Machine Learning Techniques Project

Tzu-Hsien Tsai, Ji-Yun Tu, Yun-Ping Lin
B08902139, B08902071, B08902075

January 19, 2021

1 Introduction

In this report, we proposed three models for the revenue prediction problem. Each model consists of three submodels. The first two predict ADR and the probability to cancel of each reservation independently. The last one predicts the label of daily revenue according to the original dataset and the output of the former two models.

Below is a brief glimpse at the submodels we used in each model.

Model	Cancellation	ADR	Label
Model 1	Neural Network	Gradient Boosting (DART)	Decision Stump
Model 2	Neural Network	Random Forest	Decision Stump
Model 3	Neural Network	Neural Network	Recurrent Neural Network

Table 1: Submodels in each model

2 Data Preprocessing

1. **Drop `ADR` and `IS_CANCEL`**
2. **Add Features**:

   We add four additional features, with the first two being numerical and the last two being one-hot.

Features	Description
`stays_in_total`	`stays_in_weekend_nights` + `stays_in_week_nights`
`children_in_total`	`children` + `babies`
`night_type`	Stay on weekend nights, on weekday nights, or both.
`reserved_assigned_same`	⟦`reserved_room_type` = `assigned_room_type`⟧

3. **Drop the useless columns**:

   Drop [`ID`, `distribution_channel`, `agent`, `company`, `days_in_waiting_list`] because the correlation between these features and the target value(adr or is_canceled) is close to 0.

4. **Replace Nan and NULL with 0**

5. **Dummy Variable**:

   For non-numerical variables, we convert it to dummy variables.

3 Model 1

1. **Result**:

   The model 1 scores 0.350649 on both the public and private dataset.

2. **ADR Prediction**

   (a) **Model**
   Gradient Boosting

   (b) **Implementation**
   Among a few variants of gradient boosting, we use DART (Dropouts meet Multiple Additive Regression Trees) instead of the ordinary decision tree. DART will drop a fraction of previous results, which prevents a typical drawback of ordinary GB. Trees inserted in the beginning are likely to dominate the total performance of GB. Thus, it can prevent overfitting.

   When constructing the decision tree, we remain 20% of the data for validation to prevent overfitting, which is similar to the concept of bagging introduced in class.

   Next, we decrease the learning rate and increase the number of tree in the GB in order not to arrive at the sub-optimal point.

   Last, use GridSearchCV to find the optimal set of other parameters.

   (c) **Performance**: Recorded in the comparison section.

3. **Cancellation Prediction**

   (a) **Model**
   A dense neural network is used here. The output is a vector with two entries, the probability of canceling and the probability of not canceling. Thus, we can update the weights depending on the cross entropy.

   - **Input**: After doing the feature transform mentioned above, feed them directly to the NN.
   - **Architecture**:
     Dense(dim = 512, dropout = 0.1, activation = relu) $\rightarrow$ Dense(dim = 256, dropout = 0.15, activation = relu) $\rightarrow$ Dense(dim = 2, activation = softmax)

- **Initialization**: He initialization.
- **Optimizer**: Adam.
- **Loss Function**: Cross entropy.

(b) **Performance**

Metrics	Training Set	Validation Set
Cross Entropy	0.2434	0.2851
Accuracy	0.8855	0.8816

Table 2: Performance of the NN Cancelling Prediction Model

(c) **Some Reasons**

- **Initialization**: Multiply the random number by $\sqrt{\frac{2}{size^{(l-1)}}}$. Perform slightly better than the default initialization in keras.
- **Activation**: Use RELU in order to prevent vanishing gradient that will happen in tanh and sigmoid.
- **Optimizer**: Concept of RMSprop along with momentum. Momentum is that the gradient we use to update weights in DNN in the last epoch will be stored and add to the next gradient that we use to update weights in the next epoch. This approach is efficient in preventing stucking to local minimum.

4. **Label Prediction**:

The model is implemented with 18 decision stumps. Label = $\sum_{n=1}^{18} 0.5 * DS(x)$. The decision stump is set as follow, if a is the last value of label i and b is the last value of label $i+1$, we set one decision stump to be $\frac{a+2b}{3}$ and the other be $\frac{2a+b}{3}$.

4 Model 2

1. **Result:**

The model 2 scores 0.434211 and 0.428571 on the public and private dataset, respectively.

2. **ADR Prediction**:

(a) **Model**: Random Forest

(b) **Tuning Hyperparameters**: The GridSearchCV function in sklearn package tries all combinations in the parameter space we provided and returns the best parameter combination according to the cross-validation score. The function search for the best parameters in the following parameter space:

- `n_estimators`: $[100, 300, 500, 700, 900, 1100]$
  The number of trees in the forest, the larger the better, but it also consumes greater time.

- `max_depth`: $[10, 30, 50, 70, 90, 110]$
  The maximum depth of the tree. The bias decreases when the trees are getting deeper, but on the other hands, the variance increases.

Eventually, `n_estimators` and `max_depth` are chosen to be 700 and 90 respectively.

3. **Cancellation Prediction**: The same as model 1.

4. **Label Prediction**: The same as model 1.

5 Model 3

1. **Result**:

   This model scores 1.117332 and 1.080799 on the public and private dataset respectively.

2. **ADR Prediction**

   (a) **Model**

   - **Input**: After doing the feature transform mentioned above, feed them directly to the NN.
   - **Architecture**:
     Dense(dim = 256, activation = relu) $\rightarrow$ Dense(dim = 256, dropout = 0.2, activation = relu) $\rightarrow$ Dense(dim = 512, dropout = 0.5, activation = relu) $\rightarrow$ Dense(dim = 256, dropout = 0.4, activation = relu) $\rightarrow$ Dense(dim = 1, activation = linear)
   - **Optimizer**: Adam.
   - **Loss Function**: Mean squared error.

   (b) **Performance**: Recorded in the comparison section.

3. **Cancellation Prediction**: The same as model 1.

4. **Label Prediction**:

   (a) **Why RNN?**
   After calculating $P_{cancel}(x)$ and $ADR(x)$ of each reservation x, instead of taking the inner product of them, we can see them as a sequence with each entry being a pair $(P_{cancel}(x), ADR(x))$. In fact, we need to add more features to each entry and make it a vector that stores more information of a reservation. Now the problem becomes a sequence-to-1 prediction problem. Thus, we decide to try RNN out, which is a widely used model to tackle sequence-to-1 prediction problems.

   (b) **Model**:

   - **Input**: Sequences of reservations on a single day. Each reservation is a vector consists of various attributes, such as `predicted_adr`, `predicted_cancel`.
   - **Architecture**:
     RNN(dim = 16) $\rightarrow$ RNN(dim = 16) $\rightarrow$ Dense(dim = 1)

- **Optimizer**: Adam.
- **Loss Function**: Mean squared error.

(c) **Potential Reasons of Poor Results**:
Holding the two submodels fixed, we can get substantially better results by using a plain linear regression or a decision stump to predict the labels. This implies that RNN is not the best model to predict the labels.
In most of the cases, people use RNN not just to solve sequence-to-1 problem, they use it because the order of the sequence matters. RNN can solve many NLP problems, where the order of the words in a sentence matters. However, the order of the reservations in a single day does not affect the revenue of that day. Thus, we believe that RNN is not the best model to deal with this problem.

6 Comparison of the three models

Besides the RNN used in model 3 to predict the labels, the main difference between the three models is in the ADR prediction model. Thus, we'll focus on the comparison of the three ADR prediction model in this section.

1. **Performance and Efficiency**:

	Gradient Boosting (DART)	Random Forest	Neural Network
MSE_{train}	107	81.85	397
MSE_{val}	369	362	481
time	Around 16 min	Around 7 min	Around 17 min

Table 3: Performance of the Three Model

DART and Random Forest are the two models that have a great balance between performance and efficiency, while Neural Network performs way worse and is even more time consuming.

2. **Interpretability**:

(a) **Random Forest**: The decision tree is highly interpretable since its decision is easy for people to comprehend. By uniformly combining multiple decision trees, we only compromise little interpretability. This is because we can still understand how the model gets the result.

(b) **Gradient Boosting**: Gradient boosting approximates the label by amending the gap between our predictions and labels, this is somehow comprehensible but not that transparent compared with to RF.

(c) **Neural Network**: Neural network is almost unexplainable at first glance. It automatically extracts the important features from the data. In most cases, the extracted features are rather complicated. However, this ADR prediction model is nothing fancy but only a non-linear regression in essence.

Thus, $RF > GB \gg NN$ in terms of interpretability.

7 Recommended Models

The results of the three models are summarized in the Table 4. We recommend Model 1 for the label prediction, as it results in lowest error on private score board. The pros and cons of the three models implemented in model 1 are analyzed below.

	Model 1	Model 2	Model 3
Public Dataset	0.350649	0.434211	1.117332
Private Dataset	0.350649	0.428571	1.080799

Table 4: Results of Three Models

Gradient boosting is an effective technique that can be applied in a wide range of applications. XGBoost, the package we used in model 1, is based on Gradient boosting but is further designed to be highly efficient and portable. Using these techniques, we achieve a good prediction of ADR (MSE=375.73) within a relatively short time (16 m).

DNN, on the other hand, is a very different model from Gradient Boosting, while it generates a better result when predicting cancellation. It requires an advanced hardware(GPU) to run faster and some domain knowledge to design each layer. Generally, it is a popular model in every application, and also obtains a satisfactory result(Accuracy=0.8816) when predicting cancellation.

Finally, the Decision stump model is recommended for the transformation from daily revenue to label. Simple as it is, it avoids overfitting and being time consuming when it's trained with a large data set. Eventually, it obtains better results than other models.

Combining these three models, we can get a label prediction of MAE = 0.350649 in 30 min.

8 Collaboration Among Team Members

Tsai and Lin did the data preprocessing. Model 1 to Model 3 are proposed, trained and fine-tuned by Tsai, Tu, and Lin respectively. The best results on the public scoreboard and the private scoreboard are accomplished by Tu and Tsai respectively.

References

[1] Scikit-Learn official Document

[2] Keras official Document

[3] XGboost official Document