Optimizing hotel bookings with machine learning

Bryan Lee, Gan Rui Le, Justina Wong,
Tok Tze Kiat, Wang Xing Jie
{bryan.lee.2019, ruile.gan.2019, justinawong.2019,
tzekiat.tok.2019, xjwang.2019}@scis.smu.edu.sg

I. PROBLEM DEFINITION

Last-minute cancellations are a logistical and financial headache for hotels, but they have always been an unavoidable part of the industry.

A recent study by D-EDGE into reservation revenue shows that the overall cancellation rate across all channels reached a peak of 40% in 2018. This can be devastating for hotels that rely on predictable demand for the bulk of their revenue. To combat this, hotels try to overbook their rooms in anticipation of cancellations. However, if there are fewer cancellations than expected, hotels run the risk of angering their potential customers and receiving negative backlash.

Hence, this project aims to explore different machine learning techniques and models to generate predictions for the expected number of cancellations based on the attributes of each hotel bookings. This will allow hotels to:

- under book fewer rooms to reduce unallocated resources (fewer false negatives); and
- reduce the number of overbooked rooms (fewer false positives).

The negative effect of overbooking rooms is much greater than that of under-booking rooms. Therefore, a secondary objective is to **optimize for reducing overbooked rooms** i.e., optimize for precision over recall.

II. DATASET AND PRE-PROCESSING

The dataset initially contains a total of 31 feature columns and 1 dependent variable *is_canceled*. This project aims to utilize the feature columns to predict the value of *is_canceled*. **The data set is split with a 4:1 train-test ratio**.

A. Missing data

Figure 1 shows the number of missing or null values per column.

The *company* feature is dropped due to the high percentage of missing values (95.14%). The agent feature is kept as null values have semantic meaning representing bookings made without an agent. Entries with null children are treated as 0 children.

B. Distribution across predicted class

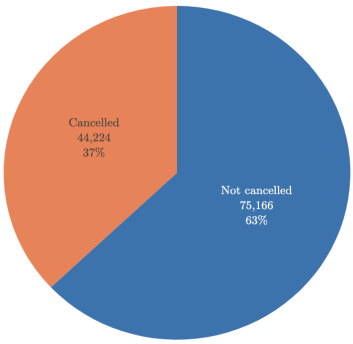

Figure 2. Distribution across is_canceled

hotel	0	*is_repeated_guest*	0
is_canceled	0	*previous_cancellations*	0
lead_time	0	*previous_bookings_not_canceled*	0
arrival_date_year	0	*reserved_room_type*	0
arrival_date_month	0	*assigned_room_type*	0
arrival_date_week_number	0	*booking_changes*	0
arrival_date_day_of_month	0	*deposit_type*	0
stays_in_weekend_nights	0	*agent*	16340
stays_in_week_nights	0	*company*	112593
adults	0	*days_in_waiting_list*	0
children	4	*customer_type*	0
babies	0	*adr*	0
meal	0	*required_car_parking_spaces*	0
country	488	*total_of_special_requests*	0
market_segment	0	*reservation_status*	0
distribution_channel	0	*reservation_status_date*	0

Figure 1. Number of null values per column

feature	**unique values**
hotel	['Resort Hotel' 'City Hotel']
meal	['BB' 'FB' 'HB' 'SC' 'Undefined']
market_segment	['Direct' 'Corporate' 'Online TA' 'Offline TA/TO' 'Complementary' 'Groups' 'Undefined' 'Aviation']
distribution_channel	['Direct' 'Corporate' 'TA/TO' 'Undefined' 'GDS']
is_repeated_guest	[0 1]
reserved_room_type	['C' 'A' 'D' 'E' 'G' 'F' 'H' 'L' 'P' 'B']
deposit_type	['No Deposit' 'Refundable' 'Non Refund']
customer_type	['Transient' 'Contract' 'Transient-Party' 'Group']
reservation_status_month	[7 5 4 6 3 8 9 1 11 10 12 2]
through_agent	[0 1]

Figure 3. Selected categorical features

When split along *is_canceled*, the dataset is mildly unbalanced, with 37% of bookings being cancelled and 63% non-cancelled. To compensate for the imbalance, **imblearn.over_sampling.RandomOverSampler** is applied to over-sample the training data for more cancelled samples to create a balanced training dataset.

C. Features

The dataset initially contains a total of 31 feature columns.

a) Derived features

reservation_status_date is split into *reservation_status_month* and *reservation_status_year*.

There are too many unique values for *agent*. Therefore, *through_agent* is derived to represent whether a booking is made through an agent; *through_agent* is true if and only if *agent* is not null.

b) Dropped features

Numerical features with less than 5% correlation coefficient to *is_canceled* are automatically dropped.

reservation_status_year and *arrival_date_year* are dropped as yearly effect on cancellation rates cannot be extrapolated into the future.

days_in_waiting_list is dropped due to the extreme outliers present. The minimum, median, lower quartile, and upper quartile is 0 while the maximum value is 391 days. As such, any correlation between the feature and the target variable is solely the result of outliers.

country is dropped due to the over-representation of 5 values: PRT, GBR, FRA, ESP and DEU. The top 5 countries represented 69.07% of all samples.

As a result, the following categorical and numerical features below remain.

c) Categorical features

Each categorical feature is one-hot encoded into dummy columns. The first column for each feature is dropped to prevent **multilinearity**.

d) Numerical features

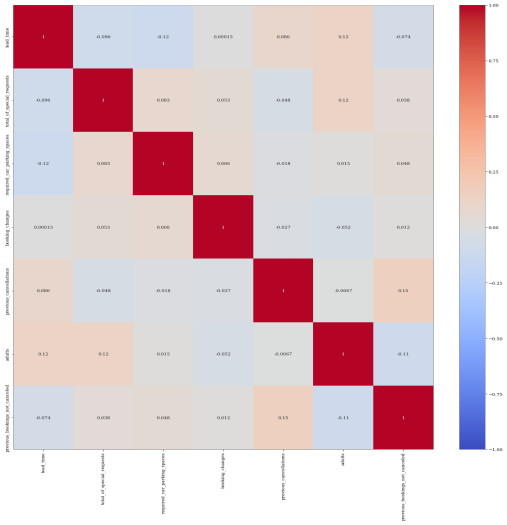

Figure 4. Correlation heatmap of numerical features

A correlation heatmap was produced to analyse the correlation between numerical features. Given the low absolute correlation (< 0.70) between features, principal component analysis (PCA) was not required as it would obfuscate the semantic meaning of each principal axis without providing much benefit.

feature	correlation to *is_canceled*	variance
lead_time	0.2931	11420
total_of_special_requests	0.2346	0.6285
required_car_parking_spaces	0.1955	0.0602
booking_changes	0.1444	0.4255
previous_cancellations	0.1101	0.7129
adults	0.0600	0.3355
previous_bookings_not_cancelled	0.5736	2.242

Figure 5. Selected numerical features

After categorical and numerical features were combined, a second stage of correlation analysis reveals high correlation from *distribution_channel* to *market_segment* and *through_agent*. Therefore, *distribution_channel* is also dropped.

This results in the reduction of 31 features to 16 features that will be used to train the machine learning models.

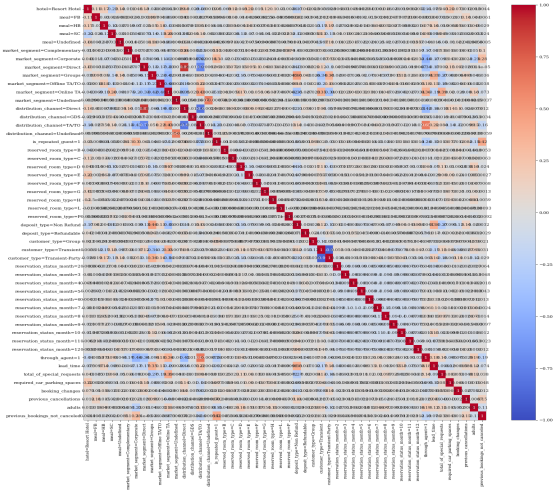

Figure 6. Correlation heatmap of all features

e) Feature Scaling

When combining numerical and categorical variables into a single data frame, a large difference in variance between features is observed, especially between numerical and categorical features. This causes features with large variances to dominate others during model training. To resolve this, all features in **X_train** are scaled with **StandardScaler**.

III. MACHINE LEARNING MODELS AND TECHNIQUES

Five different machine learning techniques are explored: (1) Decision Tree Classification, (2) Logistic Regression, (3) Artificial Neural Networks, (4) Random Forests, and (5) Ensemble Stacking.

A. Decision Tree Classification

Decision trees make use of observations about each sample to conclude its target value – in this case the classification of a sample on the *is_canceled* feature. Training a decision tree classifier involves recursively finding the optimal decisions to split a problem into subproblems with the highest information gain.

B. Logistic Regression

Logistic regression classifiers represent features as numerical data on which to find a linear decision boundary to classify samples into a positive or negative class. Each sample is given a probabilistic score by comparing its log likelihood of belonging to either class, then a threshold value can be tuned to adjust the boundary. Training a logistic regression classifier involves finding the weights of features that produce the maximum likelihood of observing the training samples.

C. Neural Networks

A neural network is a network of artificial neurons (nodes) organised in layers: with an input layer, an output layer, and hidden layers in between. Each node comprises a non-linear function applied on the dot product of weights and inputs with a bias. Training a neural network involves repeatedly finding the gradient of changes required to reduce loss on the output layer, then finding the changes on the weights and biases of all nodes via backpropagation to produce the gradient required.

D. Random Forest Classification

A random forest is an ensemble learning method where a collection of simple random decision trees is constructed and trained together. At every layer, each forest is assigned a set of randomly selected features to optimally split the problem on. The output of the random forest during a classification task is the class selected by most trees.

E. Stacking

Stacking combines heterogeneous weak learners to create an ensemble model. Each learner is trained on a subset of training data and validated against the remaining subset to produce predictions against the training dataset. Each learner is then fitted against the training dataset and predictions are made on the test dataset. Stacking creates a meta-model that is trained on the initial training predictions to make a final prediction against test predictions.

IV. HYPERPARAMETER TUNING

A. GridSearchCV

The hyperparameters of the Decision Tree Classifier, Logistic Regressor, and Random Forest are tuned with **GridSearchCV**.

GridSearchCV accepts a collection of hyperparameters to explore and automatically finds the optimal set of parameters for a given model and training dataset. **GridSearchCV** automatically performs stratified K-Fold validation for binary classifiers during the optimization of hyper-parameters to minimise overfitting. **cv**=5 is chosen as it provides a good compromise between the number of fits required and performance.

The different hyperparameter values explored for each model and the optimal hyperparameters are presented next.

a) Decision Tree

parameter	values	optimal
depth	[20, 30, 40, 50]	50
min_samples_leaf	[2, 3, 5, 8]	2

Figure 7. Decision tree hyperparameters and optimal values

b) Logistic Regression

parameter	values	optimal
C	[0.1, 0.01, 0.001]	0.1
solver	['lbfgs','sag']	'lbfgs'

Figure 8. Logistic regression hyperparameters and optimal values

c) Random Forest

parameter	values	optimal
n_estimators	[100, 200]	100
max_depth	[5, 8, 13]	13
min_samples_split	[2, 4]	2
min_samples_leaf	[2, 4]	2

Figure 9. Random forest hyperparameters and optimal values

B. Ensemble stacking configuration

The ensemble stacking classifier is built using the Decision Tree, Logistic Regression, and Random Forest models with the optimal parameters provided by GridSearchCV.

C. Neural network configuration

For the Neural Network, two models are created with similar configurations but with variations in the activation functions of the hidden layers. Both models are configured with:

(1) an input layer with 46 nodes to capture each feature of the processed dataset;
(2) two dense layers with 64 nodes; and
(3) an output layer with one node and a sigmoid activation function.

The activation function for the two hidden layers is *ReLU* for the first model and *tanh* for the second.

The models are compiled with a binary cross entropy loss function and trained with stochastic gradient descent. The models are trained with 10 epochs, a batch size of 32 and a learning rate of 0.01.

Model	Accuracy	Precision	Recall	F-0.5	ROC	MCC
Decision Tree	0.8346	0.7860	0.7691	0.7825	0.8604	0.6459
Logistic Regression (threshold = 0.7)	0.8061	0.9354	0.5199	0.8065	0.8578	0.5937
Neural Network (threshold = 0.6)	0.8136	0.9442	0.5356	0.8192	0.8940	0.6109
Random Forest (threshold = 0.6)	0.8028	0.9611	0.4952	0.8088	0.8992	0.5922
Ensemble Stacking	0.8320	0.8358	0.6879	0.8014	0.9012	0.6356

Figure 10. Model performance results

Neither neural network significantly outperforms the other. The model with *tanh* activation is selected for further analysis.

V. PERFORMANCE EVALUATION

The performance of each model is evaluated on accuracy, precision, recall, F-0.5, ROC score, and Matthews Correlation Coefficient (MCC). The results are consolidated in Figure 10.

An F-beta of 0.5 is selected to give more weight to precision over recall. Threshold values are selected to optimize F-0.5 when appropriate.

VI. MODEL SELECTION

Given the project goals to minimise overbooked rooms and optimize the model on precision and F-0.5, our recommendation is to select the **decision tree**.

A. Advantages of decision tree

The recommendation is based on four considerations:

(1) decision trees are highly performant with high accuracy and MCC;
(2) decision trees are transparent and can be used to better explain how the model classifies a booking either as cancelled or not;
(3) decision trees are resilient to dirty data and will not be overly sensitive to input; and
(4) decision trees are quick to train and require minimal resources.

We note that there are a few shortcomings with the decision tree, mainly the lack of an easily tuneable threshold parameter to improve precision and F-0.5 as per our project goals. However, we believe that the advantages granted by the decision tree outweigh the costs.

B. How the model explains the predictions made

Decision trees make use of observations about each sample to conclude its target value – in this case the classification of a sample on the *is_canceled* feature. This offers a distinct advantage in the transparency of the model as decision epochs can be followed. The head of a sample decision tree is provided in Figure 11, with the number of samples considered per node split on *is_canceled* value=[0, 1]. The feature names are provided in Figure 12.

For example, the decision tree indicates that given a sample, if deposit type is non-refundable, then the working sample size drops from 120,518 to 100,836. If the required number of parking spaces is also greater than 2.041, then the decision can be made that the booking will likely not be cancelled.

VII. FUTURE ISSUES

A. Introduction of new features

An increase in observations could result in features that were previously dropped due having highly skewed outliers to be more normally distributed. These features might be an important feature that influences the cancellation rate.

Furthermore, advancements in data collection could provide more insightful features.

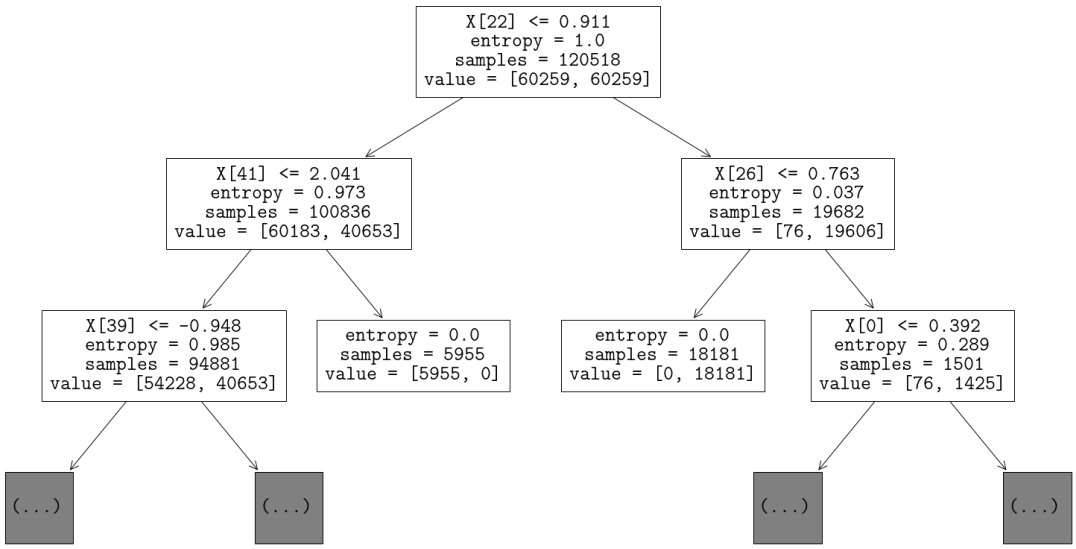

Figure 11. First two layers of decision tree

Feature	Name	Feature	Name
39	lead_time	10	market_segment=Online TA
22	deposit_type=Non Refund	44	adults
40	total_of_special_requests	42	booking_changes
41	required_car_parking_spaces	15	reserved_room_type=D
43	previous_cancellations	0	hotel=Resort Hotel

Figure 12. Decision tree important feature names

Therefore, the current models may become outdated. In updating the model, simply training on the new data is insufficient as the models may overfit the newly collected features if they are trivial.

Instead, new rounds of data exploration and pre-processing (or regularisation if resources are limited) would have to be performed. For example, pruning can be introduced on decision trees to prevent splitting on too many new features.

B. Imbalance of future dataset

Another risk as the dataset increases in size is that the dataset could become more imbalanced. As a result, accuracy would not be as effective in evaluating the model as predicting the majority class would lead to a high accuracy. Instead, more over-sampling techniques such as SMOTE or random over-sampling would have to be introduced. This runs the risk of over-representing certain samples with large skews in data.

C. Semantic meaning

A tricky issue would be accounting for the changing importance in some features. Features that were once thought to be relatively stable and insignificant could change and become highly important due to external factors.

Hotel bookings – being heavily influenced by both domestic and global travel – have seen a large change in the last few years due to the COVID-19 pandemic. Considering this realisation, features like *arrival_date* would become more significant. Perhaps new features, such as the travel policy adopted by source and destination countries at the time of booking, could be proactively collected and be used to train models for improvements in performance.