figure 7: Training Progress of Boost Tree

©IJRASET: All Rights are Reserved

The below Figure displays the Prediction Plot against the original price range. The model performance seems quite prominent.

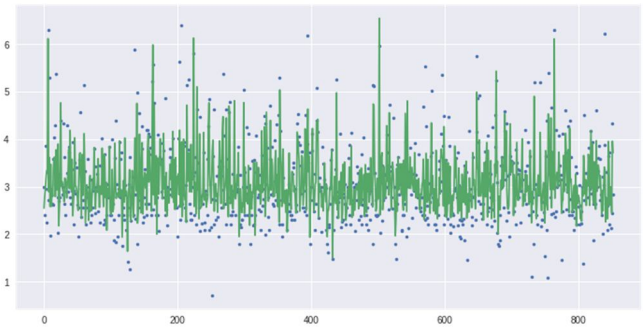


Figure 8: Prediction Scatter Plot of Boost Tree

5) *Random Forest Regression:* [4] The chief analysis to operate on Random forest was its outsized number of trees within the forest which ends up in high accuracy and prevents overfitting. An optimal number of features were fitted to the random-forest model since the model seems to be underfitting. Initially, we tweaked the min_loss_reduction parameter with a higher range to prune undesired nodes but the model was enduring high loss. So, we decided to lower the value up to 0.1. Max_depth was measured till 100, further gain was not seen in max_depth. The max_iterations were found prominent at 100 iterations. We tweaked the column subsample and row_subsample which is a bagging trick to deal with overfitting. After trying out different combinations of parameters we valued row_subsample to 0.5 and column subsample to 0.25. After tweaking all the parameters and evaluating the model on test data we could lower the RMSE to 0.60. Further is displayed the Prediction Plot of the model.

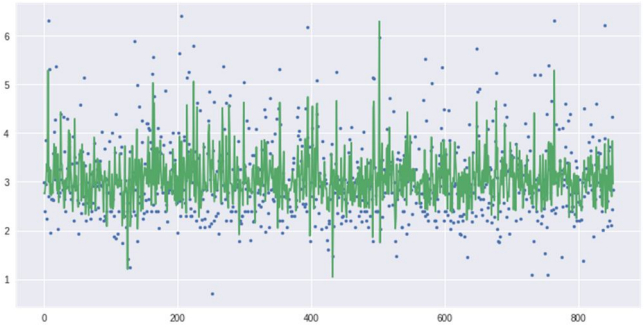


Figure 9: Prediction Scatter Plot of Random-Forest

IV. RESULT AND ANALYSIS

In this section, we have discussed the numerical evidence of why we have selected certain models among given options. Firstly, a brief description of our dataset, followed by the different evaluation metrics used to assess the accuracy, and last but not least models' comparison and selection.

A. Brief Description of the Dataset

We decided to figure on a dataset that's extracted or collected from an E-commerce website that has hands-on worked on the given modules. The Dataset we used is an updated version of the Amazon review dataset. This dataset includes reviews (ratings, text, helpful vote), product metadata (descriptions, category information, price, band, and image features), and links (also viewed/also bought graphs). [Metadata]: Product information, e.g. colour (white or black), size (large or small), package type (hardcover or electronics), etc. Product images are taken after the user received the merchandise. Bullet-point descriptions under the merchandise title. Technical details table (attribute-value pairs). Similar products table. [More categories]: Included 5 new product categories.

B. Evaluation Metric

While building machine learning models, we want to keep the error as low as possible. There are two major sources of errors which are bias and variance. We have tried to maintain an optimal trade-off point between bias and variance. [2] The base metric use to evaluate is mean absolute error & root mean squared error the base idea of this metric is to tell us how much-predicted output is deviating from the actual label.

©IJRASET: All Rights are Reserved

C. Model Comparison and Selection

In this section, we discussed the performance of the given models based on some factors:

1) *Bias Variance Tradeoff:* So, all the given models are trained on the same processed data and have been estimated on validation data. An 80 - 20 % approach was applied. The approach used was we split the training data into different segments e.g. 1: 100: 500 and so on. And trained the model on every segment and estimate it on validation data.

Figure 10: Learning Curve of SVR Model

Since our SVR model, Fits the training data very well, it means it has low bias with regard to that set of data. By analyzing the gap between the validation learning curve and the training learning curve we could mark that the gap is far-flung this symbolizes that our model had high variance this is because we have an enormous quantity of features which are causing a complex model trained and habitually complex model tend to have high variance. So, the validation curve still had some potential to converge towards the training curve. To enhance the bias slightly and minimize the variance, we tweaked the regularization parameter, subdued the number of features in training data, due to the diminished number of features, the model would be less complex. But there was still some gap as you could see in the SVR model diagram. Even after tweaking more parameters, the model tends to persist the same loss. This indicates that we had seized our optimal trade-off point where even after adding more features or tweaking parameters the model would rotate around the converging point.

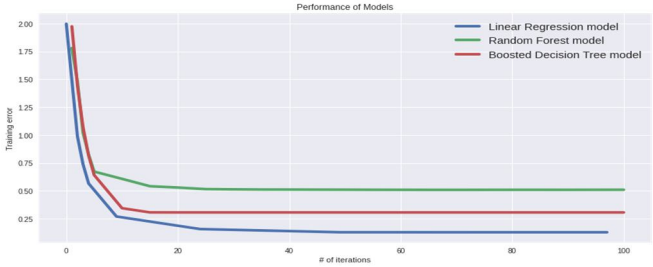


Figure 11: Training Error of our models

Examining the diagram of the training and validation error. The linear model fits the training set very well and so we could see a sense of overfitting here and the validation error seems to converge around 0.55 RMSE so we could conclusively say this is the identical case as the SVR model our model tends to have low bias and high variance. There is still a likelihood that the validation error could converge towards training error. So, to find the optimal trade-off point we have to slightly enhance the bias and decrease the variance So, we tweaked the regularization parameter, subdued the number of features. After performing this there was a significant effect in a validation error.

But there was nevertheless some gap as you could see in the diagram. We decided this as an optimal trade-off point even after tweaking and squeezing out the parameter the model perseveres the same loss.

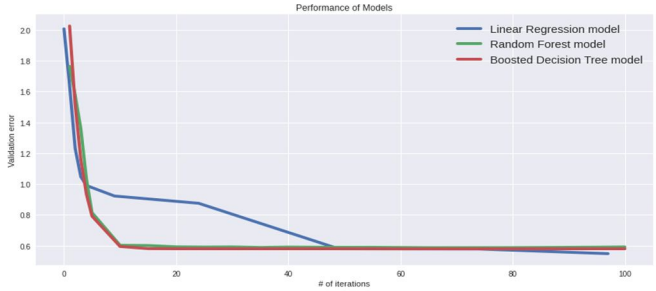


Figure 12: Validation Error of our models

©IJRASET: All Rights are Reserved

The Random Forest and the Boosted Decision Tree each have a significant training error, showing underfitting and bias. The validity error for Boosted Decision Tree is approximately 0.57, while Random-Forest has a validity error of 0.60. This indicates a contracting vacuum. In general, the larger the bias and the smaller the deviation, the narrower the difference. If a learning algorithm's variance is low, the algorithm will generate a simple and similar model. We discover that we have a problem with low variance. Reduce the regularisation of the current learning algorithm so that the model suits the training data well. Since we tweaked all of the model's parameters. The validation error was not convergent to the desired hash. It is prone to revolving around the converging point and producing the same loss.

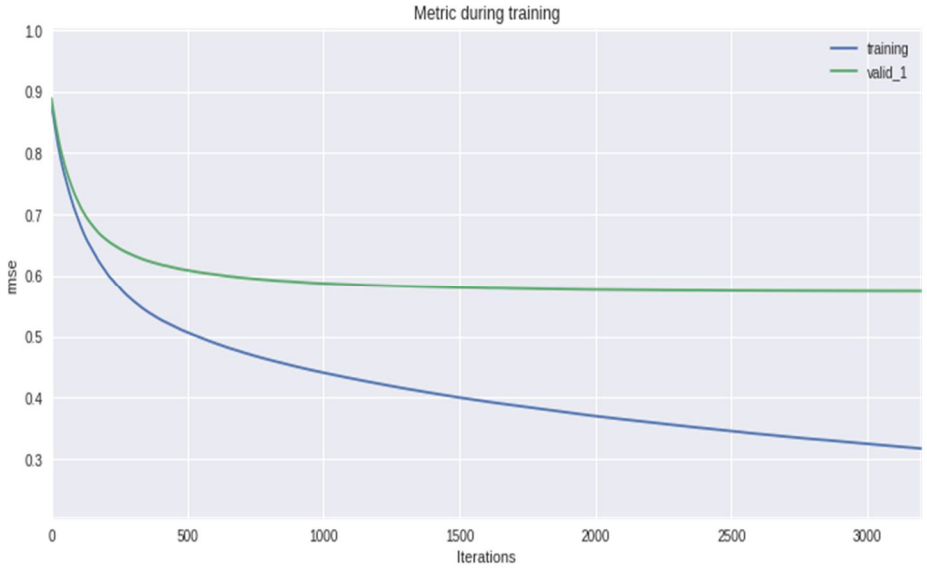


Figure 13: Learning Curve of LightGBM

In the case of LightGBM, a traditional Gradient Boosting Decision tree solver was used. The LightGBM model has high training error which implies underfitting and High Bias. The validation error resembles to be approximately 0.576 since there exists a small gap. This model also tends to show the same loss of high bias and low variance. To cope-up with the solution we examined several solvers of LightGBM but there was no notable change in performance. So, we reduced the regularization parameter so the model will fit training data efficiently this would boost up the variance and lower the bias. After tweaking we were able to minimize the bias significantly and boost the variance. There was still some gap between the validation and training error but we chose to stop because further tweaking the parameter had no vital change in the model. The model was enduring the somewhat same loss.

2) Model's Comparison Based on RMSE and Max-error

Models' performance metrics comparison

Model	Max-error	Rmse
Linear Regression	2.856	0.556
Random Forest Regressor	2.613	0.605
Boosted Decision Tree	2.578	0.565
SVR Regressor	NA	0.545
LightGBM	**NA**	**0.574**

Figure 14: Model's Performance Comparison

In the above model's performance statistics, one could perceive that all the model's RMSE vary very lightly. This is because we have operated on every model very specifically and tried to fit the trained data efficiently on each model. Even though some models have outperformed which includes the SVR model and Linear regressor.

©IJRASET: All Rights are Reserved

3) Computational Cost

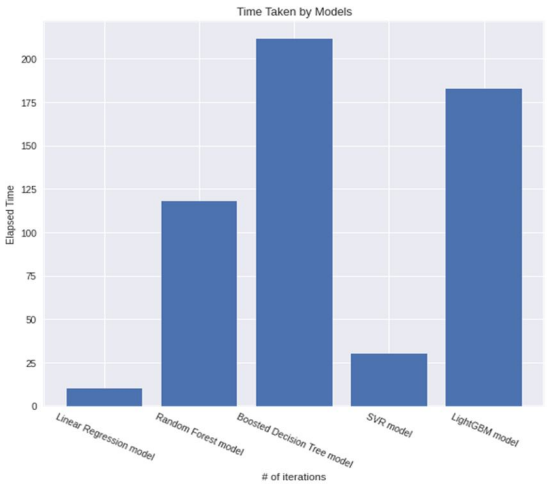


Figure 15: Computational Performance of various models

So, the linear regressor and SVR model have a very low computational cost which was foreseen because a relevant solver was utilized for both the models according to the dataset and model effectiveness. The boosted decision tree has the highest peak this could be because it, has to train distinct trees on a different subsample of the dataset and assign weight on each tree based on their influence on the accuracy of the model and lastly ensemble of this tree, fits us the result. The Random Forest model has a satisfactory amount of training cost which was presumed since we had an enormous number of features and the model has to evaluate the features and build a tree out of it. LightGBM is also very close to boosted decision tree this was anticipated since our model had a depth of 20, and the learning rate was finally low and num_boost_round was high. We worked to lower the training cost but limiting the parameter was affecting the accuracy of the model.

Ultimately after analyzing all the above factors, in terms of RMSE and max-error SVR model is lightly leading but a 1 % or 2 % variation is not a prominent deal. The linear regressor has a very low computational cost of around 10 sec which makes it very reliable and profitable.

Also, the Linear regressor has a very small gap between the validation and training error which facilitates a low bias and mediocre variance model. So, we concluded that we would continue with Linear Regressor with the solver as (l-bfgs: limited memory BFGS).

V. CONCLUSION

Any top competitor is now using competitive pricing. The biggest challenge posed is how to sell dynamically in such a way that it is profitable for the retailer. Price Prediction helps vendors in pricing their goods and services at the optimum discount price. We proposed to create a simple model that could analyze the product's price point. In this article, we stressed the importance of individual product metadata information that would help the model predict the demand and price of the product. Later our paper evaluated the accuracy of various models based on three main factors. First, the Bias-Variance trade-off point, then the RMSE and max-error, and finally the computational expense. Price prediction using a linear regressor and a solver with L-BFGS is thought to be optimal. In the future, we want to add the pricing history of goods purchased. Data from previous advertisements and campaign campaigns. Competitor pricing for comparable goods. This will assist our model in learning the demand and pricing versatility based on rivals' recent practices and previous marketing strategies.

REFERENCES

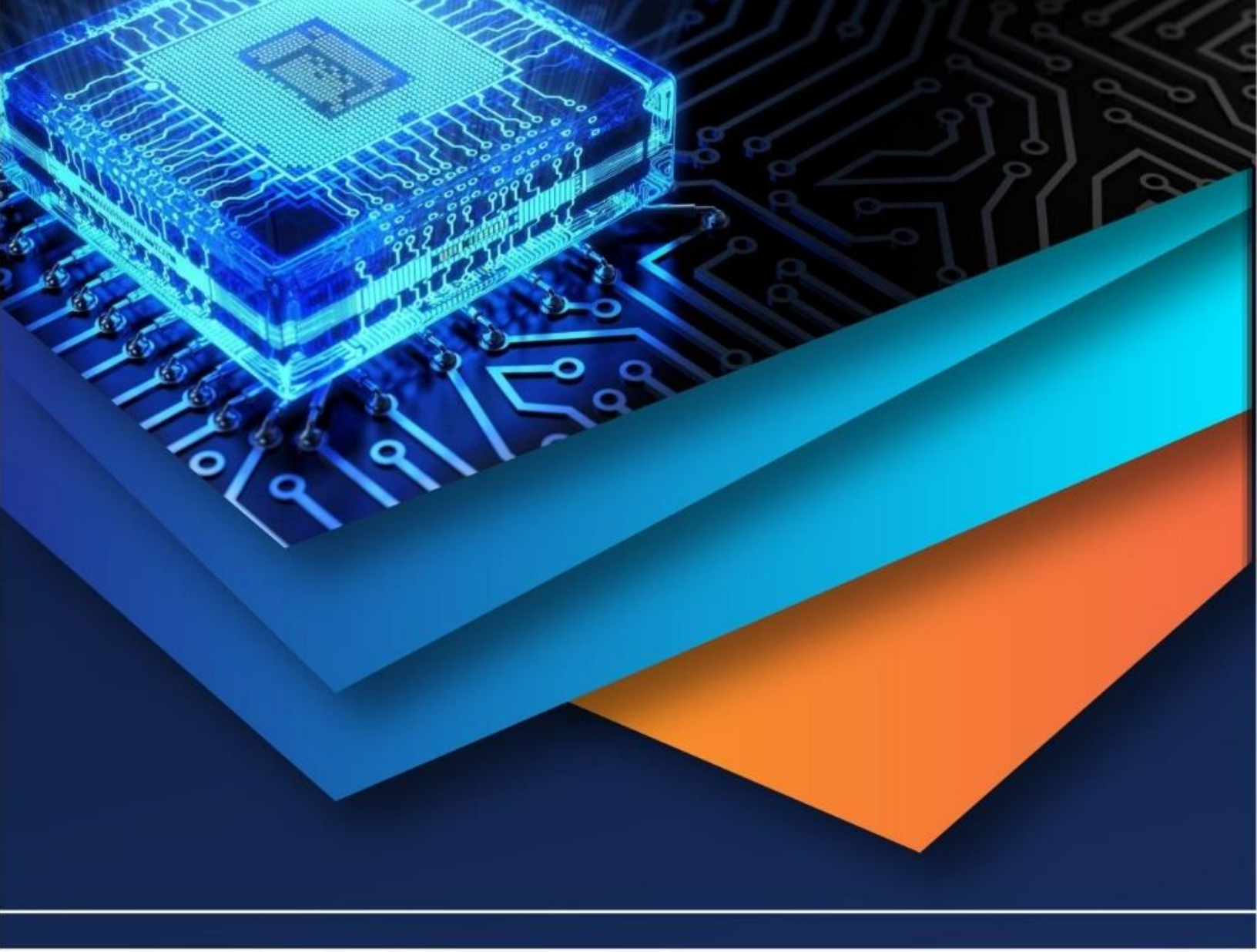
[1] M. Z. Shahrel, S. Mutalib and S. Abdul-Rahman, "PriceCop - Price Monitor and Prediction Using Linear Regression and LSVM-ABC Methods for E-commerce Platform.," *International Journal of Information Engineering & Electronic Business*, vol. 13, no. 1, pp. 1-14, 2021.

[2] S. Kedia, S. Jain and A. Sharma, "Price Optimization in Fashion E-commerce," *ArXiv*, vol. 2007.05216, 2020.

[3] J. Mou, W. Zhu and M. Benyoucef, "Impact of product description and involvement on purchase intention in cross-border e-commerce," *Industrial Management & Data Systems*, vol. 120, no. 3, pp. 567-586, 2020.

[4] G. A. Spedicato, C. Dutang and L. Petrini, "Machine Learning Methods to Perform Pricing. A Comparison with Standard GLMs," *Variance, Casualty Actuarial Society*, vol. 12, no. 1, pp. 69-89, 2018.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)