Machine Learning Techniques Final Project

HAO-CHENG LO, D08227104
TZU-YANG CHUANG, B06703098
WENFENG CHENG, D09944008

1 INTRODUCTION

The goal of this project is to build a daily revenue prediction system. The inputs of the system include features from the data of reservation, and the output will be a 10-classes revenue level. More precisely, an income of a fulfilled request will be the average rate of the room (ADR) multiplied by the number of days that the customer is going to stay. The daily revenue is the sum of all incomes from the same day. This number will then be quantized into 10-classes label (Figure 1).

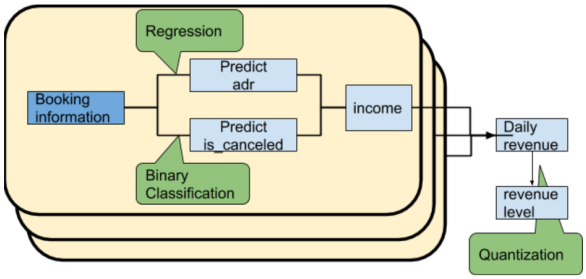

Fig. 1. The proposed income level prediction system includes 3 models. First, we used a regression model to predict the ADR. Second, a binary classification model is leveraged to predict if the booking was canceled. We then use the results to compute the predicted daily revenue. Finally, a quantized model is applied to map the revenue number into a 10-class income level.

2 DATA PREPROCESSING

Since features of booking information included continuous and categorical variables, we abided by the following encoding procedures: First, we dropped irrelevant or overlapping features, including *ID*, *arrival-date-year*, *reservation-status*, and *reservation-status-date*. Second, we encoded categorical features as one-hot-vector, including *hotel*, *arrival-date-month*, *meal*, *country*, *market-segment*, *arrival-date-year*, *distribution-channel*, *reserved-room-type*, *assigned-room-type*, *deposit-type*, *agent*, *company*, and *customer-type*. It is worth noting that since *country* had large number of categories, we encoded such variable as 13 categories (top 12 counts of country in train data and others). In the end, total features of 92 were served as predictors of adr and cancellation.

3 NEURAL NETWORK

3.1 Overview

Neural network has become the state of the art approach in several areas, including computer vision, natural language processing. So it is not surprised that we want to leverage the power of neural network to predict the hotel revenue level. Considering that we need to predict both cancellation and ADR with same features, in here, we proposed a multi-head classification model. As shown in Figure 2, this model includes a multilayer perceptron (MLP) encoder, and two separated prediction head. The classification head is used to predict the cancellation, and the regression head is used to predict the ADR. Surprisingly, while we are able to achieve 93.8% training accuracy for cancellation prediction, we are not able to get a reasonable performance for ADR. In most of the hyperparameter setting, the model try to predict average ADR for all of the bookings.

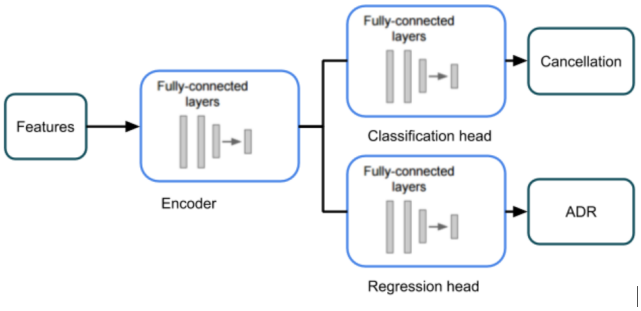

Fig. 2. The proposed multi-head approach include a encoder and 2 separated prediction head, the classification head for cancellation and the regression head for ADR.

3.2 Tuning Strategy

3.2.1 Model Structure. For a neural network model, there are lots of different hyperparameters we can try in model structure, including the number of layers, the hidden unit size, activation function. In here, we use ReLU as our activation function and optimize the layer number and hidden unit size. To make it simple, we first optimized the hidden unit size, we chose 256 for the encoder and 64 for the prediction head. After fixing the hidden unit size, we then tried to optimize the layer number. In most of the experiments, deeper model can achieve lower training error, but not necessarily improve the prediction quality on test set. We finally chose 3 layer encoder combined with 2 layer pr edition head. The detail results can be found in Figure 3.

3.2.2 Optimizer Hyperparameters. While optimizer is the key piece that enables Machine Learning to work for input data, it also includes a lot of adjustable options. In this project, e use Adam [Kingma and Ba 2014] as our Optimization algorithm. Another important parameter would be the learning rate, we start from 0.01, a 0.95 learning rate decay also applied for each epochs.

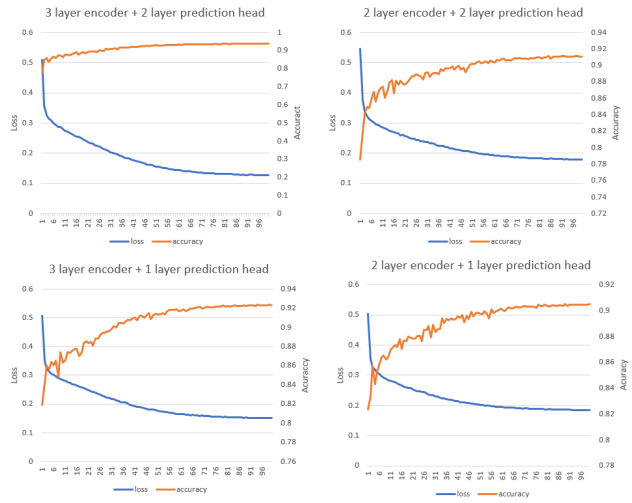

Fig. 3. Comparison for different combinations of layer numbers.

3.2.3 Dropout. Last but not the least, while neural network is a very powerful approach, without sufficient data, over-fitting will be a huge problem. In here, we applied dropout to avoid the over-fitting issue. During the training phase, a random set of neurons will not be considered during a forward or backward pass. When we change the dropout rate to 0.2, while the train error increase 1.89%, we are able to improve the test performance.

4 RANDOM FOREST

4.1 Environment and Packages

- Programming environment: *Python 3.7.3.*
- Packages: *sklearn, numpy, pandas.*

4.2 Overview

In our understanding, the prediction of revenue labels each day involves the predictions of whether the reservation requests were canceled, and the adr for each of the requests. After taking a further look at the training data, we found that most of the features are categorical features, and Random Forest, as a powerful tree-based model, is one of the most suitable candidates in this case. Therefore, for the prediction of cancellation (classification problem) we used `RandomForestClassifier` to build the model, and `RandomForestRegressor` for predicting adrs.

We performed several experiments to test the hyperparameters and the effect each hyperparameter has on our predicting outcomes. In the following sections, we will demonstrate how we tuned the hyperparameters, the results corresponding to the experiments, and the predicted outcomes.

4.3 Tuning Hyperparameters

4.3.1 n_estimators. This hyperparameter represents the number of trees in Random Forest. Its searching range $\in \{100, 200, 500, 1000\}$.

Table 1. Performances of different max_features

max_features	Training Data Score	Testing Data Score
auto	93.97%	82.81%
log2	93.41%	80.43%
sqrt	94.24%	81.59%

4.3.2 max_features. This hyperparameter represents the number of features to consider for the best split. We did an experiment on the three options stated in sklearn document, and found that there is no improvement in the performances from changing the default option. We therefore go for the default "auto" option.

4.3.3 max_depth. This hyperparameter sets a limit to the maximum number of depth in the trees. Its searching range is set to $\in \{\text{None}, 100, 200, 300\}$.

4.3.4 min_samples_split. This hyperparameter limits the minimum number of samples required to split an internal node. Its searching range is set to $\in \{2, 4, 6\}$.

4.3.5 tuning strategy. We used 3-fold cross validation to measure the performances of the combination of hyperparameters. As shown in Table 2, the best combination of hyperparameters we found is `n_estimators` = 100, `max_depth` = 200, `min_samples_split` = 2.

Table 2. Partial Results of Cross-Validation Error

n_estimators	max_depth	min_samples_split	CV error
100	None	4	11.70%
	None	6	14.02%
	100	6	10.97%
	200	2	**6.06%**
	200	4	6.07%
	300	2	16.04%
200	None	2	26.24%
	100	4	**10.85%**
	200	2	21.20%
	200	4	15.40%
	300	4	16.75%
	300	6	17.79%
500	None	4	17.91%
	100	2	17.91%
	200	6	**16.58%**
	300	2	21.57%
	300	4	23.74%
	300	6	17.90%
1000	None	4	20.90%
	100	2	21.20%
	100	4	**19.65%**
	100	6	22.72%
	200	4	20.06%
	300	4	20.57%

Table 3. Performance of Random Forest (cancellation)

Score Type	Score Value
Training Data Score	99.44%
OOB Score	89.25%
Testing Data Score	89.55%

Table 4. Performance of Random Forest (adr)

Score Type	Score Value
Training Data Score	93.97%
OOB Score	69.20%
Testing Data Score	82.81%

4.4 Performance

By training `RandomForestClassifier` and `RandomForestRegressor` with the best hyperparameters we derived from the searching process mentioned above, we can have the performances of these two models. It can be seen from Table 3 that the testing score of `RandomForestClassifier` is 89.55%, while from Table 4 we can see a testing score of 82.81% in `RandomForestRegressor`.

5 EXTREME GRADIENT BOOSTING

5.1 Environment and Packages

- Programming environment: *Python 3.7.3.*
- Packages: *sklearn, xgboost, hyperopt.*

5.2 Overview

Given its unprecedented level of predictive performance in numberless competitions, such as Kaggle and KDDCup, extreme gradient boosting (XGBoost), as a state-of-the-art tree-based machine learning model, has drawn much scholarly and pragmatic attention [Chen and Guestrin 2016]. Accordingly, we employed XGBoost to conquer the current task, performing the following procedures: First, *data preparation.* After data preprocessing, features and its corresponding labels (adr, cancellation) were split into two statistically equivalent datasets, namely, training data and validation data. Next, *model training and hyperparameters tuning.* `XGBoostRegressor` was utilized on training data for predicting adr; `XGBoostClassifier`, for predicting cancellation. The final optimal model of adr or cancellation was selected via Bayesian optimization process and experiments on early stopping, both of which chose the best combination of hyperparameters that had the lowerest E_{val}. Last, *label prediction.* Through our XGBoost "optimal" models, the labels of testing data were predicted; overall performance (i.e. public score and private score) was computed.

5.3 Hyperparameter Tuning Strategy

5.3.1 Bayesian Optimization. Facing myriad combinations of hyperparameters XGBoost involving, we firstly adopted Bayesian optimization process, an effective hyperparameter tuning paradigm based on *a prior* distribution of objective parametric function and loss functions of optimal sequences of validation metrics, for selecting the best model [Bobak Shahriari and de Freitas 2015]. Below describes hyperparameters considered that Bayesian optimization explored over.

- **General Parameters**. `eta` defines the step size at each iteration while moving toward a minimum of a loss function; its searching range $\in \{0.1, 0.01\}$.
- **Tree-specific Parameters**. `min_child_weight` defines the minimum sum of weights of all observations required in a child; its searching range $\in \{20, 21, \ldots, 30\}$. `max_depth` defines the maximum depth of a tree; its searching range $\in \{30, 31, \ldots, 50\}$. `gamma` specifies the minimum loss reduction required to make a split; its searching range $\in \{0.1, 0.2, 0.3\}$. `subsample` denotes the fraction of observations to be randomly samples for each tree; its searching range $\in \{0.6, 0.7, \ldots, 1\}$. `colsample_bytree` denotes the fraction of columns to be randomly samples for each tree; its searching range $\in \{0.5, 0.8, \ldots, 0.9\}$.
- **Regularization Parameters**. `lambda` defines L2 regularization term on weights, its searching range $\in \{10^0, 10^{-1}, \ldots, 10^{-6}\}$. `alpha` defines L1 regularization term on weights, set on 0.1.

5.3.2 Experiments on Early Stopping. Early stopping is a common regulation strategy for adverting overfitting when XGBoost model is trained [Zhang and Yu 2005]. Specifically, `early_stopping_rounds` specifies maximal rounds that validation error is not significantly decreasing. After selecting the best model via Bayesian optimization, we considered and experimented four `early_stopping_rounds` candidates $\in \{1, 2, 5, 10\}$ on the best model. We then selected the model having lowerest E_{val} as our final optimal model.

5.4 Predicting adr

To predict adr, we applied `XGBoostRegressor` on the training data and used the metric of 1−variance explained (R^2) and root mean square error (RMSE) of validation data as E_{val} for choosing the best model.

5.4.1 The Model Selected from Bayesian Optimization. After 100 evaluations of Bayesian optimization, the best model was with the hyperparameters that `n_estimators` = 1000, `colsample_bytree` = 0.7, `min_child_weight` = 27, `max_depth` = 38, `gamma` = 0.2, `subsample` = 0.8, `lambda` = 10^{-3}, `alpha` = 0.1, and `eta` = 0.01. Such model had R^2 = 84.20%.

5.4.2 The Optimal Model Selected From Experiments on Early Stopping. Experiments on four values of early stopping rounds were conducted. Table 5 presents the R^2 between predicted value and real value of validation data on models without and with four values of early stopping rounds. The corresponding learning curves are shown in Figure 4. The results indicate that the model with `early_stopping_rounds` = 10 serves as the optimal model (R^2 = 84.90%).

5.4.3 Feature Importance. Figure 5 indicates top 15 features important in the optimal model for predicting adr. Specifically, the importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance is.

Table 5. R^2 (adr) and accuracy (cancel) of Early Stopping (ES) Experiments

ES rounds	R^2 (adr)	accuracy (cancel)
Without ES	84.20%	88.01%
ES rounds = 1	84.43%	88.43%
ES rounds = 2	84.78%	84.44%
ES rounds = 5	84.87%	84.66%
ES rounds = 10	84.90%	84.66%

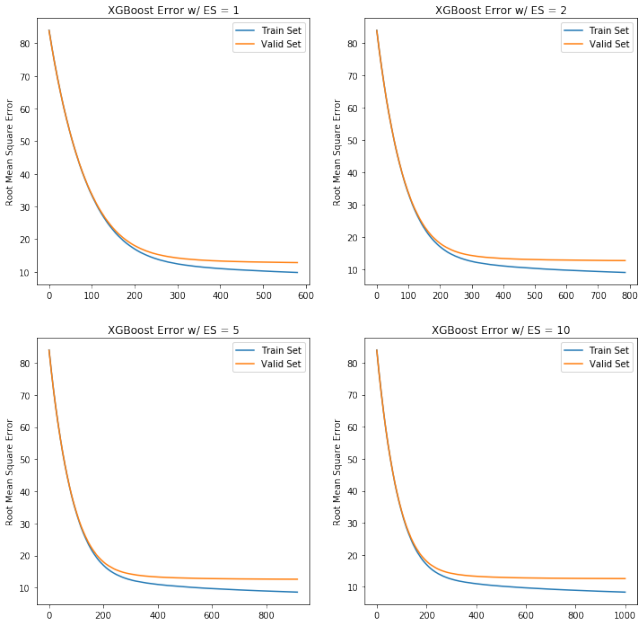

Fig. 4. Learning Curves of Early Stopping (ES) Experiments for adr

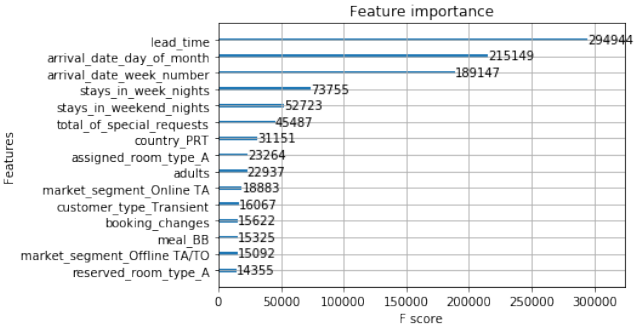

Fig. 5. Top 15 Important Features for Predicting adr

5.5 Predicting cancellation

To predict cancellation, we applied `XGBoostClassifier` on the training data and used the metric of 1−accuracy and logloss of validation data as E_{val} for choosing the best model.

5.5.1 The Model Selected from Bayesian Optimization. After 100 evaluations of Bayesian optimization, the best model was with the hyperparameters that `n_estimators` = 1000, `colsample_bytree` = 0.6, `min_child_weight` = 23, `max_depth` = 44, `gamma` = 0.1, `subsample` = 1.0, `lambda` = 10^{-1}, `alpha` = 0.1, and `eta` = 0.1. Such model had accuracy= 88.01%.

5.5.2 The Optimal Model Selected From Experiments on Early Stopping. Experiments on four values of early stopping rounds were conducted. Table 5 presents the accuracy between predicted value and real value of validation data on models without and with four values of early stopping rounds. The corresponding learning curves are shown in Figure 6. The results indicate that the model with `early_stopping_rounds` = 5 serves as the optimal model (accuracy = 88.66%).

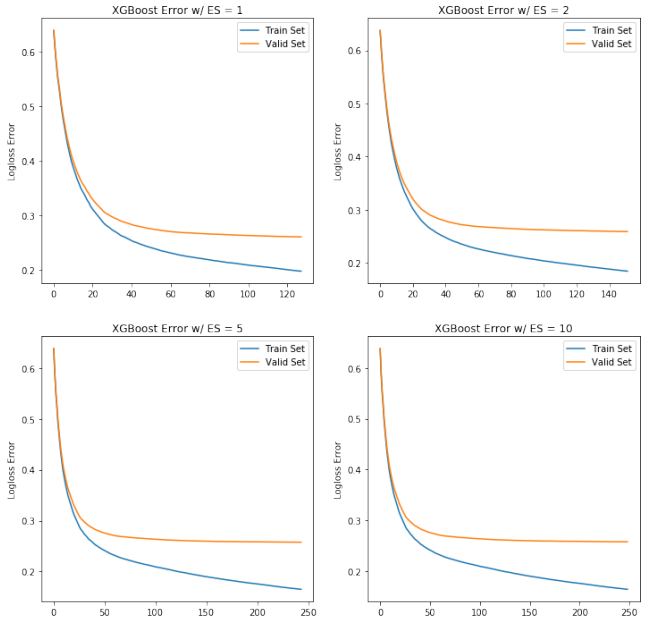

Fig. 6. Learning Curves of Early Stopping (ES) Experiments for cancel

5.5.3 Feature Importance. Figure 7 indicates top 15 features important in the optimal model for predicting cancellation.

5.6 Overall Performance

As aforementioned, to get the labels (i.e., revenue level of each day) of testing data, we need to calculate income of each reservation in advance through its predicting adr and cancellation. In this section, we predicted adr via our optimal XGBoost regression model; cancellation, via our optimal XGBoost classification model. The overall performance of testing data measured by MAE was public = 0.421 and private = 0.428.

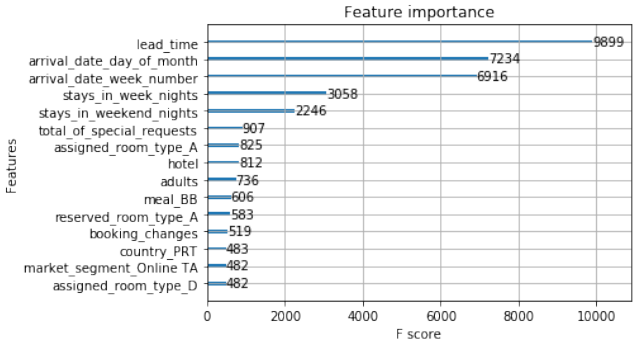

Fig. 7. Top 15 Important Features for Predicting cancel

6 COMPARISON

6.1 Efficiency

- *XGBoost*: implemented in Python language and executed the program on a PC with an Intel Core i5-8400 CPU of 2.80 GHz frequency and 16 GB memory under Windows 10 64 bit OS. It took about 4 minutes to train an XGBoost model.
- *Random Forest*: implemented in Python language and executed the program on a PC with an Intel Core i5-7200U CPU of 2.50 GHz frequency and 4 GB memory under Windows 10 64 bit OS. It took about 4 minutes to train an Random Forest model.
- *Neural Network*: implemented in Python language and executed the program on a laptop with an Intel Core i7-1065G7 CPU of 1.30 GHz frequency and 16 GB memory under Windows 10 64 bit OS. Each epoch took about 15 secs with pytorch CPU mode.

6.2 Scalability

- *XGBoost.* To achieve scalable learning, XGBoost enables out-of-core computation. Specifically, XGBoost divides the data into multiple blocks and utilizes independent threads to compute each block in parallel. Thus, XGBoost has been depicted as a highly scalable end-to-end tree boosting system [Chen and Guestrin 2016].
- *Random Forest.* Random Forest has good scalability due to its bootstrapping mechanism(sampled with replacement).
- *Neural Network.* With stochastic gradient decent, neural network model can be easily apply on million size or even trillion size data.

6.3 Popularity

All of the three approaches we adopted are popular in modern machine learning world. Benefited from the incredible model capacity, neural network has become the state of the art approach for lots of area. And it is becoming even more popular due to the increasing needs for A.I applications. Likewise, Random Forest is a classic and powerful model. As for Extreme Gradient Boosting, given its unprecedented level of predictive performance in numerous competitions, it has become one of the most popular models, especially for small scale dataset.

6.4 Interpretability

- *XGBoost and Random Forest.* Both of XGBoost and Random Forest are tree-based models, so they both have relatively high interpretability since the predictions can be easily followed by walking through the path on each tree. On the other hand, predictions from neural networks usually pass through multiple layers of computation, which make it almost like a black box.

7 CONCLUSION

In this project, we leverage random forest, xgboost, and neural network to predict the hotel revenue level. We also try to ensemble intermediate results from different models. By combining the ADR prediction from XGBoost and cancellation prediction from random forest, we achieved 34.2% error rate on public dataset and 37.3% error rate on private dataset. We also did a comparison among these three approaches with considering efficiency, scalability, popularity and interpretability. If a single approach need to be chosen for this task, our answer will be XGBoost. While neural network approaches dominate huge amount of tasks, according to our experiment results, it's actually the worst model for this task. Which shows that there's no one best model for all tasks and scenarios. We need to choose our approach carefully according to the task, the scenario and even the data behaviors.

A CONTRIBUTIONS

- d08227104: xgboost, datapreprocessing, evaluation and comparison
- b06703098: random forest, evaluation and comparison
- d09944008: neural network, evaluation and comparison

REFERENCES

Ziyu Wang Ryan P. Adams Bobak Shahriari, Kevin Swersky and Nando de Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. In *Proceedings of the IEEE*. 148–175.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Tong Zhang and Bin Yu. 2005. Boosting with early stopping: Convergence and consistency. *The Annals of Statistics* 33, 4 (2005), 1538–1579.