CIS520 Project: Hotel Cancellation Prediction

Yifei Li
School of Engineering and Applied Science
University of Pennsylvania
liyifei@seas.upenn.edu

Zhijian Yang
School of Engineering and Applied Science
University of Pennsylvania
zjyang@seas.upenn.edu

Abstract

The demand forecast is critical to the hospitality industry which always suffers from the market fluctuations. Compared to the traditional statistics and time series model used in hotel revenue management, the machine learning approaches have great potential to enhance the hotel demand prediction by analyzing the customer characteristics. We compared the performance of different machine learning models from classical techniques to deep learning architecture DeepFM, tuning out that a soft-voting estimator comprised of a tuned neural network, random forest and decision tree has an outstanding 97.52% accuracy. Besides, the impact of dataset imbalance and hyperparameter tuning have been explored and discussed.

Notebook links: (1) *Dataset Cleaning and Visualization* (2) *Basic Models Experiment* (3) *Advanced Models Exploration*

1. Motivation

In hospitality industry, the customer demands are seasonal and fluctuating for various reasons. The customer criteria such as travel frequency and service expectations plays a significant role in the hotel market. An accurate hotel demand prediction based on these customer features is valuable for revenue management by optimizing the room occupancy [9] while avoiding the risk of reputation damage caused by an excessive overbooking.

Machine learning models can seize both the breadth and depth of customer characteristics and have the potential to outperform the traditional hotel forecasting methods which are merely based on the date or several features. Besides, as a seasonal service industry, in hospitality, time-sensitive approaches such as classical time series model is meaningful to the demand prediction. Last, it's meaningful to see whether the model architectures from other fields such as recommendation system can be transferred to hotel demand prediction and work well.

2. Related Work

The old school forecast models of hotel demand can be put in three categories: historical booking models, advanced booking models and combined models, involving deeply in time series and classical regression [9].

As the emergence of machine learning, their applications in hotel industry rapidly grow up. Y. Zhang [12] applies multiple classical machine learning models in hotel demand forecast based on pricing, location, etc., ending up that these approaches especially SVM and random forest beat the traditional model by empirical study. Also, Sánchez-Medina *et al.* [8] explores the usage of neural networks cooperated with genetic algorithms to predict not only the cancellation rate but also the likely customer who may cancel the room. Also, this techniques can set the structural parameters to find an optimal solution while lowing the risk of being stuck in the local maxima. The necessity of collecting the customer's historical records for hospitality enterprises as key asset has also been addressed from a managerial point of view.

Besides, there are increasing deep learning deployments in hotel demand forecast. B. Zhang *et al.* [10] explores the potential of LSTM model integrating the internet search index on the hotel accommodation demand forecast, showing the robustness of this model by large and different datasets. Compared to its benchmark models, the artical's novel constructed forecasting method can largely simulate the overnigt customer's dynamic features and hereby optimizes the prediction. Further, Q. Zhang *et al.* [11] introduces a deep learning based dynamic pricing model which contains DeepFM, seq2seq and DNN frameworks for hotel revenue management. It can improve the prediction accuracy without a very complex feature engineering.

Last, for the methods, Kumari and Srivastava [5] completed a taxonomy for binary classification. A bunch of machine learning approaches like SVM, decision tree, and nueral network are mentioned and compared, though there is not a model with absolute advantage since the standard and requirement changes as the dataset differs.

3. Dataset

3.1. Introduction

In this project we use a dataset[1] provided by Nuno Antonio. It contains hotel booking information between July 1st, 2015 and August 31st, 2017 [1]. There are in total 119,390 bookings data from a city hotel and a resort hotel, with 32 features. The features are a mixture of 15 categorical variables and 17 numerical variables (Table 1), ranging from hotel type and country location to canceled status and arrival time.

Integer	*LeadTime, ArrivalDateYear, ArrivalDateWeekNumber, ArrivalDateDayOfMonth, StaysInWeekendNights, StaysInWeekNights, Adults, Babies, IsRepeatedGuest, PreviousCancellations, PreviousBookingsNotCanceled, BookingChanges, DaysInWaitingList, RequiredCarParkingSpaces, TotalOfSpecialRequests*
Float	*Children, Agent, Company, ADR*
Object	*ArrivalDateMonth, Meal, Country, MarketSegment, DistributionChannel, ReservedRoomType, AssignedRoomType, DepositType, CustomerType, ReservationStatusDate*

Table 1: Feature Data Types

Since the data performance between the resort hotel and city hotel is quite different, we decided to focus on the analysis of resort hotel's data for the purpose of this project. The resort hotel's dataset that we are using leaves us 40,060 booking data.

3.2. Summary Statistics and Visualization

The numerical data has a statistical description showed in Table 2.

The categorical data is comprised of customer characteristics and service types. For instance, Figure 1 shows the origin country percentage of customers and Figure 2 provides which market segment the customers come from.

Overall, as showed in Figure 3, the variables like *Adults*, *Children*, *StaysInWeekendNights*, *StaysInWeekNights*, *Meal*, *Country* and *AssignedRoomType* are clearly distributed differently corresponding to the status of *IsCanceled*.

[1]Source: `https://ars.els-cdn.com/content/image/1-s2.0-S2352340918315191-mmc2.zip`

Variable	Mean	MIN	MAX
ADR	94.95	-6.38	508
Adults	1.87	0	55
ArrivalDateOfMonth	15.82	1	31
ArrivalDateWeekNumber	27.14	1	53
ArrivalDateYear	2016.12	2015	2017
Babies	0.014	0	2
BookingChanges	0.29	0	17
Children	0.13	0	10
DaysInWaitingList	0.53	0	185
LeadTime	92.68	0	737
Previous...NotCanceled	0.15	0	30
PreviousCancellations	0.1	0	26
RequiredCarParkingSpaces	0.14	0	8
StaysInWeekendNights	1.19	0	19
StaysInWeekNights	3.13	0	50
TotalOfSpecialRequests	0.62	0	5

Table 2: Numerical Data Description

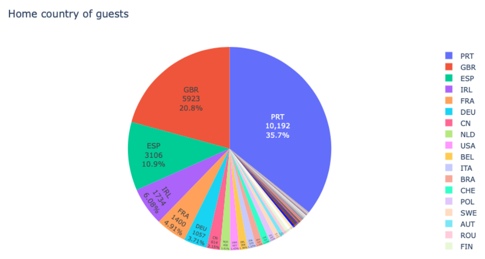


Figure 1: The Percentage of Customer Origin Country

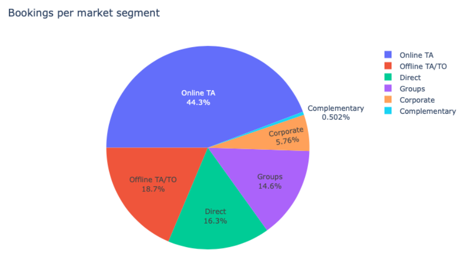


Figure 2: The Percentage of Customer Market Segment

3.3. Pre-Processing and Feature Engineering

3.3.1 Data Imputation

In our dataset, we have three features with null values showed in Table 3.

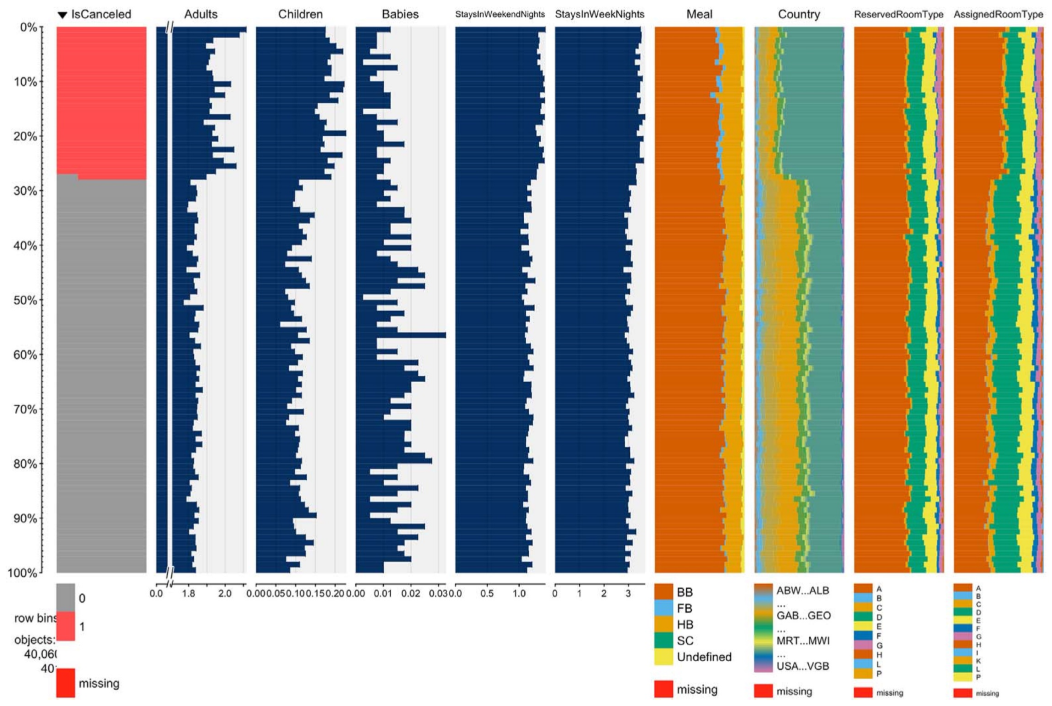


Figure 3: Dataset Visualization of All Observations

Feature	Number of Null	Percentage of Null
Country	464	1.16%
Agent	8209	20.49%
Company	36952	92.24%

Table 3: Distribution of Null

Because the null values in each feature stand for similar meaning, there is no need for regression in data imputation. We would just replace the nulls with particular integer or text instead.

For *Country*, the null values means that the booking either not having country information so we replace the null values with ”other” .

The null values in *Agent* imply that the booking does not come from a booking agent. Similar for *Company*, this feature implicate the ID of the company/entity that made the booking, and null value means that this is an individual booking. The data type for both *Agent* and *Company* is float, so we replaced the null values with 0 [1].

3.3.2 Category Numericalization

Table 1 describes the data type for each feature. To numericalize the categorical feature, we use different approaches:

- Label Encoder: For *ArrivalDateMonth*, which is originally month in text, we replace it by the numerical expression of month via label encoder.

- Timestampe: The *ReservationStatusDate* feature is originally object in format of ”yyyy-mm-dd” and we reconstruct it into two numerical features: *DayOfWeek* and *DayOfYear* due to the seasonal nature of hotel demand.

- One-Hot Encoder: For other categorical variables, we simply resort to one-hot encoding.

3.3.3 Feature Selection

For our project, we are trying to prediction the cancellation status of a booking. After accessing the data, we found that *IsCanceled* and *ReservationStatus* are completely correlated with each other. When cancellation status is 0 then reservation status would be *CheckOut* otherwise cancellation status would be 1. The feature importance generated by decision tree in Figure 4 proves our guess. As only one prediction variable is needed, we decide to drop *ReservationStatus* feature and use *IsCanceled* as a binary label target.

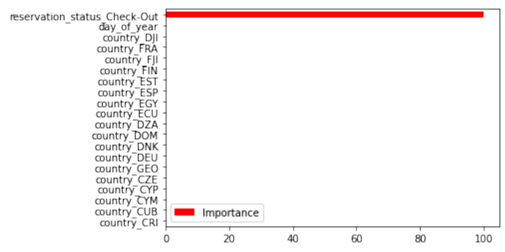


Figure 4: The Importance of Reservation Status

3.3.4 Feature Extraction

To enhance the feature description and richness, based on the domain knowledge, three new features are created:

- *IsFamily*: An binary indicator describing whether the hotel guests come as a family or not.

$$\textit{IsFamily} = \begin{cases} 1 & \text{if } \textit{Adults} > 0 \text{ and} \\ & [\,\textit{Children} > 0 \text{ or } \textit{Babies} > 0] \\ 0 & \text{Otherwise} \end{cases}$$

- *CustomerNumber*: The total number of customer.

$$\textit{CustomerNumber} = \textit{Adults} + \textit{Children} + \textit{Babies}$$

- *NightNumber*: The total number of stayin nights.

$$\begin{aligned} \textit{NightNumber} = & \textit{StaysInWeekendNights} \\ & + \textit{StaysInWeekNights} \end{aligned}$$

4. Problem Formulation

The feature set are processed as above by a bunch of different encoding and extraction methods. As a result, we have 195 number-only features which are convenient for model deployment and friendly for description.

Our goal is to apply different machine learning techniques to predict whether a booking of 40,060 would be canceled or not based on the selected feature set and analyze the importance of each feature in that prediction (See Table 4).

$\mathbf{y}$	$\mathbf{X}$
IsCanceled	194 Processed Features

Table 4: Statistically-Defined Problem

Since this is a binary classification problem, where the canceled status $y = 1$ denotes *IsCanceled* while $y = 0$ denotes *IsNotCanceled*, then for the cancellation prediction, we use the cross-entropy loss function

$$\mathcal{L} = -[y \log(p) + (1 - y) \log(1 - p)]$$

where p is the predicted probability generated by the corresponding rank of $\mathbf{X}$.

The models we're going to explore in the next section are from the classic classification models such as logistic regression and decision trees and also some paper-based advanced models such as ensemble and DeepFM.

5. Methodologies

5.1. Logistic Regression (Baseline)

The baseline model is set as the benchmark for the performance comparison. Since this is a binary classification problem and logistic regression can map all data points into a value between 0 and 1 by the sigmoid function, then it's used as the baseline model.

Package: sklearn.linear_model.LogisticRegression

5.2. Decision Tree

The decision tree can break down the problem like binary classification into a bunch of subsets with homogeneous values. The splitting procedure is helpful to show the importance of each feature and thus provide us some insights. Afterall, a big advantage of decision tree is its interpretability which is close to human-being's decision-making process.

Package: sklearn.tree.DecisionTreeClassifier

5.3. Random Forest

Random forest is an ensemble learning method which can be used for classification. It's comprised by a multitude of decision trees but without the cost of the overfitting prone and thus may have higher accuracy than decision tree.

Package: sklearn.ensemble.RandomForestClassifier

5.4. Extra Trees

Extra trees is also ensembled by decision trees. However, different from random forests's subsampling and optimal splitting points, extra trees uses the whole original sample and split nodes by random. In other words, extra trees has a higher degree of randomness but also keeps optimization. Therefore, extra trees may executes faster [3].

Package: sklearn.ensemble.ExtraTreesClassifier

5.5. Support Vector Machine

Support vector machine is a good general-purpose classification algorithm. It aims to find the beset decision boundary that splits a dataset into two or more classes by maximum margin. Despite the disadvantage of the $\mathcal{O}(n^2 p)$ complexity, it's still feasible in our problem given the relatively small scale of dataset.

Package: sklearn.svm.SVC

5.6. Neural Network

The neural network, stems from the biological neurons mimic, is useful in classification domain. The key of neural network is the extraction of the linear combination of the features, and then models the observation as a nonlinear function of the fitted features. Therefore, the neural network can be used as a multi-step regression. Further, by increasing the hidden layers, we can construct a deep learning model to dissect the features and solve the binary classification problems to a fairly high degree.

Package: sklearn.neural_network.MLPClassifier

5.7. XGBoost

XGBoost is an algorithm designed by Dr. Tianqi Chen that is considered one of the fastest implementation of gradient tree boosting. It it suitable for medium and small structure data and can perform parallel processing to increase speed. By pruning and regularization, the model also outperform other boosting models in preventing over-fit.

Package: xgboost.XGBClassifier

5.8. AdaBoost

AdaBoost is a meta-learning method that is initially created to increase the efficiency of binary classifiers and thus perfectly suits this problem [2]. The adaptive behavior allows it to focus on the mistake of weak classifier and then turn them into stronger ones. As a result, it can seize the depth of detail thoroughly.

Package: sklearn.ensemble.AdaBoostClassifier

5.9. Soft-Voting (Ensemble)

Soft voting is selected as our ultimate ensemble method. Whereas hard voting prefers a majority-take-all policy, soft voting can make use of each model more evenly by taking into account how certain each voter. In detail, after running the models above once, we can pick the most accurate ones to combine as a new multi-classifier estimator, where the voting weights are subject to the rank of accuracy.

Package: sklearn.ensemble.VotingClassifier

5.10. Deep Factorization-Machine

DeepFM is a neural network framework that combines the power of factorization machines (FM) for recommendation and deep learning for feature learning in a new neural network architecture (DNN). It focuses on both the wide and deep. In the DeepFM's neural network part, the activation function of the hidden layer uses ReLu and tanh for non-linear signal mapping, and the sigmoid function is used as the output function for Click Through Rate (CTR) estimation.

FM is to extract the high-order features and solve the combination of features given the systematic coefficient. Due to the application of hidden variable, FM can stay a good performance even if under the circumstance of hidden variable. The DNN is to extract the low-order features. They shared a weight matrix, that is, embedding layor. DeepFM can build a model based on the interaction of high-order features and low-order features without feature engineering [4]. The computation is showed as below.

$$\hat{y} = \text{sigmoid}\left(y_{FM} + y_{DNN}\right)$$

Since CTR is a variant of binary classification problem, and a deep learning network can solve the sparse feature dimension explosion, such as the feature *Country* in our dataset, caused by one-hot encoding, then DeepFM is suitable for this problem.

Package: deepctr.models.deepfm and also self-construct via pytorch

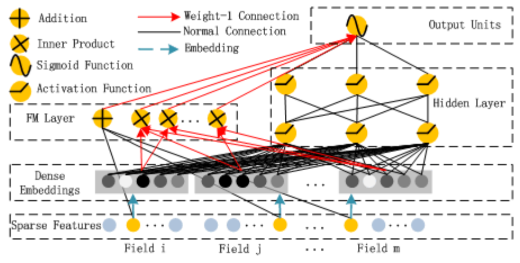


Figure 5: Wide and Deep Architecture of DeepFM

5.11. Others

We tried several other methods such as Naive Bayes Gaussian but they perform awfully in this problem and thus are omitted here.

6. Experiments and Results

6.1. Dataset Rebalance

Imbalance can cause the underrepresention of the minority and thus harm the accuracy. By Figure 3, the percentage of *IsNotCanceled* is about 30%, indicating slightly imbalance. The most traditional approaches to offset this impact is by resample, that is, undersample or oversample. However, both of them would bring new problems and a possible superior alternative is SMOTE. Their descriptions are as below.

- Undersample: rebalancing the dataset by duplicating the minority class at the cost of overfitting

- Oversample: rebalancing the dataset by removing the majority class at the cost of information loss

- SMOTE: rebalancing the dataset by synthesizing new class from the minority ones with the aid of KNN and thus avoid the costs above

We test these three techniques by the prediction accuracy of the logistic regression baseline model given the balanced training set (Table 5). All the rebalance methods successfully improve the prediction accuracy and among them, SMOTE's balanced dataset has the highest accuracy as expected. Therefore, we rebalance our training set by SMOTE before testing our models.

Rebalance Method	Balanced	Imbalanced
SMOTE	0.8401	
Oversample	0.8387	0.8028
Undersample	0.8379	

Table 5: Dataset Rebalance Accuracy

6.2. Metric

To evaluate the performance of each model, we're going to generate the ROC curve plot to visualize the differences with the help of AUC scores. Due to the requirement of the precise cancellation prediction in hospitality to optimize the over-booking strategy, the prediction accuracy of each model has the last say of the performance. Also, runtime would also be taken into account since the deployment speed is important especially in the industry.

6.3. Hyperparameter Tuning

Before deploying the models and comparing the results, we should first find the optimal hayperparameter combinations of the models. Here, we try to tune partial models which are easy to overfit and the hyperparameter has great impact.

6.3.1 Random Forest

Since it's well-known that random forest can offer good performance with its default settings so this algorithm is less tunable [7]. Therefore, We're going to pick just several parameters to see the difference. Because the number of trees is no risky to cause overfitting thus it should be set as large as possible [6]. The grid search is showed below where the optimal parameters are bolder.

- max_features: auto, **sqrt**, log2
- min_samples_split: **2**, 5, 10
- n_estimators: 100, 200, **500**

6.3.2 Support Vector Machine

Since in SVM, the overfit can be avoided by choosing appreciate kernel function and necessary regularization. It's important that we do tuning here to find the best pair of kernel and penalty. Hence, we decide to loop each kernel function plus a set of regularization parameter to see the best combination. The grid search is showed below where the optimal parameters are bolder, with an accuracy heatmap Figure 6.

- Kernel: **linear**, poly, rbf, sigmoid
- C (Regularization Parameter): 50, 10, 1.0, **0.1**, 0.01

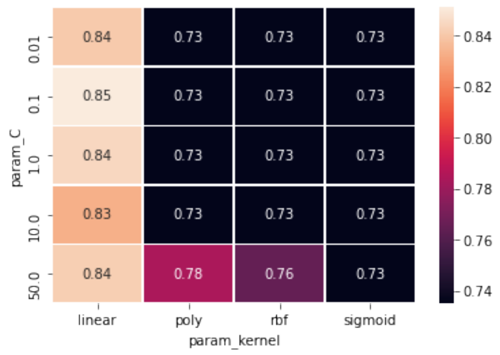


Figure 6: Accuracy Heatmap of SVM Tuning

6.3.3 Neural Network

The activation functioins are for nonlinearity creation which invloves the deep learning models later, where both logsitic and ReLu are classical. The hidden layer size determines the complexity of model and thus a larger size may improves the accuracy. The grid search is showed below where the optimal parameters are bolder.

- alpha: 1, 0.1, 0.01, **0.001**
- hidden_layer_sizes: (50, 50, 50), **(100, 100)**
- solver: **adam**, sgd
- activation: **logistic**, relu

6.4. Basic Model Implementation

Due to the time limitation, it's not possible to tune all the models. However, we can pick the hyperparameter candidates as suitable as possible by the prior knowledge of other binary classification problems samples. Also, the tuned parameters above give us some insights. For instance, we set the n_estimators as 500 in other tree family algorithms too due to their similarity with random forest.

During the empirical study, some models need a special handling of the training set and/or test set. For instance, as a gradient descend based algorithm, neural networks prefers the training set to be normally distributed data as a Gaussian with 0 mean and unit variance and thus a standardization is necessary. Another example is XGBoost, which requires the column number of training set and test set is the same. Hence, when deploying it to the balanced training set, we rebalance the test set as well.

By training each model excepts soft-voting and DeepFM and make a prediction on the test set, we generate an overlapped ROC curve with the AUC score of each model (Figure 7) for better visual comparison. Besides, we measure the execution time displayed as histogram in Figure 10. A summary metric table is showed in Table 6 which is comprised of the AUC score, accuracy score and runtime of each model, grouped by the imbalanced and balanced dataset. We will discuss it in the next section from different perspectives.

6.5. Advanced Exploration

6.5.1 Soft-Voting (Ensemble)

After completing the basic models, we pick the top three models with the highest accuracy in balanced training set as showed in Table 6. Next, them are ensembled as a new estimator by soft-voting technique. The weight of their votes is descending based on the accuracy of the imbalanced training group. In this case, we combine tuned neural network, tuned random forest and decision tree with the weight ratio 2:2:1 because while tuned neural network has a very close accuracy as tuned random forest, decision tree has a relatively worse performance. The accuracy we got is 0.9752, higher than all the single classifiers we implemented so far.

6.5.2 Deep Factorization-Machine

We first use pytorch to construct a simply DeepFM network to test a subset with partial feature and the training loss per epochs is showed in Figure 8. As the rise of training round, the parameters of model keep updating and therefore the loss value is decreasing. After about 40 rounds, the loss value remains nearly the same which indicts that the model begins to perform at the stable level and the efficiency of training is small after 40th round.

But since the DeepFM requires a special pre-processing of dataset to separate them as sparse and dense feature dictionaries, we resort to the deepctr package but unfortunately haven't completed it yet due to the time limitation and colab computing power constraint. Therefore, we cannot fully explore the potential of this state-of-the-art recommendation system algorithm and see whether it can be successfully transferred to another binary classification problem like the hotel demand.

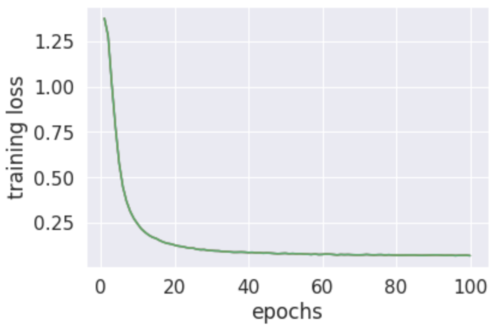


Figure 8: DeepFM: Training Loss vs Epochs

7. Conclusion and Discussion

7.1. Feature Importance and Relationship

With the help of decision tree, we get the feature importance (Figure 9).

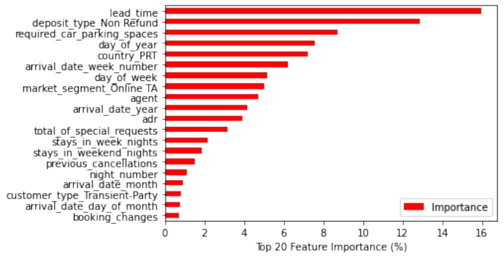
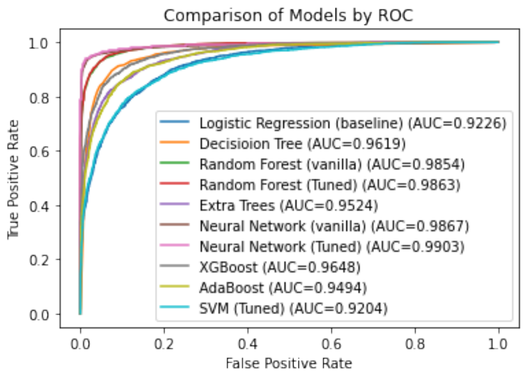


Figure 9: Feature Importance

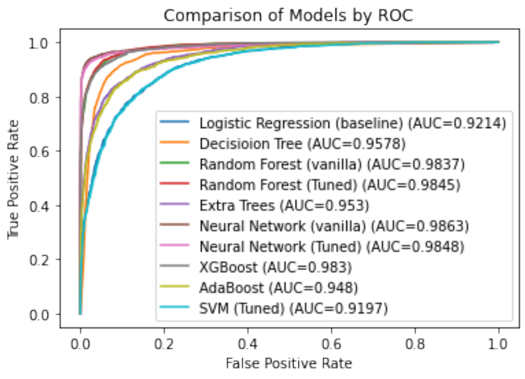
Although we cannot make an assertion of the causality from the feature importance, they provide us the pattern between the features and cancellation status. We will walk through a tip of them to guess the possible reasons behind.

1. *LeadTime*, the number of days that elapsed between the entering date of the booking into the PMS and the arrival date, is the most significant feature, with amost 15% importance. It makes sense since the longer the duration before arrival, the larger the possibility that the guests may cancel the booking due to unexpected factors.

2. The feature of the second importance is *DepositType-NonRefund*, it's probably because the existance of refund could encourage the customers to cancel the booking without the cost of losing money.



(a) Imbalanced



(b) Balanced

Figure 7: Basic Models Comparison by ROC

Model	AUC	AUC*	Accuracy	Accuracy*	Runtime(s)	Runtime(s)*
Neural Network (Tuned)	0.9885	0.9889	0.9725	0.9672	209.7754	253.2274
Neural Network (vanilla)	0.9868	0.9833	0.9705	0.9614	132.9359	125.3403
Random Forest (Tuned)	0.9863	0.9848	0.9467	0.9441	32.1794	54.7000
Random Forest (vanilla)	0.9853	0.9839	0.9458	0.9447	6.3899	11.2010
Decision Tree	0.9619	0.9555	0.9217	0.9146	0.5261	1.3537
XGBoost	0.9648	0.9833	0.9096	0.9308	13.4476	20.0968
AdaBoost	0.9494	0.9489	0.8938	0.8949	47.5919	102.2639
Extra Trees	0.9527	0.9527	0.8812	0.8922	27.0275	44.5968
SVM (Tuned)	0.9204	0.9199	0.8665	0.8326	989.7524	2564.1543
Logistic Regression (baseline)	0.9226	0.9218	0.8663	0.8361	7.4758	7.3667

Table 6: Basic Models Performance Comparison. Sorted in a descending order by the accuracy of imbalanced training, where the asterisk ($\star$) denotes being trained by balanced dataset.

3. Next, *RequiredCarParkingSpaces* is important. It is refer to the customers who are likely driving the car or plan to drive the car to the hotel. Their travels are more stable because they don't worry much about the weather condition and public traffic unlink the customers who go there by bus or plane. A group of them might already on the road when booking the hotel and thus less likely to cancel.

4. The feature *DayOfYear* is always important considering the seasonality of hospitality. It also shows that compared to other timestamps such as *DayOfWeek*, *DayOfMonth*, *Month* and *Year*, this feature has a much greater influence probably because the annual events such as holidays are more regular predictable.

Obviously, by the help of the important features we mentioned in graph and above, the hotel can predict the cancellation likelihood of a guest much easily. Moreover, the hotel can adjust the business strategy by these feature-based insights. For example, since the *DayOfYear* is a significant indicator, the hotel can make their overbooking policy more aggressive during the off season to make the best use of their rooms. Also, when seeing a customer with a reuest of car parking spaces, the hotel should predict that this customer is very likely to show-up and can begin the welcome procedure more confidently.

7.2. Model Performance and Optimization

As showed in ROC Comparison (Figure 7) and Accuracy Table (Table 6), nearly all the basic models we selected perform equal to or better than the performance of the logistic regression baseline. In the imbalacned training, the best ones without the vanilla are tuned neural network (0.9725), tuned random forest (0.9467) and decision tree (0.9619).

A soft-voting ensemble estimator combined with these three classifiers has an outstanding accuracy 0.9752, outperforming any single of them. It shows the success that a proper comprehensive usage of models have the potential of exceeding the single classifier. This time we assigned the weight by ratio 2:2:1 by the scale of accuracy difference. However, this accuracy is only 0.0025 larger than the best single classifier, tuned neural network. In the future, the research can explore and compare different combination of the weight ratio to see whether weighting strictly based on the accuracy of single classifier is a good idea and whether a better ensemble can improve the accuracy significantly further.

The tuning technique grid search improves the accuracy but not that much. Both in neural network and random forest, we can see a trivial enhancement after tuning when being trained by the imbalanced dataset. However, they're worse than the vanilla ones in the balanced training. It's possible that because the vanilla we use here already has a reasonable default parameter based on the prior experience. It requires us to do a much larger set of hyperparameter combination further to justify this assertion when the time is allowed.

As showed in Figure 10, the execution time of support vector machine is extremely high due to its $\mathcal{O}(n^2p)$ complexity, whereas almost other models finish the implementation within 1-2 minutes. While SVM is suitable for classification, it needs some performance optimization when being deploying on the large size dataset.

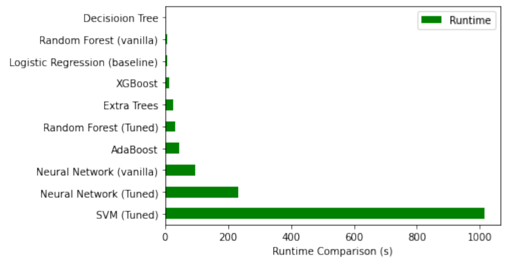


Figure 10: Runtime of Each Model

The difference between imbalanced and balanced dataset is subtle and it's not rare that the imbalanced one even has a higher accuracy than the balanced's. Because rebalance reduces the possibility of overfitting, it makes sense that the classifier trained by balanced training set has lower prediction accuracy.

References

[1] Nuno Antonio, Ana de Almeida, and Luis Nunes. Hotel booking demand datasets. *Data in Brief*, 22:41 – 49, 2019.

[2] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[3] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Language*, Apr 2006.

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction, 2017.

[5] Roshan Kumari and Saurabh Kr Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017.

[6] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest?, 2017.

[7] Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3), Jan 2019.

[8] Agustín J Sánchez-Medina, C Eleazar, et al. Using machine learning and big data for efficient forecasting of hotel booking cancellations. *International Journal of Hospitality Management*, 89:102546, 2020.

[9] Larry R Weatherford and Sheryl E Kimes. A comparison of forecasting methods for hotel revenue management. *International journal of forecasting*, 19(3):401–415, 2003.

[10] Binru Zhang, Yulian Pu, Yuanyuan Wang, and Jueyou Li. Forecasting hotel accommodation demand based on lstm model incorporating internet search index. *Sustainability*, 11(17):4708, 2019.

[11] Q. Zhang, L. Qiu, H. Wu, J. Wang, and H. Luo. Deep learning based dynamic pricing model for hotel revenue management. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 370–375, 2019.

[12] Yueqian Zhang. *Forecasting Hotel Demand Using Machine Learning Approaches*. PhD thesis, 08 2019.

Acknowledgment

The authors especially thank the University of Pennsylvania CIS520 Machine Learning instructors, Dr. Ungar and the TAs, for their advice and support throughout this project. We would also like to show our gratitude to Dr Antonio, Dr. Almeida and Dr. Nunes, who provide this dataset.