CS5691 : Pattern Recognition and Machine Learning

Chellarapu Priyanka & Meghana Gopal Soni

4th December 2022

1 Introduction

In this project, we were asked to experiment with a real-world data set to build a prediction model for customers to predict rating scores that they will likely assign to hotel bookings. The training data given has customer ratings corresponding to each booking id. Extra information was provided such as customer info, his/her demographic details, past experience ratings, hotel offering, etc. We were supposed to come up with a ML model that predicts customers' scores among different hotel bookings.

2 Metrics

MSE (mean squared error) is a common metric for regression models, it is the squared difference between the predicted value and the real value.

$$MSE = \frac{1}{n}\Sigma(y - \hat{y})^2$$

The lower the MSE, the better the prediction.

3 Data Provided

3.1 bookings.csv

- **booking_id** : Unique identifier for each booking.
- **customer_id** : Unique identifier for each customer.
- **booking_status** : Categorical data describing the stage of booking.
- **booking_create_timestamp** : DateTime data when booking was created.

- **booking_approved_at** : DateTime when booking was approved by the hotel.
- **booking_checkin_customer_date** : DateTime for customers check-in schedule.

3.2 bookings_data.csv

- **booking_id** : Unique identifier for each booking.
- **booking_sequence_id** : Integer 1,2,3,4... representing any other sub-requests within the same booking.
- **hotel_id** : Unique identifier for each hotel offering.
- **seller_agent_id** : Unique identifier for agents who advertise/list the hotels.
- **booking_expiry_date** : DateTime when the booking is deemed invalid.
- **price** : Price amount payable by customers.
- **agent_fees** : The intermediate fees charged by agents.

3.3 customer_data.csv

- **customer_id** : Unique-ID, these are assigned by the system to each booking. This means that the same person will get different ids for different bookings.
- **customer_unique_id** : Unique-ID, can help in identifying the same person that made re-bookings at multiple occasions.
- **country** : Demographic data of customer.

3.4 hotels_data.csv

- **hotel_id** : Unique-ID for each hotel offering listed.
- **hotel_category** : Categorical data segregating hotels by internal cataloguing methods.
- **hotel_name_length** : Integer, to depict the length of name of hotel advertising.
- **hotel_description_length** : Integer, depicts the length of text description on hotel advertisement listings.
- **hotel_photos_qty** : Integer, number of images uploaded for the hotel listing.

3.5 payments_data.csv

- **booking_id** : Unique identifier for each booking.
- **payment_sequential** : Integer, capturing sequence id of multiple payments made for a booking.
- **payment_type** : Categorical data, depicting the mode of payment.
- **payment_installments** : Integer, number of installments to complete the payment over a longer time for a particular hotel booking.
- **payment_value** : Real value, the price paid by the customer eventually to complete the booking's related payment.

4 Handling data

4.1 NaN entries

Since a few field entries in the provided data were empty or not available, we sanitized the data before moving ahead by filling the entries that were initially NaN to the average value of the feature.

Fields for which average could not be taken which are
booking_create_timestamp, booking_approved_at,
booking_checkin_customer_date, booking_expiry_date,
the dates were converted to a time-delta format using pd.to_datetime() and then changed to no. of seconds using to_seconds() function before taking the average for NaN entries.

4.2 Enumeration of features

Now, fields like agent fees and price can be used for say, regression, directly. However, others like customer_unique_id, booking_status, country and payment_type have to be encoded as an enumerated type. pd.factorize has been used for this purpose.

4.3 Time fields

The following new fields are created:

- **booking_time_taken**: the amount of elapsed between creation and approval of a new booking in seconds.
- **booking_time_crea_check**: the amount of time elapsed between creating a booking and checkin in seconds.
- **booking_time_exp_check** : the amount of time elapsed between expiry date of booking and checkin date in seconds.

These time fields that are created then turned out to be one of the most important features for this prediction model as it had **reduced the MSE from 1.66 to around 1.36**.

4.4 Final set of features

The final training dataset has 16 columns having the following fields:

- **booking_sequence_id** : maximum value of the booking_sequence_id column of the entries with same booking_id is taken as the final value of the feature.
- **price** : sum of price for all sub-bookings having same booking_id.
- **agent_fees** : sum of agent_fees for all sub-bookings having same booking_id.
- **hotel_category** : mean of the hotel_category of sub-bookings,
- **hotel_name_length** : mean of the name length of the hotel for each sub-booking all having the same booking_id
- **hotel_description_length** : mean of the length of description of different hotels for each sub-booking all having the same booking_id
- **hotel_photos_qty** : mean of the number of photos of different hotels for each sub-booking all having the same booking_id
- **booking_status** : a number from 1 - 8, representing the current booking status
- **booking_time_taken** : the amount of elapsed between creation and approval of a new booking in seconds.
- **booking_time_crea_check** : the amount of time elapsed between creating a booking and checkin in seconds.
- **booking_time_exp_check** : the amount of time elapsed between expiry date of booking and checkin date in seconds.
- **payment_installments** : average of the payment installments taken for each payment sequence with same booking_id.
- **payment_sequential** : maximum value of the payment_sequential column of the entries with the same booking_id is taken as the final value of the feature.
- **payment_type**: a number is assigned to each payment_type possible and the average of that number of each payment_sequence with the same booking_id is taken as the final value for this feature.

- **customer_unique_id**: a number is assigned to each customer
- **country**: a number is assigned to each country possible as is taken as the final value.

5 Prediction model: Histogram-based Gradient Boosting Classifier

Histogram-based gradient boosting is a technique for training faster decision trees used in the gradient boosting ensemble. We use the scikit-learn machine learning library that provides an implementation of gradient boosting that supports the histogram technique.

5.1 Hyper-parameters for the model:

There are three different hyper-parameters that we tuned in our algorithm:

- **learning_rate**: This is used as a multiplicative factor for the leaves values.
- **max_depth**: The maximum depth of each tree. The depth of a tree is the number of edges to go from the root to the deepest leaf.
- **min_samples_leaf**: The minimum number of samples per leaf.

5.2 Prediction on test set:

After getting the final set of optimal hyper-parameters the model on the 30% validation set and 70% training set, the model was trained using these hyper-parameters on the entire training set and ratings were predicted for the given public sample test set.

5.3 Final errors :

Errors were obtained on the public test set and later on the private test set as well.
The least error we could obtain for the public test set was: **1.38487**
Error, when the model was tried on the private data set, was: **1.38238**

6 Hyper-parameter tuning

We split the given data into 2 data sets namely, train data and validation data in a 0.7-0.3 split. For getting the optimal hyper-parameters, we check the MSE for different combinations of parameters and choose the combination that gives the least mean squared error on the validation set.

The ranges of values tried are as follows:

- **learning_rate** : [0.01, 0.02, 0.05, 0.1, 0.2]
- **max_depth** : [2,3,5,7,9]
- **min_samples_leaf** : [2,3,5,7,9]

7 Choice of the best model

Splitting the data into training and test data:
We split the data into 70% training data and 30% test data to check which model works the best.
The following models were tried:

- **Linear regression**: The MSE while using the linear regression model turns out to be 1.504.
- **Polynomial regression**: The MSEs with different hyper-parameters are -
  Degree = 2 : 1.360
  Degree = 3 : 3.165
  Degree = 4 : 2.747
- **Random forest regression**: The least MSE using a random forest regressor turned out to be around 1.35 - 1.36.
- **Hist Gradient Boosting Regression** : The least MSE using a Hist Gradient Boosting regressor was almost the same as a random forest regressor.

The decision to choose Hist Gradient Boosting Regression over Random Forest is because of the error we were getting on the final sample test set, which was lesser for Hist Gradient Boosting regressor as compared to Random Forest. Also with the number of combinations of hyper-parameters, we had to check for getting the least validation error, it was best to decide a regressor that was fast, hence we chose HistGradientBoostingRegressor as our final model. The other models (Linear Regressor and Polynomial Regressor) were given significantly more error on the final test set.

8 Other experiments and conclusions:

The reason why Linear regression did not work as well for this model could be because of the way the features influenced the rating given by customer. For example, features like price were not linear with the ratings, i.e for very low and for very high price we could see customers giving a 1 rating to the booking.
We also experimented with some of the other features like the seller_agent_id by assigning a number to each seller_agent, but in turn increased the error because when merging the payments file with the features file, the rows with different seller_agent_id could not be boiled down to a single number, taking average of them all did not make sense and was giving higher error than the model without this feature at all.