Job Assignment

朱瑞斌: Doing experiments, writing codes, writing the report

蘇浚笙: Emotion controller B-)8===D, supportive work, dev-ops, debugger

謝辰陽: Data analyzation, experiments analyzation, writing the report

Package Used:

We use python 3.7 and the following packages to complete the final project: scikit-learn, keras, pandas, numpy and seaborn.

Data Analyzing

First, we observe that there is a simply formula for calculating the quantized label of daily revenue, which is $\lfloor \Sigma_{not\ canceled}(adr\ *\ stayed\ days)\ /\ 10000 \rfloor$.

Then, we rename the time columns 'arrival_date_***' to 'year', 'month', 'week' and 'day' respectively, while mapping 'month' to their corresponding number.

We believe that analyzing the physical meaning of the columns helps. Thus, we create a new column 'booking_size'(total number of people). Also, the total time staying at the hotel probably implies some relationship in the data, so a new variable 'duration' is created. In order to predict 'adr', we introduce a new variable 'adr_pp'(average daily rate per person). Also, since there are only two hotels in the data, we separate them in the following analysis.

Firstly, we plot the correlation between each variable. In these figures, namely 1-(a) and 1-(b), we can see that 'booking_size' is highly correlated with 'adr', so it may help the machine learn better if this column is added. Also, 'lead_time' is also highly correlated with 'is_canceled', which probably implies that the longer the time between the date of the booking and the arrival date, the more possible the booking is cancelled.

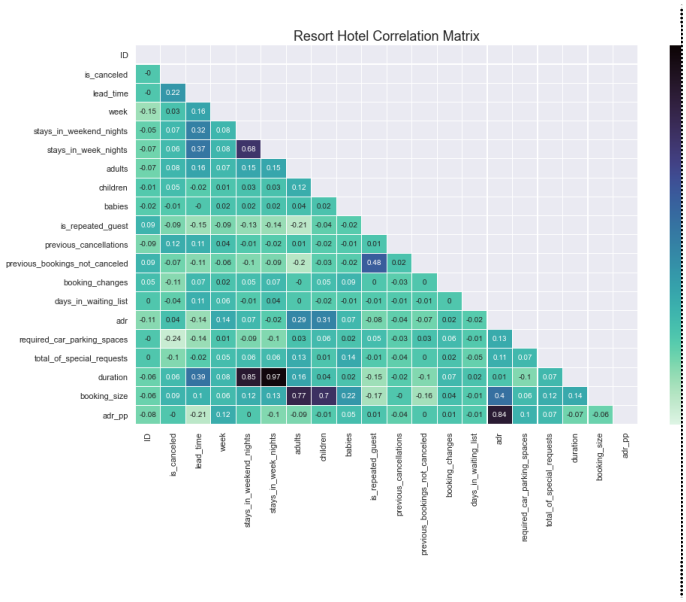

figure 1-(a)

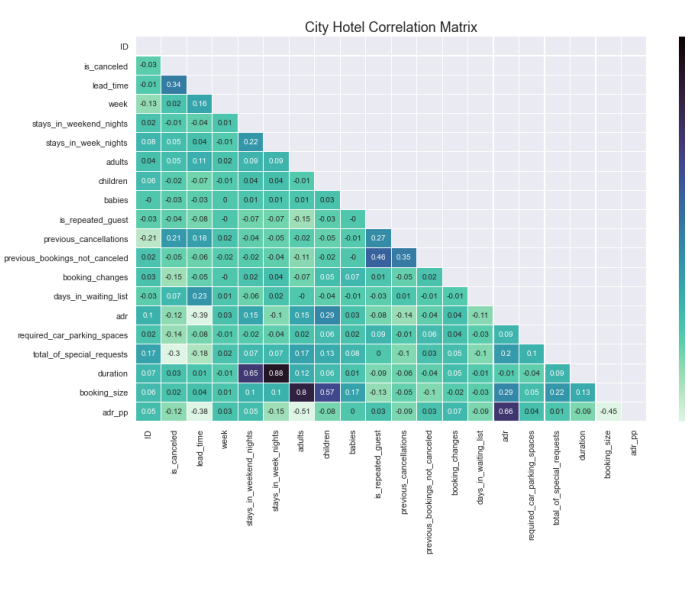

figure 1-(b)

We can visualize the correlation between categorical variables in a similar way. By doing this, we find out that 'is_canceled' is highly correlated with 'country', 'revervation_status', 'market_segment', 'distribution_channel', and 'deposit_type'.

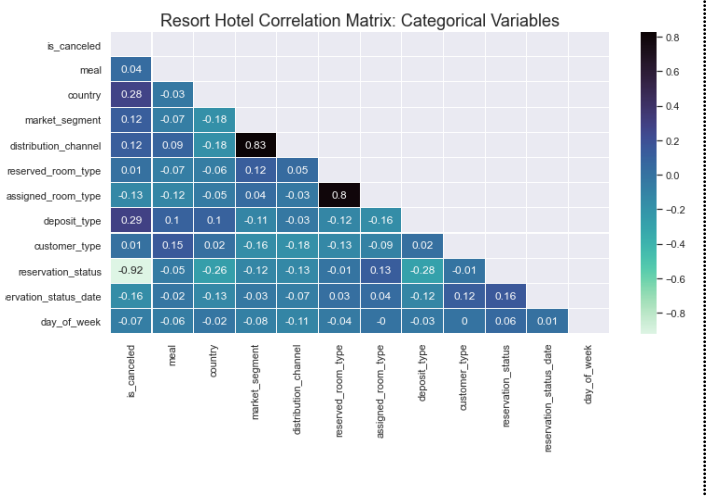

figure 2-(a)

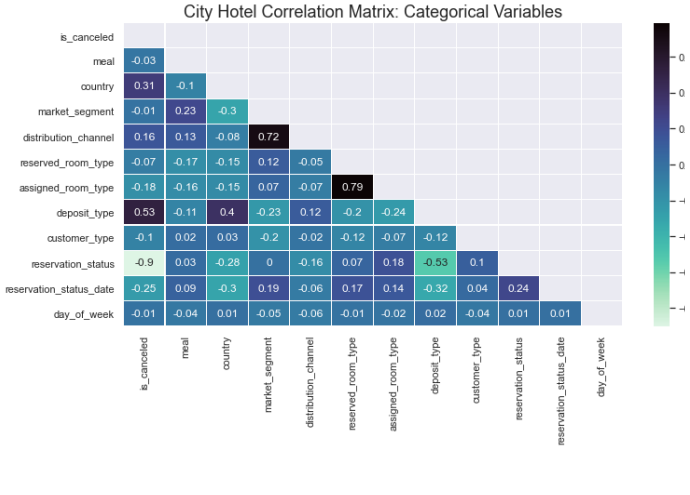

figure 2-(b)

Then, we see a ridiculously high correlation between 'is_canceled' and 'deposit_type' in figure 2-(b). After visualizing the relationship, in figure 3, it is shown that "non refund" bookings have a cancelation rate of 99.8%, implying that almost all customers making a non-refundable booking cancelled their booking, which we think is nonsense. Yet, we have no further clues on this strange data feature.

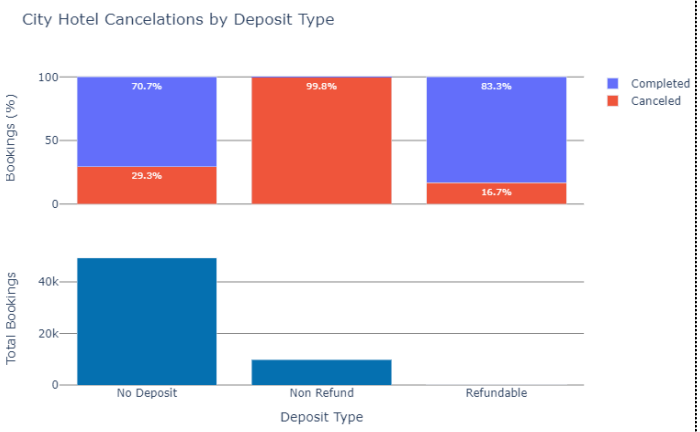

figure 3

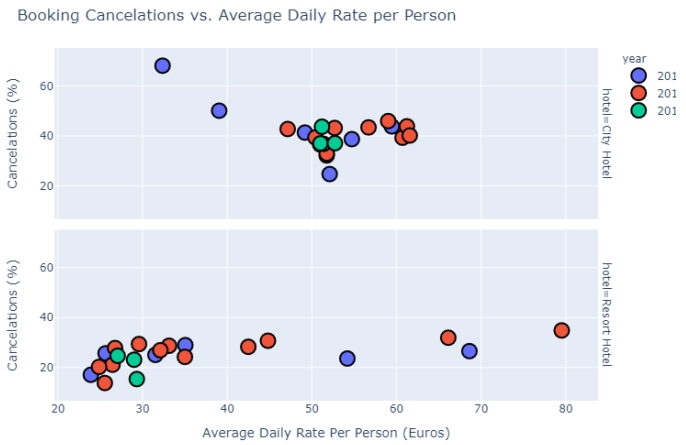

figure 4

Finally, we check the relationship between cancelation rate and adr_pp of the two hotels. In figure 4, we clearly see that the relationships behave in different ways. At the City Hotel, adr_pp seems to have no significant impact on the cancelation rate, where, at the Resort Hotel, as adr_pp goes higher, the cancelation rate increases as well, which probably implies that the customers here are more price-sensitive. By the way, it is worth noting that there is an outlier at the City Hotel, which is July, 2015, probably caused by nearby events of negative impact. Maybe this outlier will have an unpredictable impact on the models.

Data Preprocessing

For preprocessing data, we implement it from four different aspects: handling missing data, removing noise, feature encoding and scaling.

Missing data:

The data has some missing values as the following figure.

```
children       4
country      468
agent      13217
company    85917
```

From the figure we can see that there are some missing entries in the table. Since there are only 4 regarding children, we drop them. For other entries, we fill them with 0 and ‘’.

Removing noise:

We first create two new columns, which are duration and booking size. Duration is the sum of stays_in_week_nights and stays_in_weekend_nights, booking size is the sum of adults and children.
Then, we observed that there is lots of weird data. We reserve these:

1. -5 <= adr <= 3 * standard deviation + mean **2.** stays_in_weekend_nights <= 6
3. previous_cancellations <= 10 **4.** booking_size != 0 **5.** duration != 0;
6. adults <= 6 **7**. children <= 3 **8**. babies <= 2;

Feature encoding:

The data has a lot of non-numerical features, which has to convert into numerical features that can be trained.
We come up with 2 different approaches: label encoding and one-hot encoding.
Since there is no order in these categorical features, we believe that one-hot encoding specifies this point. However, label encoding is still worth a try.

Scaling:

We also implement scaling before training our models. There are 2 different methods we used: standard scaling and max abs scaling.
We want to see whether retaining the sparsity of the data, or changing the balance of the features will have an impact on the models.

Baseline Models

We choose three different models for the final project, which are random forest (RF), neural network (NN) and linear regression/ logistic regression (LR) respectively.
In the following tables, the ones colored orange are chosen as baseline models.

Implementation details:

Here are our implementation details when training following models.
Missing data: fill **Removing data:** removed
Encoding: label encoding **Scaling:** max abs scaling
Validation:
E_val: Last 20% of data
E_val': 3 mean of errors from 20% of data picked randomly
Error measurement: mean squared error for adr models and accuracy score(1 - 0/1 error) for canceled models. (higher accuracy score is better!)

RF:

For RF models, we compared the different n_estimators and min_samples split.

n_estimators: (min_samples split=2)

	adr			is_canceled		
	100	**300**	**500**	**100**	**300**	**500**
E_in	66.60	65.44	65.06	0.995	0.995	0.995
E_val'	261.82	259.46	258.78	0.906	0.907	0.906
E_val	387.19	378.24	380.97	0.805	0.806	0.805

min_samples_split: (n_estimators=100)

	adr			is_canceled		
	2	**10**	**25**	**2**	**10**	**25**
E_in	66.34	112.19	178.44	0.995	0.957	0.925
E_val'	259.12	272.90	286.85	0.905	0.903	0.896
E_val	379.02	380.76	383.55	0.805	0.804	0.803

For a larger number of n_estimators, the performance becomes better, while the improvement is relatively subtle when the number of n_estimators is large enough. Meanwhile, the value of min_samples_split should prevent the model from overfitting. We see an increase in E_in, yet, the validation error seems to increase, which probably implies a very slight underfitting.

NN:

For NN models, we compared the affection of different numbers of layers.

layers: (epochs=50, batch_size=50)

	adr				is_canceled			
	[15,15]	**[15,15, 15,15]**	**[30,30]**	**[30,30, 30,30]**	**[15,15]**	**[15,15, 15,15]**	**[30,30]**	**[30,30, 30,30]**

E_in	736.89	654.62	658.21	654.95	0.824	0.827	0.838	0.834
E_val'	742.25	665.43	660.81	654.99	0.825	0.835	0.835	0.837
E_val	730.73	598.52	732.38	516.20	0.786	0.772	0.772	0.790

It is obvious that as we increase the number of layers, the accuracy actually increases. Also, as we increase the number of neurons in each layer, performance improves. By the way, we have tried different epochs and batch_size, and the results seem to have no notable changes.

LR:

For the logistic regression model, we compared different regularizers and different c.

	adr (linear regression)	**is_canceled (logistic regression)**			
		l2 C=0.1	**l2 C=1**	**l2 C=10**	**none**
E_in	1127.16	0.749	0.746	0.750	0.749
E_val'	1133.08	0.750	0.742	0.753	0.751
E_val	1340.52	0.704	0.718	0.716	0.703

Since we haven't done feature transform for linear regression and it has no argument to be adjusted, its error is quite high. For logistic regression, whatever values of C we choose and regularizer we use, its performance seems not to be affected significantly.

Actually, whatever models we use and arguments we choose, there is no prominent improvement in the prediction of cancelation.

There is one more thing worth noticing. That is, the validation error using random sampling and that using the last 20% of the data can have a different implication on the final E_out. When using the former, we once predicted two outputs with nearly the same validation error. After submitting them onto the online judge, we got a totally different error of 0.34 and 0.5. In contrast, using the last 20% of the data to validate gives a better prediction of E_out. Since we are predicting the future, this result is quite reasonable.

Here are the final results of our baseline models.

	adr			is_canceled		
	RF	**NN**	**LR**	**RF**	**NN**	**LR**
E_in	65.44	654.95	1127.16	0.995	0.834	0.746
E_val	378.24	516.20	1340.52	0.806	0.790	0.718

Experiments

For the next experiments, we compared different methods of data preprocessing with baseline models above. The column colored with green denoted there is a notable improvement while red denoted the opposite. The implementation details are the same as above if not specified.

Not Remove Noise vs Remove Noise (baseline models)

	adr			is_canceled		
	RF	**NN**	**LR**	**RF**	**NN**	**LR**
E_in	140.92	1131.61	1772.37	0.995	0.861	0.739
E_val	521.18	782.07	1648.92	0.812	0.778	0.732

Retaining the noise helps the performance of predicting ‘is_canceled’. The reason is probably that those noises, namely outliers, are mostly cancelled, so keeping the noise actually helps the model learn slightly better. However, without removing the noise, the prediction of adr has a much worse performance.

Label Encoding vs One-Hot Encoding (baseline models)

	adr			is_canceled		
	RF	**NN**	**LR**	**RF**	**NN**	**LR**
E_in	71.37	210.52	670.63	0.995	0.939	0.846
E_val	420.35	599.73	?	0.791	0.785	0.784

Surprisingly, with label encoding, most models actually perform better. Meanwhile, NN and LR have a significantly lower E_in on predicting adr, while NN is obviously overfitting and the validation error of LR goes to infinity and beyond. Sadly, the reason behind the phenomenon remains unknown :(There is one point worth noticing, which is that LR performs actually better predicting cancelation with one-hot encoding.

Standard Scaling vs Max Abs Scaling (baseline models)

	adr			is_canceled		
	RF	**NN**	**LR**	**RF**	**NN**	**LR**
E_in	70.57	543.99	1181.13	0.995	0.854	0.756
E_val	381.79	618.74	1340.51	0.805	0.777	0.718

The different scaling methods seem to have no prominent impact on RF and LR. However, with standard scaling, it is obvious that NN behaves a lot worse. Maybe the sparsity of the data being changed by standard scaling is the reason.

Seperate Two Hotels

From the section of data analyzing, we observe that two hotels are greatly different in some of its features. Therefore, we train the models of different hotels separately in hope that the accuracy can be improved. Yet, the overall accuracy is basically the same.

(R for Resort Hotel, C for City Hotel)

	adr						is_canceled					
	RF		**NN**		**LR**		**RF**		**NN**		**LR**	
	R	**C**	**R**	**C**	**R**	**C**	**R**	**C**	**R**	**C**	**R**	**C**
E_in	58.77	76.29	554.44	565.82	1651.78	749.01	0.997	0.993	0.867	0.851	0.799	0.747
Average	70.44		562.05		1050.38		0.994		0.856		0.764	
E_val	360.81	453.90	456.65	537.42	1933.05	1005.88	0.847	0.780	0.820	0.731	0.804	0.691

Average	422.85	510.49	1315.41	0.802	0.760	0.728

Conclusion:

We observe that the cancelation models trained with different models and different methods have similar performance. Among those, training with noise retained results in the best performance.

Our final selections are the baseline models of RF, while predicting cancelation is trained with noise retained. The model gives us a public score of 0.30 and a private score of 0.38, which is VERY GOOD! :-D

Pros: Easy data preparation, fast training, sophisticated interpretation, good handling of categorical features
Cons: Poor extrapolation accuracy (only one since we didn't actually run into trouble:-))