Predict hotel survey experience and ratings

Devansh Sanghvi (be19b002@smail.iitm.ac.in)
Saurabh Tiwari (bs19b028@smail.iitm.ac.in)
CS5691- Pattern recognition and Machine Learning Data Contest (Aug 2022)

***Abstract*—This report aims to cover the fundamentals of Gradient Boosting while predicting the rating scores for hotel bookings made across multiple countries from 2006 to 2008. This paper also highlights the importance of feature extraction, feature selection, and feature manipulation for the more significant classification task.**

I. Gradient Boosting

The principle behind boosting algorithms is; first, we build a model on the training dataset, then a second model is built to rectify the errors present in the first model.

The main idea behind this algorithm is to build models sequentially, and these subsequent models try to reduce the errors of the previous model. This is done by creating a new model on the mistakes or residuals of the previous model. When the target column is continuous, we use Gradient Boosting Regressor, whereas when it is a classification problem, we use Gradient Boosting Classifier. The only difference between the two is the "Loss function." The objective is to minimize this loss function by adding weak learners using gradient descent. Following is the algorithm for Gradient Boosting:

1) Initialize model with a constant value:

$$F_0 = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{1}$$

2) for m=1 to M:

2.1. Compute residuals for i=1,...,n:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

2.2. Train regression tree with features x against r and create terminal node reasions R_{jm} for j=1,..., J_m

2.3. Compute for i=1,...,J_m:

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$$

2.4. Update the model:

$$F_m(x) = F_{m-1}(x) + v \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

Gradient Boosting has repeatedly proven to be one of the most powerful techniques to build predictive models in both classification and regression. Because Grading Boosting algorithms can easily overfit on a training data set, different constraints or regularization methods can be utilized to enhance the algorithm's performance and combat overfitting. Penalized learning, tree constraints, randomized sampling, and shrinkage can be utilized to combat overfitting. Exploring such methods is outside the scope of this project but is an exciting read.

A. How is it different from Adaptive boosting

- Gradient Boosting approach trains learners based upon minimizing the loss function of a learner while AdaBoost focuses on training upon misclassified observations. Alters the distribution of the training dataset to increase weights on sample observations that are difficult to classify.
- All the learners have equal weights in the case of gradient boosting. The weight is usually set as the learning rate, which is small in magnitude. In Adaboost, prediction is based on a majority vote of the weak learners' predictions weighted by their accuracy.

B. Advantages of Gradient Boosting

- Often provides predictive accuracy that cannot be trumped.
- Lots of flexibility - can optimize different loss functions and provides several hyperparameter tuning options that make the function fit very flexibly.
- No data pre-processing required - often works great with categorical and numerical values as is.
- Handles missing data - imputation not required.

C. Disadvantages of Gradient Boosting

- Gradient Boosting Models will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting.
- Computationally expensive - often requires many trees (more than 1000), which can be time and memory exhaustive.
- The high flexibility results in many parameters that interact and heavily influence the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.
- Less interpretative, although this is easily addressed with various tools.

D. Histogram-based Gradient Boosting Regressor

- This estimator is much faster than GradientBoostingRegressor for big datasets (n samples ¿= 10 000).
- This estimator has native support for missing values (NaNs). During training, the tree grower learns at each split point whether samples with missing values should go to the left or right child based on the potential gain.
- When predicting, samples with missing values are assigned to the left or right child consequently. If no missing values were encountered for a given feature

during training, then samples with missing values are mapped to whichever child has the most samples.

E. Why did we use it for the contest?

- Since there were a lot of features, we needed a model that should not overfit the training data but also have reasonably good accuracy.
- Hist Gradient boosting trees are more accurate because they are trained to correct each other's errors and capture complex patterns in different data types.
- Hist Gradient Boosting outperformed Random Forests and other boosting algorithms.

F. Evaluation Parameters in Gradient Boosting Regressor

To compare the various models, we need to evaluate metrics. Some metrics that can be used to compare are:

- Mean squared error: In statistics, the mean squared error (MSE) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function corresponding to the expected value of the squared error loss.
- R2-score: Coefficient of Determination: The coefficient of determination also called as R2 score, is used to evaluate the performance of a linear regression model. It is the amount of variation in the output-dependent attribute which is predictable from the input independent variable(s). It is used to check how well-observed results are reproduced by the model, depending on the ratio of total deviation of results described.
- Accuracy score: Used to compute the subset accuracy. It is equal to the Jaccard score for binary classification.

II. Preprocessing and Exploratory Analysis on the Data

In this section, we will discuss the approach used to classify user ratings for the given datasets. Given below are the datasets provided, the visualization of each feature in a specific dataset, and the modifications we made for the final model:

A. bookings.csv

Contains information about the unique bookings made. It has 99441 rows (bookings made) and six columns (booking details). **Table 1** summarizes the contents of the dataset. We will now assess the features in the dataset and modify them for maximum relevance to our model. Following are the insights gathered and the changes made:

1) **Booking Status**: As seen in Fig.1, the maximum number of bookings were completed (96478 bookings), and the rest of the categories are very small in number. Hence we can club the rest of the features as not completed (2963).
2) **Booking Create Time**: To extract information from this column, we extract the year, month, day of the week, and the time of the day.

TABLE I
BOOKINGS.CSV COLUMNS' INFORMATION

Column	Non-Null	Unique	Dtype
booking_id	99441	99441	Object
customer_id	99441	99441	Object
booking_status	99441	8	Object
booking_create_timestamp	99441	98875	Object
booking_approved_at	99281	90733	Object
booking_checkin_customer_date	96476	95664	Object

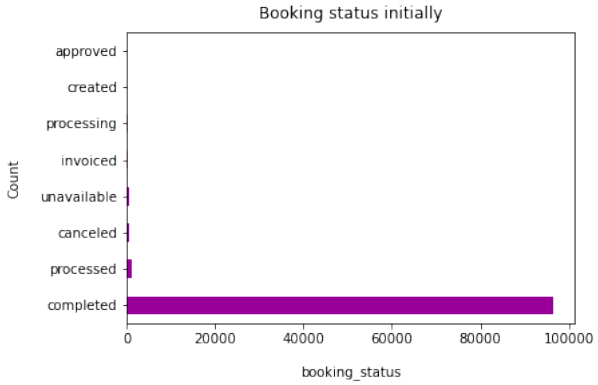

Fig. 1. Count Plot of initial booking status categories

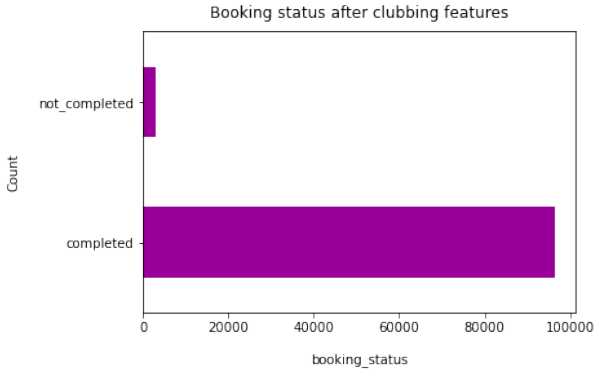

Fig. 2. Count Plot of final booking status categories

3) **Booking Approval Time**: This column, as shown in Table 1, has some missing values. These values were filled in using another criterion: if the booking was completed or not. We also created a new feature: the difference between approval time and create time (in minutes). The variation in approval time differences concerning booking status is shown in Fig.3.
4) **Checkin Time**: This column, as shown in Table 1, has some missing values. These values were filled in using another criterion: the year the booking was done. We also created a new feature: the difference between check-in time and create time (in hours). The variation in check-in time differences concerning years is shown in Fig.4.

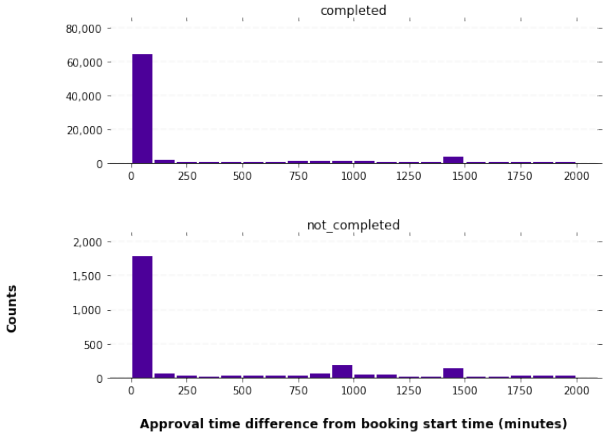

Fig. 3. Histogram of approval time difference

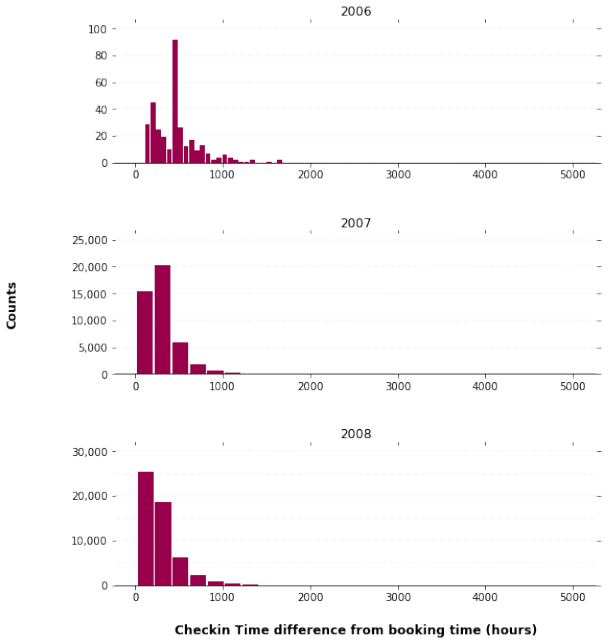

Fig. 4. Histogram of check-in time difference

B. *bookings_data.csv*

Contains information about the bookings made (non-unique). It has 112651 rows (bookings made) and seven columns (booking details). **Table 2** summarizes the contents of the dataset. We will now assess the features in the dataset and modify them for maximum relevance to our model. Following are the insights gathered and the changes made:

1) **Multiple booking ids**: There are booking IDs with multiple other bookings, as given by the booking _sequence _id. We remove the sequence id and make a column indicating the frequency of the bookings made by the same booking ID. Also, since there are only 98666 unique booking ids in our dataset, we fill in the hotel details for the other columns using appropriate measures (mean/mode).

TABLE II
BOOKINGS _DATA.CSV COLUMNS' INFORMATION

Column	Non-Null	Unique	Dtype
booking_id	112650	98666	Object
booking_sequence _id	112650	21	int64
hotel_id	112650	32951	Object
booking_expiry_date	112650	93318	Object
seller_agent_id	112650	3095	Object
agent_fees	112650	6999	float64
price	112650	5968	float64

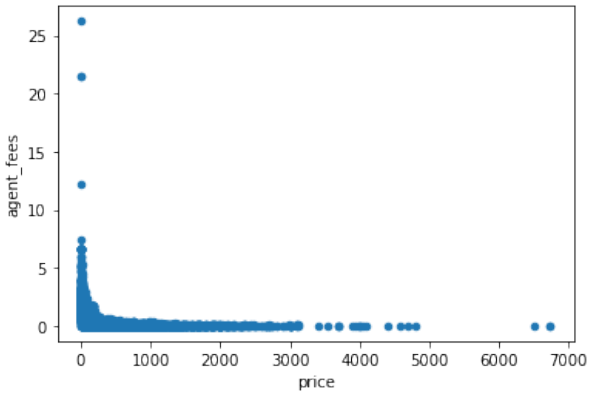

Fig. 5. Scatter plot of price v/s agent fees

2) **Hotel IDs** : Using the hotel _id, we create another column, which indicates the frequency of the bookings made for the hotel. Since the hotels are large in number, there was no point in comparing the hotel prices with mean hotel prices.
3) **Seller Agent IDs**: Using this column, we can count the number of bookings made per agent and the mean agent fees per agent. Both of them are helpful features in our model.
4) **Booking Expiry Date**: This column does not add a lot of information to the model. The expiry dates in the column are about ten years from the booking and can be put into three categories: ¡3665 days, ¿3671 days, and between the other two categories.
5) **Price**: This column will be very useful as it is since it is a major factor in user ratings. We will keep it unchanged.
6) **Visualizations**: The plot in figure 5 indicates that there is no relation between the price of the hotel and the agent fees.

C. *customer _data.csv*

Contains information about the customers who made the bookings. Has 99441 rows (customers) and three columns (customer details). **Table 3** summarizes the contents of the dataset. We will now assess the features in the dataset and modify them for maximum relevance to our model. Following are the insights gathered and the changes made:

1) **Customer Unique ID**: Using the unique IDs, we can see the number of bookings a particular customer made.

TABLE III
CUSTOMER _DATA.CSV COLUMNS' INFORMATION

Column	Non-Null	Unique	Dtype
customer_unique _id	99441	96096	Object
customer_id	99441	99441	Object
country	99441	9	Object

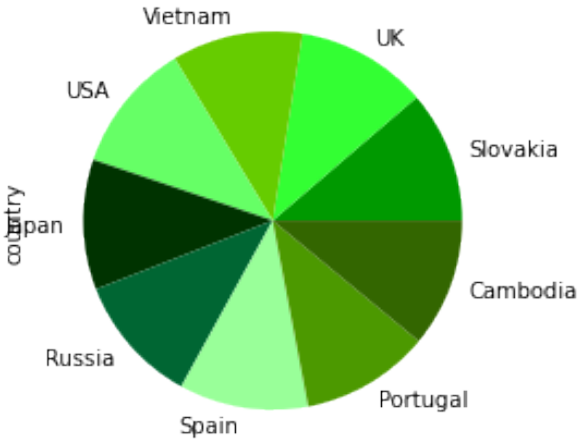

Fig. 6. Pie Chart of country v/s bookings

We make three relevant categories for the number of bookings.

2) **Country**: We add another column representing the number of bookings made from the country by a particular customer.
3) **Visualizations**: As seen in fig. 6, almost all countries have an equal number of bookings. Therefore, there is no geographical bias in the data.

D. hotels _data.csv

Contains information about the hotels booked. Has 32951 rows (customers) and five columns (hotel details). **Table 4** summarizes the contents of the dataset. We will now assess the features in the dataset and modify them for maximum relevance to our model. Following are the insights gathered and the changes made:

1) **Missing Values in all columns**: We used traditional filling techniques: mean for name length, photos quantity and description length, and mode for the category.

TABLE IV
HOTELS _DATA.CSV COLUMNS' INFORMATION

Column	Non-Null	Unique	Dtype
hotel_id	32951	32951	Object
hotel_category	32341	73	float
hotel _name _length	32341	66	float
hotel _description _length	32341	2960	float
hotel _photos _qty	32341	19	float

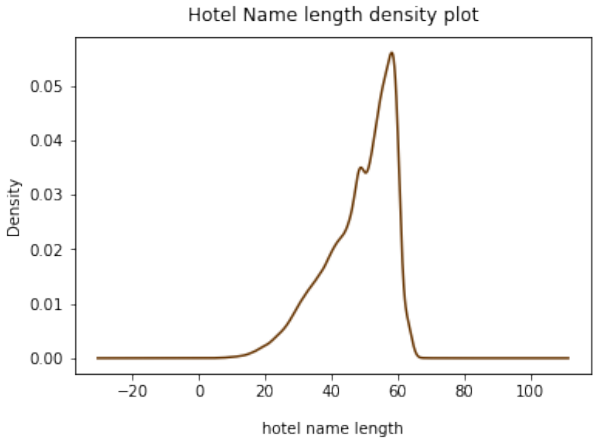

Fig. 7. Density plot of hotel name length

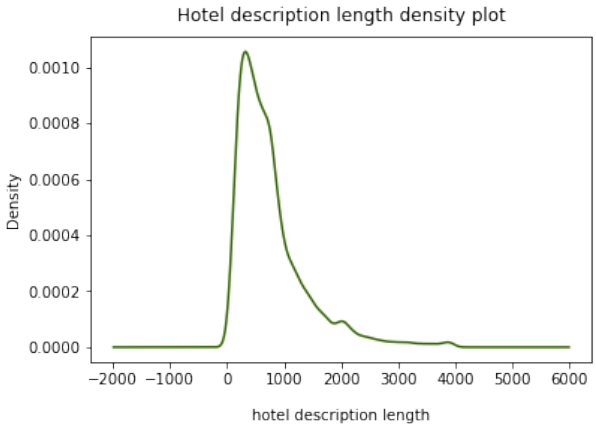

Fig. 8. Density plot of hotel description length

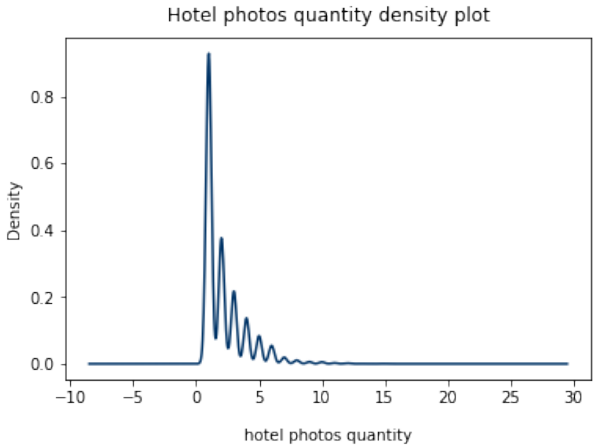

Fig. 9. Density plot of hotel photos quantity

2) **Visualizations**: The three density plots shown in fig.7, fig.8, and fig.9 show the distribution of the features: hotel name length, hotel description length, and hotel photo quantity.

TABLE V
PAYMENTS _DATA.CSV COLUMNS' INFORMATION

Column	Non-Null	Unique	Dtype
booking_id	103886	99440	Object
payment_sequential	103886	29	int64
payment _type	103886	5	object
payment _installments	103886	24	int64
payment _value	103886	29077	float64

E. payments _data.csv

Contains information about the payments made. Has 103886 rows (payments) and five columns (payment details). **Table 4** summarizes the contents of the dataset. We will now assess the features in the dataset and modify them for maximum relevance to our model. Following are the insights gathered and the changes made:

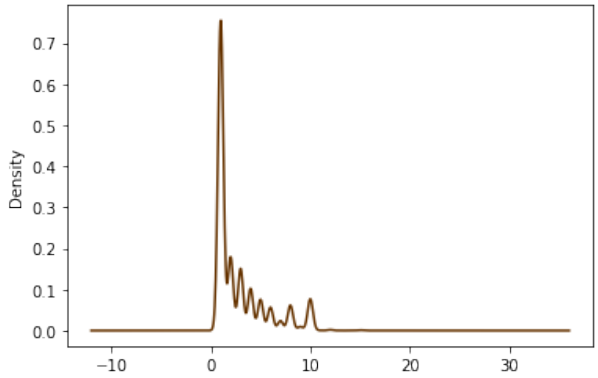

Fig. 10. Density plot of payment installments

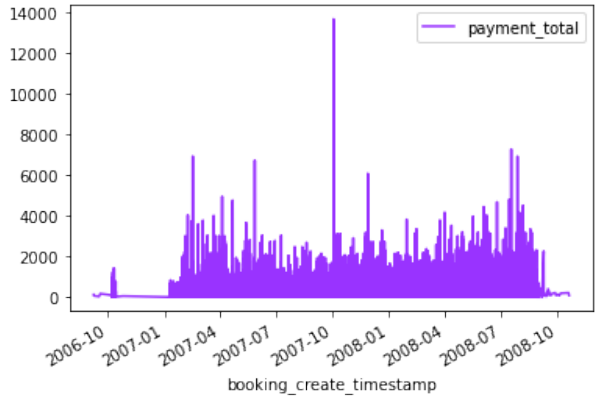

Fig. 11. Count plot of the number of payments

1) **Multiple Booking IDs**: Like the bookings _data dataset, we have multiple payments using the same booking ID. To account for this, we make a new column for the number of payments made using that booking ID. We also take the mode of payment types for the same reason. We take the sum of the installments for each booking ID, and we sum the values of each payment by a booking ID for the total payment made. These are the new columns replacing the ones before. We do not remove the duplicates, though, as it seems to give better accuracy.
2) **Visualizations**: The three plot plots shown in fig.10, fig.11, and fig.12 show the distribution of the features: payment installments, payment type, and a number of payments.

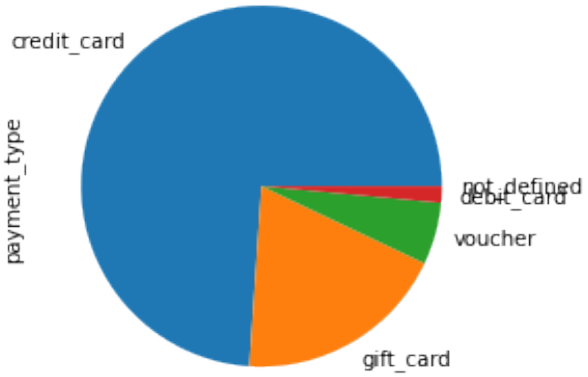

Fig. 12. Pie chart of type of payments

III. THE FINAL DATASET

After preprocessing the data in each dataset, we merge the datasets to make our final dataset used for training and testing data. The training data we provided has 50000 booking IDs and their user ratings. Using the features, we extract information about the booking IDs, and then we use the features to train our model and predict unknown/ known user ratings for booking IDs. We merge in the following order:

1) Merge the*bookings.csv* with the *customer _data.csv*, using the customer ID as the common column.
2) Merge the*bookings _data.csv* with the *hotel _data.csv*, using the hotel ID as the reference column.
3) Merge the*bookings.csv* with the *bookings _data.csv*, using the booking ID as the common column.
4) Merge the*bookings.csv* with the *payments _data.csv*, using the booking ID as the common column.

The final dataset has **118434** entries, and 33 columns. We remove the features of no use, i.e., the booking, the customer and the hotel IDs, the timestamps, and the booking expiry date. We have already extracted the information required from them. Finally, before we move to the encoding step, we have **28 meaningful features**, with both categorical and numerical features. Table 6 summarizes the final features.

Using **One-Hot Encoding**, we convert the categorical features to data that the model would understand. Hence, in our final dataset, we have **116 columns** and **118434 rows**. The memory usage for the same is 29.3 MB. For better accuracy

TABLE VI
FINAL DATASET'S FEATURES

Feature	Categorical	No of Categories
booking_status	Yes	2
approval_time _diff	No	-
booking _quarter	Yes	4
booking _year	Yes	3
booking _dayofweek	Yes	7
booking _0to8	No	-
checkin _quarter	No	-
checkin _year	No	-
checkin _dayofweek	No	-
checkin_time _diff	No	-
country	Yes	9
price	No	-
freq	No	-
customer_bookings _qty	No	-
agent_fees	No	-
agent_mean _fees	No	-
agent_count	No	-
hotel_fees	No	-
hotel_category	Yes	73
hotel _name _length	No	-
hotel _description _length	No	-
hotel _photos _qty	No	-
payment _freq	No	-
payment _total	No	-
total _installments	No	-
expiry _days	Yes	3
mode _payment _type	Yes	4

and speed, we also scale the features using *StandardScaler()* from *sklearn*.

IV. THE GRADIENT BOOSTING REGRESSOR MODEL

We used the sklearn library for our models and tested different parameters for the model, i.e., number of iterations, max depth of a tree, and others. However, to our surprise, we received the best results using the default parameters. On unseen data provided on Kaggle, our model had a loss of **1.37634** (private leaderboard). Fig.5. is the feature importance representation of our model (This indicates the importance of each feature considered in our model). Note that we have only chosen the top 10 features for better visualization.

V. LIMITATIONS AND SCOPE OF IMPROVEMENT

While the model performed decent, some improvements could be made:

- There were about 72 hotel category features out of the 116 features. As seen in the feature importance chart (Fig.5), hotel categories do not contribute much, so compressing the features could get better performance.
- Hyperparameter tuning using the most relevant hyperparameters.
- For the check-in time and the booking approval time, we filled the NA values with a value, but ideally, since the booking was not made, we should not have values for them. However, these features were quite important to our model.

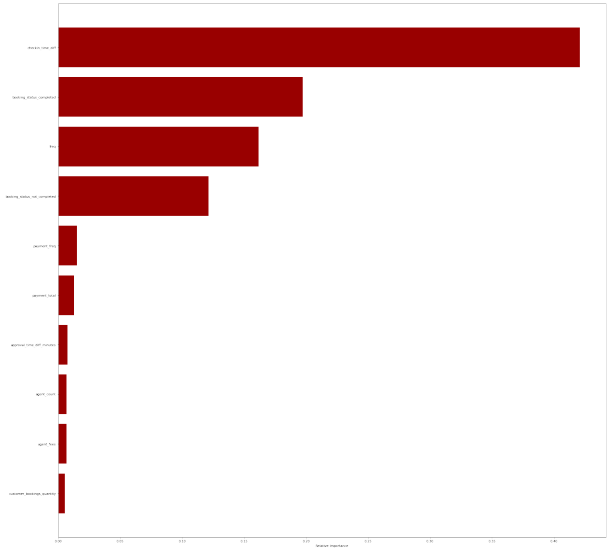

Fig. 13. Horizontal Bar plot indicating the importance of 10 best features

VI. CONCLUSION

The Gradient Boosting Regressor gave excellent results for the datasets involved. The most surprising crucial features were: the check-in time difference, approval time difference, and the timings of the bookings. This indicates that the user ratings highly depended on when they were booked. This also makes sense since the devastating financial crisis in 2008 sent the industry into a complete downfall. Other features like the number of bookings made by the customer, agent fees, and the price of the booking were of great importance, as expected.

REFERENCES

[1] "Data Contest 2022", scikit-learn.org, 2022 [Online]. Available: **Link**.
[2] "sklearn.ensemble.HistGradientBoostingRegressor", Sklearn Documentation, 2022 [Online]. Available: **Link**.