Hotel Cancellations: To Book or Not to Book

Team members and emails:
Kristen Lin (kristenlin@berkeley.edu)
Emily Zhao (emily_zhao@berkeley.edu)
Karan Patel (karankpatel192mids@berkeley.edu)
Chad Vo (chadvo@berkeley.edu)

Github Repo:
https://github.com/chadpvo/mids-207-summer-2025-keck

---

1 Abstract

Are cancellations mostly driven by impulse, or are there other variables at play—such as the customer's booking behavior, previous experiences, or the time of year they chose to travel? What predictive strategies or analytical approaches could be implemented to improve cancellation management and operational decision-making? To answer these questions, we aimed to develop machine learning models that can predict the likelihood of a cancellation based on booking-related features with high precision. We explored several supervised learning algorithms well-suited for predicting categorical outcomes, including logistic regression, decision trees, random forests, and XGBoost. Our methodology focused on building models with high precision and a good balance of recall, while also identifying the most influential features contributing to cancellations. With its gradient boosting approach and ability to model sequential trees, XGBoost outperformed other models by capturing complex, non-linear patterns present in the dataset.

2 Introduction

Hotel booking cancellations represent a significant challenge for the hospitality industry, leading to revenue loss, inefficient resource allocation, and decreased customer satisfaction. Understanding and predicting cancellation behavior can provide actionable insights for improving booking policies, optimizing inventory management, and launching more strategic marketing campaigns to appeal to customers. Various factors such as booking lead time, daily room rates, market segment, distribution channel, special requests, and other customer booking details can contribute to cancellation likelihood.

Leveraging machine learning to analyze large-scale hotel reservation data enables the development of predictive models that identify high-risk bookings before they cancel. These insights can support hotels in implementing dynamic pricing, personalized retention efforts, and overbooking strategies that balance guest experience with operational efficiency. As data availability grows, machine learning can offer increasingly accurate predictions that help turn reactive cancellation management into proactive decision-making. The input to our models consists of numerical, binary, and encoded categorical features derived from reservation details and client demographics in the Hotel Booking Demand dataset. We explored using Logistic Regression, Random Forest, XGBoost, and Neural Network models to predict whether a reservation will be cancelled.

3 Related Work

The airline and healthcare industries offer valuable perspectives on cancellations, which can inform the development of more effective hotel booking cancellation models.

In the airline industry, Cao et al. (2010) applied a range of data mining techniques to improve the forecasting of airline passenger no-shows. Their study utilized regression, decision trees, and neural networks to analyze their booking data, and ultimately found that their logistic regression model had the greatest accuracy in predicting no-shows. However, the authors emphasized the importance of selecting relevant features and concluded that the low probability event of no-show may have affected the results of their models. These were all learnings we took into consideration for this project.

Moreover, Dunstan et al. (2023) applied machine learning techniques to predict no-show appointments in a pediatric hospital in Chile. The authors compared several models such as random forests with various ensemble methods, logistic regression, and support vector machines, and applied balancing techniques due to their imbalanced data. They found that key predictors included reservation delay, historical no-show patterns, patient age, appointment type, and time of day. Additionally, across most specialties, the tree-based classifiers were selected as the best performers. This study gives us an idea of some potentially important features for predicting hotel cancellations, such as historical behavioral patterns, and which machine learning models might be suited for cancellation prediction.

4 Dataset

This study uses the Hotel Booking Demand dataset (https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand) from Kaggle which contains reservation data from a city hotel and a resort hotel from a Portuguese hotel chain. The dataset has 119,390 rows and 32 columns and includes features on booking time, length of stay, number of people (adults, children, babies), and much more. One key feature is the reservation status, which can be used to determine whether a booking was canceled.

4.1 Preprocessing

Missing values for *children* were filled with 0s, and values for *country* were binned and mapped to *continents* for better generalization, as part of our feature engineering effort. Lastly, *company* and *agent* were re-coded to binary to indicate whether companies and agents were involved in the reservation process. We determined that ADRs (Average Daily Rate) that were $5000 or negative were outliers that did not seem reasonable in the context of the dataset.

Other outliers, while seemingly improbable, were not impossible to reason about, and therefore were kept as they still seemed to contain meaningful data. To preserve the significance and scaling of high-magnitude values and address asymmetrical distributions, logarithmic transformation was used to compress the scale of *adr* (average daily rate), *lead_time*, and *days_in_waiting_list* as they had right-skewed distributions in the several hundreds. The table in Appendix 3.2 documents the

cleaning and filtering steps applied to each column of the dataset to prepare the features for analysis.

Five columns were dropped as they were either related to the label, related with another feature, or provide no additional information for the prediction problem. Two categorical variables were combined into one so that the value is binary and potentially would provide better interpretation than either on its own. The final set of 26 features (Appendix 2.1), including the outcome, were selected to move forward with the next stage of data exploration and feature engineering.

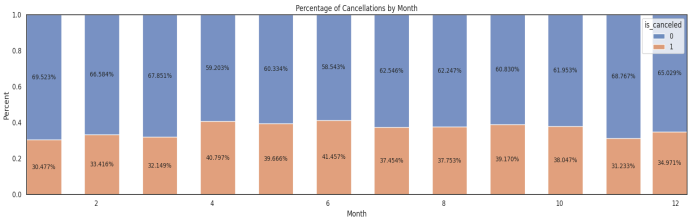
4.2 Train/Validation/Test Split

Post-processing, the final dataset had 119,388 rows with 26 columns (before encoding). The data was shuffled and stratified on the target variable to ensure balanced datasets. The columns were then encoded, creating 53 total columns:

Data Set	Split % (X and Y)	Rows (X and Y)	X Columns (After Encoding)	Y Column(s)
Train	60%	71632	53	1
Validation & Test	20% (each)	23878 (each)	53	1

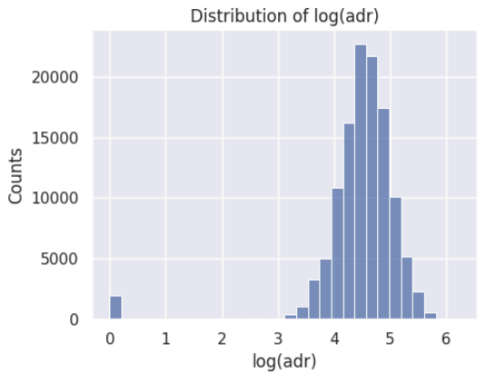
4.3 Exploratory Data Analysis

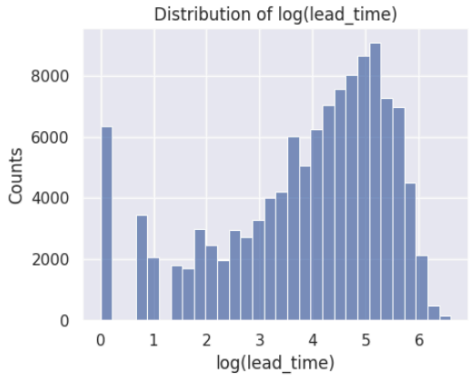
At first glance, monthly groupings of the data revealed stable cancellation rates between 30 - 40% throughout the year, with slightly less cancellations during months at the start and end of the year.

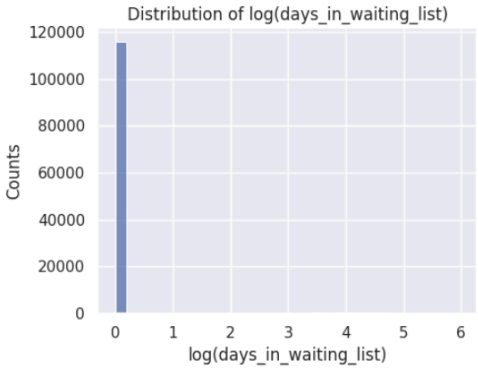


The distribution and boxplots of each feature was visualized to assess spread, data quality, and outliers (Appendix 4.1.1). The most common values of each feature were identified (Appendix 4.1.2) to highlight dominant patterns within the data.

In particular, the features *adr* (average daily rate), *lead_time*, and *days_in_waiting_list* were noted to potentially influence model performance greatly due to right-skewed distributions into the several hundreds. The values of these features were adjusted using log transformation.







Correlation heatmaps (Appendix 4.2 – 4.3) were used to track the top 5 negatively and positively correlated features to the target variable *is_canceled*. For categorical features, the correlation heatmaps were generated based on the one-hot encoded columns. These features were ranked by their correlation values (positive and negative as shown in the following tables:

Top 5 Negatively Correlated Features		
Feature	**Categorical**	**Correlation**
deposit_type_no_deposit	Yes	-0.478
total_of_special_requests	No	-0.23
required_car_parking_spaces	No	-0.20
market_segment_direct	Yes	-0.154
distribution_channel_direct	Yes	-0.152

Top 5 Positively Correlated Features		
Feature	**Categorical**	**Correlation**
deposit_type_non_refund	Yes	0.481
lead_time	No	0.29
is_reserved_room_type	No	0.25
market_segment_groups	Yes	0.222
distribution_channel_ta_to	Yes	0.176

From the data exploration and correlation heatmaps, deposit type seemed to be an important categorical feature in the model, followed by market segment. Among non-categorical variables, special requests and car parking were correlated with reservations that were less likely to cancel, whereas longer lead times and reservations that were not assigned their reserved room type were more likely to cancel. These features were used as a starting point for our initial models as they had the strongest correlation values.

5 Methods

Hyperparameter Tuning

For faster algorithms, we used GridSearch for hyperparameter tuning to systematically explore pre-defined hyperparameter combinations and identify the optimal configuration. However, this would not have been feasible for algorithms such as SVM, or neural networks, especially since they are more computationally intensive, and our dataset is not small. For these algorithms, we instead leveraged RandomSearch for hyperparameter tuning.

Baseline Results

We chose logistic regression to be our baseline model. It is a commonly used algorithm for binary classification tasks and works by applying the logistic function on the linear combination of input features to estimate the probability that an observation belongs to a particular class (in this case, whether a booking will be cancelled). It assumes a linear relationship between the feature and the target variables. We got approximately 82% and 81% accuracy for the train and test datasets respectively. Also, we got 82% precision for predicting cancellations. However, our recall was sitting at about 60% with a log-loss rate for both training and test at 0.41 and 0.42.

Improved Model Version 1: Random Forest Classifier

To address the limitations of our baseline model (logistic regression), we implemented a Random Forest classifier as one of our improved models. Logistic regression assumes a linear relationship between the features and the outcome, which may not hold true for our dataset. Random Forest is well-suited for capturing complex, non-linear interactions. It is an ensemble learning algorithm that constructs multiple decision trees during training. Each tree is trained on a random subset of the data (both rows and features), and the final prediction is determined by majority voting across the trees. This ensemble approach helps mitigate overfitting.

Random Forest is particularly effective when the decision boundary between classes is not clearly linear. We tuned hyperparameters such as number of trees (n_estimators), maximum depth of the trees (max_depth), minimum number of samples required to split an internal node (min_samples_split), the number of features considered for splitting (max_features), etc using a combination of grid search with 5-fold cross-validation and manual experimentation.

Improved Model Version 2: XGBoost

To further improve our model performance, we explored XGBoost for its robustness and ability to capture complex patterns in structured data. Unlike our baseline model, it builds trees sequentially where each new tree corrects the errors made by the previous one. It also includes regularization techniques that help reduce overfitting, which is especially useful when working with the volume of features and noisy data. Additionally, the built-in handling of missing values and support for early stopping, which helps enhance the flexibility during training.

We tuned several hyperparameters to optimize performance using a combination of GridSearch and manual adjustments. The key parameters include use of 200 trees to ensure model robustness and set the maximum depth of each tree to 15 to balance complexity with generalization. The parameters min_samples_split=10 and min_samples_leaf=4 were used to prevent overly deep trees that could lead to overfitting. Additionally, we applied class_weight='balanced' to account for class imbalance, giving more weight to cancellation class and allowing the model to improve sensitivity to minority class outcomes.

Improved Model Version 3: Neural Network

To improve upon the performance of the baseline model, we also implemented a Multi-Layer Neural Network. Neural networks are capable of capturing complex, non-linear patterns in the data through a series of interconnected layers and non-linear activation functions. This makes them suitable for problems like ours, where interactions between features are not easily captured by linear models.

Our neural network consists of an input layer, two hidden layers, and an output layer. Each hidden layer uses the ReLU activation function to introduce non-linearity, allowing the network to learn intricate patterns. The output layer uses a sigmoid activation function to generate a probability score for binary classification. We compiled the model with binary cross-entropy loss and optimized it using the Adam optimizer, which adapts the learning rate during training for faster convergence.

To avoid overfitting, we incorporated dropout regularization between the hidden layers, randomly dropping a portion of neurons during each training step. We also used early stopping to halt training if the validation loss did not improve for several consecutive epochs. Hyperparameters such as the number of units in each hidden layer, batch size, learning rate, and number of training epochs were tuned through manual testing and experimentation.

Additional model exploration and descriptions for SVM and LightGBM can be found in Appendix 5.1.

6 Experiments, Results, and Discussion

Validation Strategy

Our data was randomly split into train, validation, and test datasets and stratified on the target variable *is_cancelled* so that each set was balanced by the outcome variable. We leveraged the validation set to assess how well our models performed on unseen data before applying the models on the test data for a final evaluation.

Evaluation Metrics

We used the evaluation metrics, precision, recall, F1 score, ROC AUC, and PR AUC, to assess model performance. The metric we wanted to emphasize was precision, as we did not want to

incorrectly flag guests who were committed and would not cancel, as this could cause poor guest experience and loss of loyalty. However, the other metrics were also important as we still wanted to capture cancellations when they happened, even if some are missed to maintain precision.

Model Comparison Strategy

To compare the models, we ranked them on the following evaluation metric (ordered from most to least important):

1. Precision: to avoid falsely flagging loyal guests
2. PR AUC: how reliable are the model's cancellation predictions
3. F1- score: balance precision and recall and can help with optimizing overbooking
4. ROC AUC: how well does the model separate cancelled and non-cancelled guests

Train set results:

Model	Precision	PR AUC	F1-score	ROC AUC
Logistic Regression	0.81	0.84	0.72	0.78
Random Forest	0.84	0.89	0.76	0.90
XGBoost	0.90	0.96	0.86	0.97
Neural Network	0.92	0.97	0.89	0.98

Validation set results:

Model	Precision	PR AUC	F1-score	ROC AUC
Logistic Regression	0.82	0.84	0.72	0.78
Random Forest	0.86	0.90	0.79	0.93
XGBoost	0.86	0.91	0.8	0.93
Neural Network	0.84	0.87	0.75	0.91

Test set results:

Model	Precision	PR AUC	F1-score	ROC AUC
Logistic Regression	0.82	0.84	0.71	0.77
Random Forest	0.87	0.90	0.79	0.92
XGBoost	0.86	0.91	0.80	0.93
Neural Network	0.84	0.87	0.75	0.91

The full results table, including additional model results for SVM and LightGBM, can be found in Appendix 6.1.

Overfitting Concerns and Mitigation Strategies

In our modelling process overfitting was a potential issue, especially for more complex models like neural networks and XGBoost. We identified overfitting by observing a large gap between training and validation metrics (e.g., precision, F1 score, AUC). To mitigate this, we applied several strategies across different models: For Random Forest and XGBoost restricted tree depth (max_depth), the minimum number of samples required to split a node (min_samples_split), the minimum number of samples required at a leaf node (min_samples_leaf), and limiting the number of features considered at each split (max_features). The Neural Network model added dropout layers to randomly deactivate neurons during training, which prevents the network from becoming too reliant on specific paths. We also used L2 regularization (via kernel_regularizer) to penalize certain large weights and included batch normalization to stabilize and regularize training. Additionally, we added early stop training by monitoring the validation loss if performance stopped improving.

Performance Evaluation

Based on the results across all models, XGBoost consistently outperformed the others due to its ability to handle complex, structured data and capture nonlinear patterns effectively. Unlike the baseline Logistic Regression model, XGBoost builds trees sequentially, with each tree learning from the previous one's errors, enabling it to reduce both bias and variance. Its built-in regularization mechanisms also help control model complexity, preventing overfitting. This was particularly beneficial for our imbalanced dataset, as revealed by our EDA, where XGBoost's gradient boosting approach helped the model focus on hard to classify examples. Random Forest also showed strong performance through ensemble learning and variance reduction, but lacked the fine-grained boosting mechanism and regularization control that gave XGBoost an edge. Neural Networks, while capable of learning nonlinear relationships, were more sensitive to hyperparameter tuning and required larger datasets and training time to generalize as well as the tree-based models. Given its strong generalization to the test set, highest F1 and AUC scores, and robust handling of both data complexity and imbalance, XGBoost was selected as the final model as of this time.

7 Conclusion

Across all models, we observed measurable improvements over our logistic regression baseline, particularly in recall, F1 score, and ROC AUC. Logistic Regression, while interpretable and fast, struggled to capture non-linear relationships, resulting in a moderate F1 score (0.72) and lower ROC AUC (0.78). However, with XGBoost, we achieved a precision score of 0.86 with significant improvements across all metrics of F1 score (0.8), ROC AUC (0.93), and PR AUC (0.91), proving this to be our best model from the overall benchmark.

Limitations and Future Improvements

One of the primary challenges is that the dataset spans from 2015 to 2017, which limits its relevance to current market trends. The year variable was excluded to improve consistency and reduce noise by focusing on the cancellation predictions at the monthly level. The data also had a strong geographic skew

with over 90% of booking reservations coming from European customers. This imbalance, combined with high-dimensional inputs could have led to overfitting or distortion in the model performance. If time permitted, we would have liked to further explore additional models, ensemble methods, and collect more recent and richer data. Additional features, such as more hotels, seasonal periods, holidays, and other external drivers of demand, would have also further enriched our data and allowed us to apply other forms of analysis, such as panel or time series analysis.

8 Contributions

Emily Zhao - Github Notebook

Individual Work:

Milestone report write-ups on sections:

Data Preprocessing (Encoding Categorical Variables and Feature Selection), Exploratory Data Analysis and Visualizations (Correlation Analysis Between Features and Target Variables), Methodology (all sections except for preliminary results), Appendix.

Final report write-ups on sections:

Introduction, related work, hyperparameter tuning, evaluation metrics, validation and test results, future work, appendix

For notebook contributions:

Data cleaning and preprocessing, some exploratory data visualization, exploration of feature selection methods, exploration of methods to best split dataset, model exploration (LR, RF, XGBoost, SVM, Extra Trees)

Other:

Literature review for machine learning models best suited for the prediction problem

Kristen Lin - Github Notebook

Individual Work:

Milestone report write-ups on sections:

Data Description (Dataset Sizes and Splitting), Data Preprocessing (Handling Missing Values, Outlier Detection and Treatment, Feature Scaling), Exploratory Data Analysis and Visualizations (Feature Distributions, Correlation Analysis Between Features and Target Variables), Appendix.

Final report write-ups on sections:

Introduction, Dataset (Preprocessing, Train/Val/Test Split, Exploratory Data Analysis), Methods (Baseline Results, LightGBM - in Appendix 5.1), Experiments (Validation Strategy, Evaluation Metrics), Limitations and Future Work, Appendix.

For notebook contributions:

Basic EDA, filtering/processing, visualizations, and uploaded split datasets with one-hot encoded columns to Github. GridSearchCV and explored supervised models: Random Forest, XGBoost, and LightGBM.

Chad Vo - Github Notebook

Individual Work:

Milestone report write-ups on sections:

Motivation, Data Description (Dataset), Preliminary results, Appendix.

Final report write-ups on sections:

Methods and Models Descriptions/Discussions, Abstract, Performance Evaluation, Conclusions, Results and Discussion.

For notebook contributions:

First iteration of testing model using logistic regression library with all available features. Final supervised models of Random Forest, XGBoost, Decision Tree, and MLP. Also tested different engineering features, which are not included in the final selected features.

Karan Patel - Github Notebook

Individual Work:

Milestone report write-ups on sections:

Data Challenges.

Final report write-ups on sections:

Methods and Model Description, Results, Overfitting Concerns and Mitigation Strategies.

For notebook contributions:

Development and analysis of supervised ML models: Decision trees, random forest, Gradient Boost and Neural Network.

References

1. R. Z. Cao, W. Ding, X. Y. He and H. Zhang, "Data mining techniques to improve no-show forecasting," Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics, QingDao, China, 2010, pp. 40-45, doi: 10.1109/SOLI.2010.5551620. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5551620
2. Dunstan, J., Villena, F., Hoyos, J. et al. Predicting no-show appointments in a pediatric hospital in Chile using machine learning. Health Care Manag Sci 26, 313–329 (2023). https://doi.org/10.1007/s10729-022-09626-z

Appendix

https://docs.google.com/document/d/1iUSBADNApk72x3u2DbwFxIoVoMQBwkZmLHLod1cJLso/edit?tab=t.udac5fbtbau7