Machine Learning Techniques Final Project

Group: - - - Baseline - - -

Members: B06705055 陳柏儒 (Section 1~5)、R08723025 許惟傑 (6,7,9)、R09922A15 龍冠宇 (6,7,8)

1. Exploratory Data Analysis

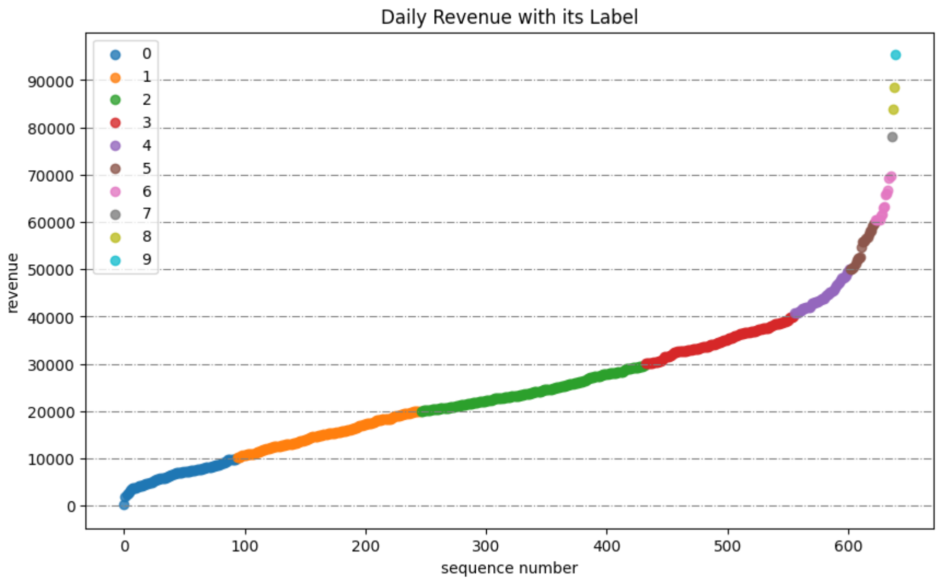
Before we conduct any analysis, it's necessary to dig into the data and get some useful information especially under the circumstance that we don't have any related domain knowledge. Thus, we are going to do some visualization below:

- Daily revenue with its label

  We first define that

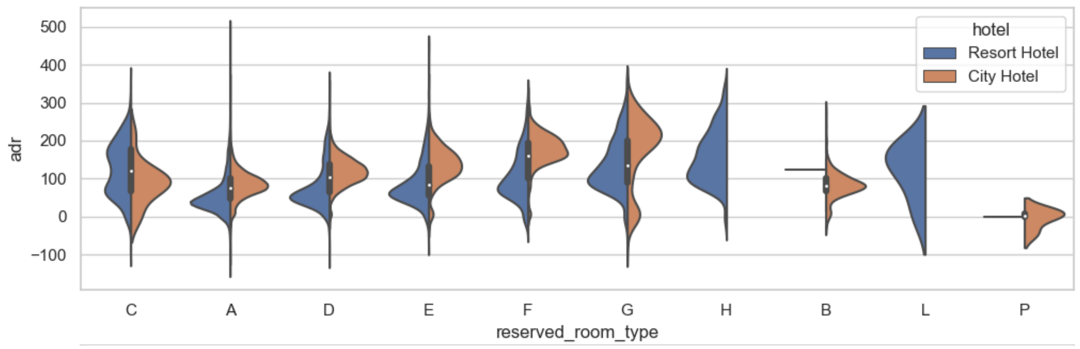
  $$Revenue = (1 - is\_canceled) * adr * (stays\_in\_weekend\_nights + stays\_in\_week\_nights)$$

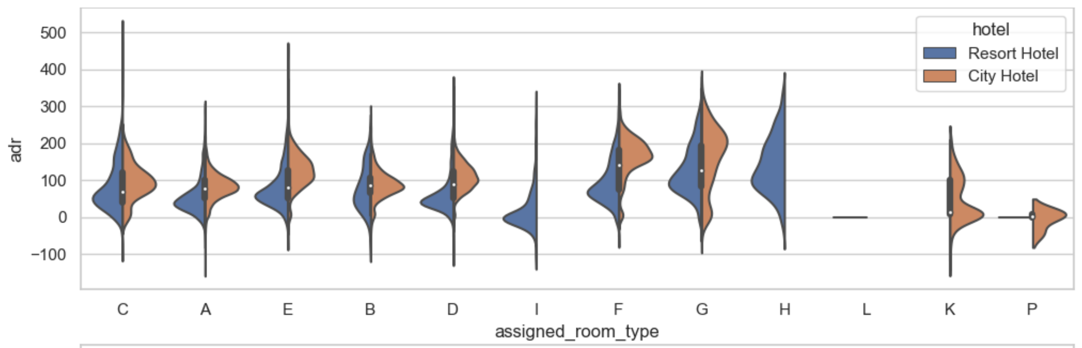
  Then, we can sum over each record's revenue and plot daily total revenue with the given label. It indicates that the label can be mapped to several intervals perfectly. Therefore, the main task is to determine the two random variables including 'is_canceled' (binary classification problem) and 'adr' (regression problem) and apply the formula above to generate the labels of the test dataset.

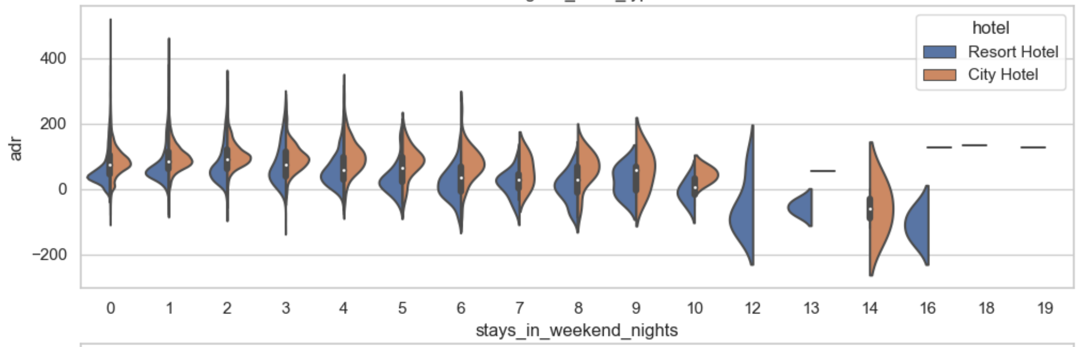


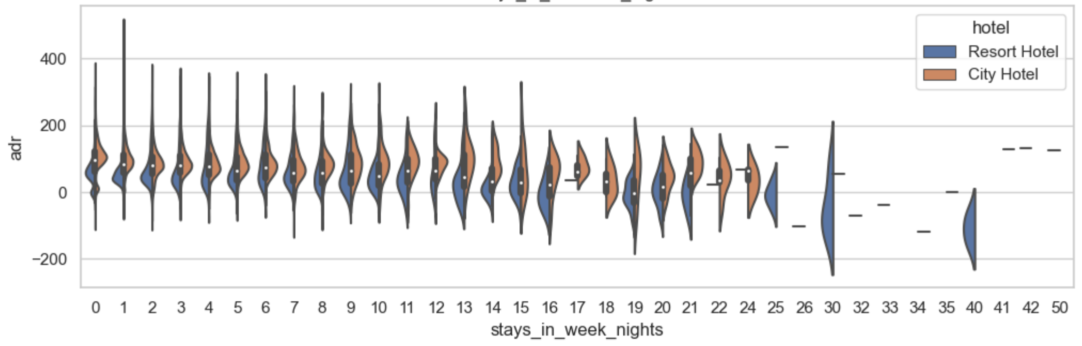
- Different populations of two hotels

  We are wondering whether the data of two hotels share the same population or not. Thus, we conduct violin plots of 'reserved_room_type', 'assigned_room_type', 'stays_in_weekend_nights' and 'stays_in_week_nights' according to the two hotels. The basic assumption is that the average 'adr' of the same room type may be similar; however, we can find that the same room type in the two hotels may have different meanings based on the plot. The similar outcome can also be found in other hotel-related features.
  Thus, we have to do some feature engineering corresponding to this property.

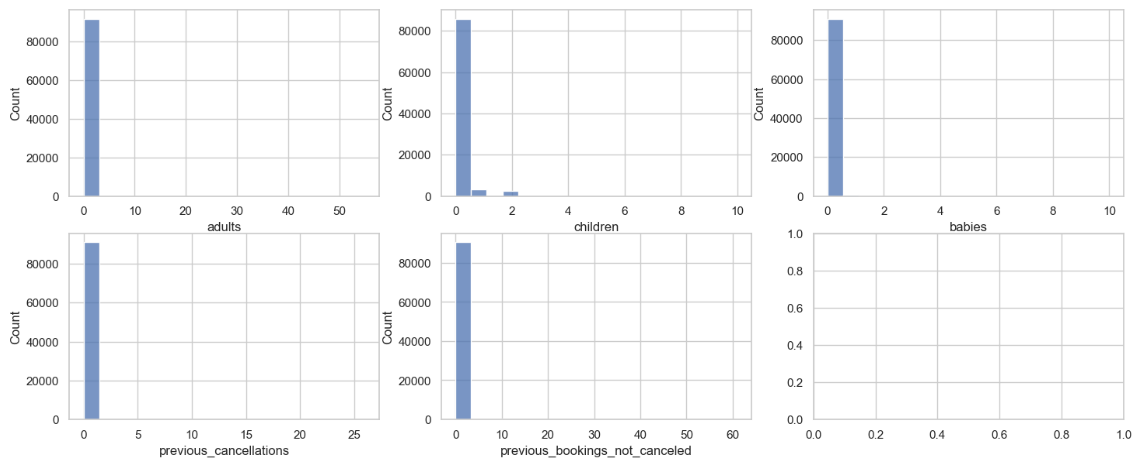






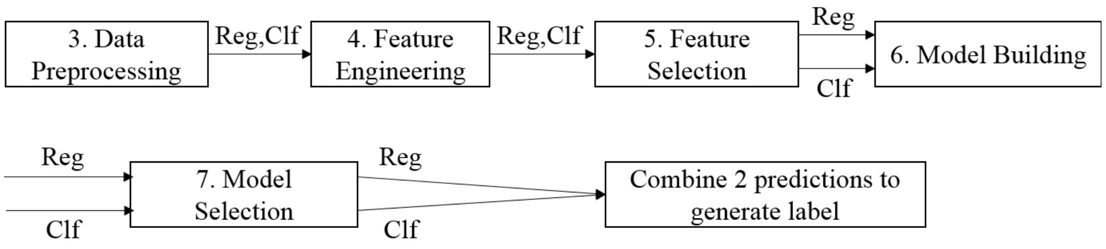


- Highly-skewed features
  From the histogram of ‘adults’, ‘children’, ‘babies’, ‘previous_cancellations’ and ‘previous_bookings_not_canceled’, we can find that these features are highly-skewed. Thus, we will perform log transformation on them.



2. Workflow

Reg: regression problem for ‘adr’; Clf: binary classification problem for ‘is_canceled’.
For step 5, 6, and 7, we conduct two methods for regression and classification respectively.



3. Data Preprocessing

3.1. Missing Value Handling

- From the dataset description[1], we know that missing values in some features represent another category e.g. ‘agent’ and ‘company’. Thus, we don’t drop or make imputation of these values but give them an unique label instead.
- For those training data who have missing value in ‘children’ or ‘country’, we drop them, since the imputation on these values doesn’t make sense.
- For those test data who have missing values in ‘country’, we fill in the mode (‘PRT’) in the training dataset.

3.2. Outlier and Future Data Removal

- We drop ID 31980 datapoint in the training dataset, since it has abnormally high ‘adr’ value.
- For ‘reservation_status’ and ‘reservation_status_date’ in the training dataset, we drop them because we can’t access them in the test dataset.

3.3. Unique Value of Categorical Features Reduction

The following transformations are applied on both training and test dataset.

- For ‘agent’, we keep the top 15 categories according to its percentage, and other categories are viewed as another one new category (23.1%).
- For ‘company’, we convert it to a binary feature, since the majority accounts for 94.0 percent.
- For ‘country’, we keep the top 15 categories according to its percentage, and other categories are viewed as another one new category (8.4%)
- For ‘room_type’, type ‘L’ in the training dataset and type ‘P’ in the test dataset are considered as one feature, since the data points are both few.

4. Feature Engineering

4.1. Feature Generation

- 1, if the ‘assigned_room_type’ equals to the ‘reserved_room_type’; otherwise, 0.
- ‘arrival_date_month’ are converted to categorical data to capture the seasonality. Other features related to time are dropped.

4.2. One-Hot Encoding:All categorical features are converted to binary variables with respect to unique value.

4.3. Log Transformation: From the EDA result, high-skewed features are transformed by log(x+1). Add one to avoid log(0).

4.4. Interaction: From the EDA result, we find that the two hotels may have different properties. Thus, each feature related to the hotel is converted to two new features e.g. ‘hotel_A_adults’ and ‘hotel_B_adults’.

4.5. Standardization: We use mean and standard deviation from the training dataset to standardize both the training and test dataset.

5. Feature Selection

Although the popular method to do dimension reduction is principal component analysis (PCA), we want to retain the interpretability as possible. Thus, we are going to use some simple models to extract the important features instead.

[1] https://www.sciencedirect.com/science/article/pii/S2352340918315191

5.1. Regression (adr): We use Lasso regression (linear regression with L1 penalty), since we believe that the relationship between adr and the features are linear. The number of features reduces from 340 to 40.

5.2. Classification (is_canceled): We use Logistic regression, since this linear model already performs well. (Linear First!) The number of features reduces from 340 to 47.

6. Model Building

6.1. Regression (adr)

For adr value prediction, we regard this problem as a regression problem. That is, we aim to take a sequence of input features, and predict the corresponding output value, where the value is a real number. To achieve this goal, we adopt several regression algorithms to validate which model performs the best on this particular dataset. To be more specific, we tried five algorithms, which are linear regression, Lasso, Gradient Boosting Regressor, XGBoost, and a DNN Fully Connected Network. Noted that we applied standard scaler normalization to our adr input features to standardize our input features by removing the mean and scaling to unit variance. Otherwise, the regression output distribution may be quite different from the input data distribution.

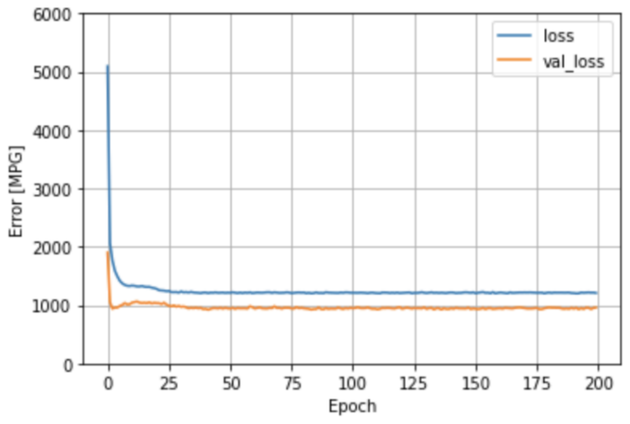
For each algorithm we tried, there is a reason behind it. The first method that comes into our heads is the well-known linear regression, since it is simple, straightforward and does not require exhausted hyper-parameter tuning. Although we know that this simple algorithm might not perform very well on this problem, we followed what this lecture has taught us, "Try simple models first" to avoid making our approach over-complicated.

The second algorithm is called LASSO (least absolute shrinkage and selection operator), which is a regression analysis model that performs L1 regularization. This method can enhance the prediction accuracy and interpretability of the resulting statistical model by lessening the model variables while preventing the overfitting situation.

The third method is Gradient Boosting Regressor. After our lecture covers boosting methods and aggregation models, we know that applying gradient boosting approach is an effective also essential technique when it comes to enhancing model stability and model accuracy. Therefore, we choose Gradient Boosting Regressor as our third model.

The fourth model is XGBoost (eXtreme Gradient Boosting). XGBoost is an advanced ML model which is ubiquitously used in many Kaggle competitions since it has a bunch of tunable hyper-parameters and training a XGBoost model can be relatively fast. We choose XGBoost as our fourth model since it is widely used in solving real-world problems and its performance and model stability have been tested by many ML practitioners.

The last method we choose is a DNN Fully Connected Network. Recently, Neural Networks have shown stunning performance in many research and industry fields, including advanced computer vision tasks, natural language processing, and many robotics applications. It can mimic how the human brain, e.g. biological neurons, works, and keep improving itself by minimizing the loss function. We are curious about how good DNN models can work on this particular dataset, so we build a DNN model to validate our thoughts. To be more specific, we built a 3-dropout-2-1 Fully Connected DNN model, where the "dropout" means a dropout layer with a dropout rate of 0.2. Each layer is followed by a relu activation function layer except for the last layer, because we are predicting outputs of real numbers. An interesting thing we found out is that adding more layers to the DNN model might not improve the model accuracy, instead, it can lead to severe overfitting results. Also, we found out that if we did not apply a dropout layer in the hidden layer while training, the model can still overfit with few layers and neurons. By conducting these experiments, we actually learn through practice that Machine Learning is not always about "computing power", choosing the "proper" algorithm with the proper hyperparameters are much more important than the model complexity itself. (below is a figure of our final training and validation results of the DNN model. One can see that our model does not overfit on this dataset after we applied a dropout layer.)

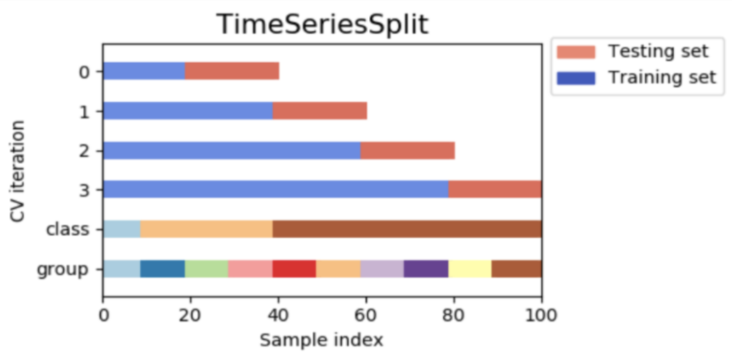


6.2. Classification (is_canceled)

Here we adapt logistic regression, naive decision tree, SVC and XGBoost to predict. We adapt logistic regression since it’s a simple model, which is easy to understand and has closed form solutions. Adapt tree method and SVC since they are well-known and can capture nonlinear relationship between feature and result. Although XGBoost requires high level ML knowledge and may be hard to explain the result, it’s a well known model used in many competitions and has a great performance.

7. Hyperparameter Tuning and Model Selection

We apply walk-forward cross-validation (WFAC) to evaluate model performance. We split the training data to 10 equal sized splits (G1, G2...G10). In each loop (i), we used G1~G(i) as a training set, and G(i+1) as an evaluation set. Finally, compute the average of the evaluation score and get WFAC accuracy.



7.1. Regression (adr)

Here is the comparison chart between all the regression models we used. We choose the top 2 candidate algorithms for hyperparameter tuning based on the walk-forward cross validation results.

Model	Linear	LASSO	GBoost	XGBoost	DNN
WFCV RMSE	33.22	34.93	27.81	27.35	29.03
Time	1.1 sec	2.4 sec	3.5 min	30.4 sec	1 hour
Efficiency	Very High	Very High	Low	Median	Low
Scalability	High	High	Median	Median	High
Popularity	Very High	Median	Median	High	High
Interpreability	Very High	High	Median	Low	Low

Details of our model hyperparameters tuning are illustrated as follows. (all the hyperparameter tuning processes are done with Hyperparameter Grid Search.)

- GBoost: 1. n_estimators: from 1000 to 3000 to enhance the model maximum capability. 2. learning_rate: from 0.1 to 0.05 to enable more precise gradient updates.
- XGBoost: 1. n_estimators: from 1000 to 3000 to enhance the model maximum capability. 2. learning_rate: from 0.1 to 0.05 to enable more precise gradient updates. 3. min_child_weight: from 1 to 1.7817 to prevent the model from overfitting.

7.2. Classification (is_canceled)

Here is the comparison between the models we used, we selected XGBoost to be our final model and started to do hyperparameter tuning since XGB has best performance when doing walk-forward CV. At the end we will predict the 'is_canceled' probability that takes value in [0,1].

Model	Logistic	Decision Tree	SVC	XGBoost
WFCV Accuracy	80.38%	76.38%	82.30%	84.46%
Time	17.58 sec	6.62 sec	3~4 hours	186 sec
Efficiency	High performs	Very High	Low	Median
Scalability	High	High	Low	Median
Popularity	Very High	High	High	Median
Interpreability	High	Very High	Median	Low

We changed the following hyperparameters. 1.scale_pos_weight : default = 1, we changed to 2 because of unbalanced data. 65% of customers will come to the hotel, the ratio of come/canceled is close to 2, so we add a larger penalty if we didn't predict the canceled case. 2.Max_depth : default = 3, we changed to 6 since it has better WFCV Accuracy. High Max_depth may cause overfitting and low Max_depth may cause underfitting (similar to the problem of tree model) 3. n_estimators : default = 100, we changed to 200 since it has better WFCV Accuracy. 4.colsample_bynode : default = 1, we change to 0.8 to prevent overfitting.

8. Final Approach

Now, we have built our models for "adr" and "is_canceled" separately as mentioned above, and we have a formula for calculating the final revenue[2] by these two. The final step to construct our final answer is to choose which adr model and is_canceled model to use to achieve the best accuracy. First, we combine each of our adr models with our is_canceled models (a total of 5*4=20 sets of model combinations).
We come up with two evaluation methods to choose our two final model combinations, which are 1. Choosing the best combined model with the highest public score since the distribution of the public testing set can be seen to be similar to the private testing set. 2. Choosing the best walk-forward cross validation adr model and is_canceled model, then combining them to prevent overfitting. By following our evaluation methods, the final models we used are 1. GBoost adr model + XGBoost is_canceled model. 2. XGBoost adr model + XGBoost is_canceled model. It is noteworthy that we adopt the probability output of the is_canceled model instead of the binary classification output.

9. Improvement and Conclusion

First, we decide to predict 'is_canceled'and 'adr' separately. After feature engineering and feature selection, we applied 5 models to predict 'adr' and 4 models to predict 'is_canceled'. We use 10 fold WFCV to evaluate the performance, and choose to apply GBoost/XGBoost adr model + XGBoost is_canceled model to compute revenue. In future, we can assign more weight on near data and create more features to approach better results.

[2] Revenue = (1 - prob(is_canceled)) * adr * (stays_in_weekend_nights + stays_in_week_nights). Here we take the probability of classification result, which is slightly different as we mentioned in section 1.