Machine Learning Techniques : Final Report

吳承哲 **R08525116**

温皓良 **D09525007**

何明昕 **R08525101**

Abstract

This report is aimed to investigate the different ability of prediction among three different models on the hotel booking demand data, and the treatment of the real-world data encoding for model training. The three different models in Scikit-learn is, Random Forest, Nearest Neighbor and Support Vector Machine. They would be compared though classification and regression problems intensively.

1 Training strategy

There are 91531 pieces of data in training data, including 33 features, while there are 29 features in test data of size 27859. Our strategy is to find the best model to predict whether the client will cancel the reservation and how much the hotel can earn from the trade, or, the corresponding averaged daily rate (ADR). Therefore, the revenue of the hotel can be calculated by following formula:

$$\text{Revenue} = \max\left(\min\left(\left\lfloor \frac{\sum(\text{stay in week/weekend nights}) \times \text{ADR}}{10000} \right\rfloor, 9\right), 0\right) \tag{1}$$

2 Data pre-processing

With the aid of Numpy[1] and Pandas[2] packages, the original data set is filtered by the process described in following tables. Secondly, "One-hot encoding" method is implemented to encode the string-like features in the data set. To avoid the inconsistent number of features in test/train data, training and testing sets are considered in the encoding process to capture all the features. Company/Agent ID will be transformed to string for the one-hot encoding. After replacing all the NAN with zero, the dimension of the training and testing data is $(88974, 931)$ and $(27859, 931)$, respectively.

Delete selected features

Name	Reason
id	This feature is useless.
reservation_status	This feature is not found in test data.
reservation_status_data	This feature is not found in test data.

Delete abnormal Data

Condition
Sum of "stay in week/weekend nights" is zero
Sum of "adults","children","babies" is zero
Negative ADR

Delete data with extreme value

Condition	Number of data
ADR > 5000	1
Babies > 1	2
Undefined distributed channel	1

3 Model selection

Three free library are chosen as our candidate models: Random Forest, Nearest Neighbor, Support Vector Machine. These models can be found in Scikit-learn[3]. The data is split into two parts with ratio of 1 : 4 for the sake of validation. For *is_cancel* and *adr* predictions, the best model of each prediction will be chosen according to validation accuracy at their default settings. After the model is selected, the model will be optimized for its best parameters. To reduce the variance of a base estimator, by introducing randomization into its construction procedure, each model is embedded in bagging meta-estimator[3]. Some important features of bagging meta-estimator from the website of Scikit-learn are quoted as follows:

- When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting.[4]
- When samples are drawn with replacement, then the method is known as Bagging.[5]
- When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces.[6]
- Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches.[7]

Define the parameter Θ as bellow

$$\Theta = \frac{\text{Size of training + validation data}}{\text{Size of all data}}, \tag{2}$$

the parameter Θ can help us to estimate the scalability of each method.

3.1 *is_cancel* prediction (classification)

3.1.1 Numerical test

Θ	Val. Acc(%)	CPU time	R.O.C.
	Random Forest Classifier		
1/16	86.53	14.55	-
1/8	86.29	5.49	-1.41
1/4	88.04	11.03	1.01
1/2	89.23	25.32	1.20
1	90.43	57.36	1.18
	Nearest Neighbor Classifier		
1/16	71.07	2.97	-
1/8	72.67	6.74	1.18
1/4	75.05	16.45	1.29
1/2	77.63	34.14	1.05
1	80.47	122.14	1.84
	SVM Classifier (RBF)		
1/16	68.82	13.18	-
1/8	68.04	44.05	1.74
1/4	68.91	175.19	1.99
1/2	68.11	702.40	2.00
1	68.46	5992.95	3.09

Table 1: Numerical test on three different models, rate of convergence (R.O.C.) is calculated from Θ and the CPU time.

According to Table 1, Random Forest Classifier has the best accuracy of prediction and the least rate of growth of CPU time as the data size increases. Hence, Random Forest Classifier is the best model of three for *is_cancel* prediction. In the next section, the model will be tested intensively to find the optimal parameters.

Due importance of parameters in SVM with RBF kernel, we pay some effort to find its optimal parameters to estimate its prediction ability. According to our test, SVM has the best validation accuracy 83.30% with parameter $C = 500, \gamma = 0.0005$.

3.1.2 Optimal parameters

According to Table 2, the optimal parameters settings for Random Forest Classifier is *max_features* = 0.125, *n_estimators* = 1000.

	10	50	100	500	1000
sqrt	90.25	90.48	90.50	90.54	90.60
$\log_2$	89.90	89.93	90.13	90.07	90.00
All	90.53	90.51	90.55	90.48	90.71
0.5	90.60	90.58	90.60	90.78	90.67
0.25	90.81	90.80	90.86	90.78	90.87
0.125	90.80	90.80	90.83	90.81	90.89

Table 2: Validation accuracy table for different parameters using Random Forest Classifier. The top row shows the number of estimators, and the left most column indicates the different option of *max_features*.

3.2 *adr* prediction (regression)

3.2.1 Numerical test

For *adr* regression, we choose the mean absolute error (MAE) as our validation criterion. According to our numerical results in Table 3, Random Forest Regressor can predict the value of *adr* most accurately in comparison to others. However, its computational efficiency is not the most efficient but the Nearest Neighbor Regressor.

Similar to section 3.1.1, we spend some time to find the best parameters for SVM Regressor, it is noted that the least MAE obtained from our test is around 20.

Θ	Val. MAE	CPU time	R.O.C.
		Random Forest Regressor	
1/16	15.28	3.43	-
1/8	14.60	4.65	0.44
1/4	13.11	9.75	1.07
1/2	12.04	26.72	1.45
1	11.27	69.06	1.37
		Nearest Neighbor Regressor	
1/16	27.76	0.16	-
1/8	26.80	0.36	1.17
1/4	23.55	0.77	1.10
1/2	21.57	1.48	0.94
1	19.52	2.94	0.99
		SVM Regressor (RBF)	
1/16	33.42	11.35	-
1/8	34.48	41.07	1.85
1/4	34.10	159.45	1.96
1/2	33.57	655.43	2.04
1	31.57	5365.73	5.07

Table 3: Numerical test on three different models, rate of convergence (R.O.C.) is calculated from Θ and the CPU time.

3.2.2 Optimal parameters

After series of numerical test is performed, it is noted that the best parameters setting for Random Forest Regressor is *max_features* = 0.5, *n_estimators* = 1000, according to Table 4. However, *n_estimators*=500 will be applied for better efficiency with neglectable sacrifice of accuracy.

	10	50	100	500	1000
sqrt	13.47	13.36	13.32	13.31	13.29
$\log_2$	14.36	14.20	14.19	14.18	14.16
All	11.33	11.26	11.26	11.24	11.26
0.5	11.24	11.22	11.21	11.21	11.20
0.25	11.53	11.43	11.41	11.41	11.40
0.125	12.05	11.91	11.94	11.90	11.89

Table 4: Validation MAE table for different parameters using Random Forest Regressor. The top row shows the number of estimators, and the left most column indicates the different option of *max_features*.

4 Revenue calculation

After getting the best parameters setting for model predicting *adr* and *is_cancel*, the revenue can be calculated by using Eq. (1) directly. After submitting the predicted labels of revenue to the project website, our model get the public score of 0.355263 and private score of 0.454545.

5 Conclusion

Scikit-learn is one of the most popular library in the society of machine learning. They provide a wide variety of choice for user to study, including supervised and unsupervised learning problems. This is the reason why we use the Scikit-learn, it's easy to use, and deep enough to dig. For three selected models, we recommend Random Forest for both classification and regression problems, in comparison with Nearest Neighbor and Support Vector Machine.

References

[1] Harris, C.R., Millman, K.J., van der Walt, S.J. et al., *Array programming with NumPy*, Nature 585, 357-362 (2020).

[2] Wes McKinney, *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56 (2010).

[3] Fabian Pedregosa, et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, 2825-2830 (2011).

[4] L. Breiman, *Pasting small votes for classification in large databases and on-line*, Machine Learning, 36, 1, 85-103, (1999).

[5] L. Breiman, *Bagging predictors*, Machine Learning, 24, 2, 123-140, (1996).

[6] T. Ho, *The random subspace method for constructing decision forests*, Pattern Analysis and Machine Intelligence, 20, 8, 832-844, (1998).

[7] G. Louppe, P. Geurts, *Ensembles on Random Patches*, Machine Learning and Knowledge Discovery in Databases, 346-361, (2012).