Final Project Survey Report

經濟四 陳昱嘉 B06303047 經濟四 羅允謙 B06303126

Predict Is_canceled

For all the bookings, we need to predict whether it will be canceled for counting revenue.

Under this framework, Predicting canceling is a classification problem, where 1 represents this booking is canceled and 0 represents it's valid otherwise.

We use **X** as a matrix containing all independent variables, such as lead time, number of customers, room type, company, agent and so on. **y** as a vector represents canceled rate.

Feature transform

Unobservable columns:

- In **X**, there are three columns actually unobservable, then they should be dropped at the first time.
  - `adr` : After this booking is verified, adr is available
  - `reservation_status` : The final status customer make, can check whether it's canceled
  - `reservation_status_date` : The same reason as above

Missing value

- In **X**, three columns need consider missing value
  - `country` : Only 1% of data is missing, in this case, we fill in the most frequent value
  - `agent` : 14% of this column is missing, we use 0 represents this booking as no agent
  - `company` : 94% of this column is missing, so we drop this column

Date

- In **X**, for Date information, we do such transform
  - `month` : Since hotel booking highly depends on seasons, we leave this columns as categorical variable
  - `year` : For new bookings, these columns as categorical variable are all 0. Thus we drop it.

New feature

- `is_same_room` : If in this booking, its `reserved_room` is the same to `assigned_room` , then it's 1 and 0 otherwise.
- `children` : It sums up `children` and `baby`
- `agent` : We assign agents to four groups, the criteria is based on previous canceling behavior.
  - `agentLikelyToCancel` : canceled rate > 0.75
  - `agentProneToCancel` : 0.5 < canceled rate <= 0.75
  - `agentNeutralToCancel` : 0.3 < canceled rate <= 0.5
  - `agentNotProneToCancel` : canceled rate <= 0.3
- `country` : We assign countries to three groups, also the criteria is based on previous canceling behavior.
  - `proneToCancel` : canceled rate >= 0.5
  - `neutralToCancel` : 0.3 < canceled rate < 0.5

    - `notProneToCancel` : canceled rate <= 0.3

Transform categorical columns

- Some columns are not numeric, we use One-hot encoding to transform them from categorical columns to numeric columns, such columns are: `hotel` , `market_segment` , `distribution_channel` , `deposit_type` , `customer_type` , `Month` , `reserved_room_type` , `assigned_room_type`

Model selection

First, we split our data to training data(70%) and testing data(30%). Then, We use 5-fold cross validation to select our model. Finally, we use the remaining testing data to see the performance of out-of sample data.

Baseline Model: Logistic Regression with l2 penalty

We decide to use `Logistic Regression` with l2 penalty. Not only because it has a superb property to deal with overfitting but also Logistic is a simple linear model which we think is very suitable for our baseline model. After training this data, we can look at the coef. to distinguish which feature is more important than others and have a overview of our feature transforming process. We found out that `total_of_special_requests` is a vital feature because when a booking has some special requests, he or she is less likely to cancel the booking. Also, using blending avoids us from algorithm variation and thus prevent overfitting.

> Time consumed for 5-fold cross validation: 11s
> 5-fold Balanced Accuracy: 0.79
> Testing Balanced Accuracy: 0.80

Best Model: Random Forest Classifier

Using random forest, we can catch the concrete information from data more easily.

> Time consumed for 5-fold cross validation: 81s
> 5-fold Balanced Accuracy: 0.89
> Testing Balanced Accuracy: 0.90

Predict ADR

For all the bookings, we need to predict the ADR of the fulfilled request(s).

Under this framework, Predicting ADR is a regression problem, where the predict ADR should be real number.

We use **X** as a matrix containing all independent variables, such as lead time, number of customers, room type, company, agent and so on. **y** as a vector represents ADR.

Dropping Data

Outlier: Noise data

From the definition of ADR

> After a room reservation request is fulfilled (i.e. no cancellation), on the arrival date, the revenue of the request is the rate of the room (called ADR) multiplied by the number of days that the customer is going to stay in the room.

This means that we should not add the one is canceled to our training data because it might be a noise for our model or it might be useless contributing our prediction of daily revenue. Therefore, we drop the booking requests canceled in the first step. Next, if we plot the distribution of ADR, we can see that ADR value is mostly bigger than 0 and lower than 400. Therefore, we decide to only keep 0 < ADR < 400 (just like making a mask: filtering out noise).

min	max	mean	std.	25%	50%	75%
-145	5400	85	51	53	80	110

Useless: booking request is canceled

From the definition of daily revenue(DR), we can exclude the booking requests that are canceled.

> The daily revenue is the sum of all those fulfilled requests on the same day.

Feature transform

Unobservable columns:

- In **X**, there are three columns actually unobservable, then they should be dropped at the first time.
  - `reservation_status` : The final status customer make, can check whether it's canceled.
  - `reservation_status_date` : The same reason as above.

Date and new data feature

- In **X**, for Date information, we do such transform
  - `month` : Since hotel booking highly depends on seasons, we transform this columns to categorical data.
  - `year` : For new bookings, these columns as categorical variable are all 0. Thus we drop it.
  - `is_weekday` : The revenue on weekday and non-weekday might be different.

Create new feature

- `is_same_room` : If in this booking, its `reserved_room` equals to `assigned_room` , then it's 1 and 0 otherwise.
  > The reason why we are creating this feature is because we think if you are booking for type-A room but are assigned to non-type-A. Maybe this may have impact on ADR and whether you want to cancel this booking.

- `total_customers` : It sums up `adult` , `children` , and `baby` .
- `children` : It sums up `children` and `baby` .
- `agent` : We assign agents to four groups, the criteria is based on previous mean ADR history.
  - `lowAdrAgent` : mean ADR < 100
  - `highAdrAgent` : mean ADR >=100
  > The reason why we are transforming agent and country is because we think there might be some pattern of that agent or country have influence to ADR.

- `country` : We assign countries to two groups, also the criteria is based on previous mean ADR history.
  - `lowAdrCountry` : mean ADR < 80
  - `midAdrCountry` : 80 <= mean ADR < 90
  - `highAdrCountry` : mean ADR >= 90

Transform categorical columns

- Some columns are not numeric, we one-hot encoding to transform them from categorical columns to numeric columns, such columns are: `hotel`, `market_segment`, `distribution_channel`, `deposit_type`, `customer_type`, `Month`, `reserved_room_type`, `assigned_room_type`

Model selection

First, we split our data to training data(70%) and testing data(30%). Then, We use 5-fold cross validation to select our model. Finally, we use the remaining testing data to see the performance of out-of sample data.

Baseline Model: Ridge Linear Regression

Instead of using linear regression of our baseline model, we decide to use`Ridge Regression`. Not only because it has a superb property to deal with overfitting than `Linear Regression` but also Ridge is a simple linear model which we think is very suitable for our baseline model. After training this data, we can look at the coef. to distinguish which feature is more important than others and have a overview of our feature transforming process.

> Time consumed for 5-fold cross validation: 0.7s(0.1s faster than Linear Regression)
> 5-fold Mean squared error(MSE): 996
> Testing Mean squared error(MSE): 984

Improved Model

Random Forest Regressor

Since some of our numeric features do not do any transform, such as`arrival_date_week_number` and `lead_time` etc. We decide to use tree-based model for our next model because tree-based model will give our nice non-linear property when it is deciding branching criteria. It turns out the performance is much better than `Ridge` as expected.

> Time consumed for 5-fold cross validation: 150s
> 5-fold Mean squared error(MSE): 840
> Testing Mean squared error(MSE): 833

LGBMRegressor

After the previous model succeed, we decide to try a more popular, faster, and have better average performance than `Random Forest`. Light GBM grows tree vertically while other algorithm grows trees horizontally. It will choose the leaf with max delta loss to grow. However, we should know than this model is very sensitive to overfitting and can easily overfit small data. We also use 5-fold to carefully watch the performance of this model.

> Time consumed for 5-fold cross validation: 12s
> 5-fold Mean squared error(MSE): 749
> Testing Mean squared error(MSE): 751

Best Model: Neuarl Network Model

After trying Machine learning models, it would be a great idea to try about simple neural network trying to improve MSE score. Since we have already done feature engineering, we think a relatively deep neural network may overfit and have bad result. Therefore, we choose a 79(input)-512(hidden)-1(output) with activation function using Relu and use earlystopping, dropout, and reduce learning rate on plateau to combat overfitting.

> Time consumed for training: 1s per epoch, about 36,000 data and batch size is 32 (Using GPU in Azure)

> Testing Mean squared error(MSE): 390

- Pros:
  1. Robustness to natural variations in the data is automatically learned.
  2. DL model will automatically non-linearly transofrom feature if it’s useful.(Since we do not use polynomial transform in our data)
  3. We can use GPU and parallel computations to speed up the long training process.
  4. DL’s architecture that can be adapted to new problems relatively easily.
  5. Although training process is time-consumed, the predictions are pretty fast once trained.
- Cons:
  1. It is extremely expensive to train and using GPU to accelerate due to complex data models.
  2. DL model is so powerful may easily be overfitting.
  3. It’s not easy to interpret how DL model makes a decision.
  4. Required lots of data compared to Machine learning.
  5. Do not have much in the way of strong theoretical foundation. This leads to the next disadvantage.
  6. Determining the topology/flavor/training method/hyperparameters for deep learning is a black art with no theory to guide you.

Predict label

Our goal is to predict the daily revenue of that day given you booking requests. Our prediction**y** should to be within {0, 1, 2, . . . , 9}.

We first utilize our first model to predict whethter the booking request is canceled. If our model predict this book will be canceled, we remove this request since it won’t contribute to our daily revenue. Next, we predict the ADR of the remaining booking requests.

Now we can calculate revenue per booking = (total stay nights) * (ADR). Finally, forecasting the daily revenue is just sum up the all of the bookings’ revenue at the same day. If we groupby `label` and calculate the mean daily revenue would look there’s absolutely a pattern bewteen `label` and `daily revenue`.

We believe no matter our model is will yield satisfactory result. As a matter of fact, our model’s mean absolute error is around 0.1 ~ 0.3 (in 5-fold cross validation). The best one is Random Forest (MAE: 0.1).

label	Mean Daily Revenue	label	Mean Daily Revenue
0	7655	5	52103
1	16368	6	62416
2	25291	7	73309
3	35056	8	83156
4	44441	9	97722