Expedia expected experience

How many days would a customer stay in an hotel?

Emanuele Mercanti 1719869 Andrea Trianni 1806198

Final Project of Machine Learning Course
A.Y. 2020/2021 - Sapienza University of Rome

Contents

Note: this written report explains the results collected in *notebook.ipybn* !

1 Introduction

1.1 Problem statement

Tourism is (was) one of the biggest business for many countries, such as Italy. When a customer is interested in purchasing a Flight, an Hotel or a travelling package, knowing in advance how long the customer wants to stay is useful to create personalized suggestions or advertisement. The more accuarate the suggestions, the more likely the customer is to buy the suggested item.

1.2 Proposed methods

The main libraries we have used are scikit-learn, pandas and numpy. In the data exploration phase, we used mostly the following tools: correlations, countplots and boxenplots (also know as letter-value plots), since they provide a better representation of a distribution than boxplots. The main issue with boxplots is information provided by them is appropriately vague beyond the quartiles, especially for large datasets, which is our case. Boxenplots conveys more detailed information in the tails.
After exploring the data, we proceed to engineer features (missing values, encoding, normalization). Then we test and evaluate five different models: Decision tree classifier, support vector machine, K-nearest neighbor, decision forest classifier and multi layer perceptron. We test the performance according to different metrics. We then proceed to select the best model and tune its hyperparameters. Lastly, to explain the results and drawn final considerations, we relied on the SHAP library.

1.3 Dataset overview

The dataset we used is the Expedia Hotel Recommendation available on Kaggle at Expedia hotel recommendations. The aim of the challange is to predict the booking outcome (hotel cluster) for a user event, based on their search and other attributes associated with that user event. Expedia has in-house algorithms to form hotel clusters, where similar hotels for a search (based on historical price, customer star ratings, geographical locations relative to city center, etc) are grouped together. However we are only given the ID of the hotel cluster and no further information about it. This would make interpreting the results not feasible. Thus, we defined our own problem: instead of predicting the hotel cluster a user would book, we predict how many nights he will spend in the booked hotel. The dataset contains data from 2013 to 2014, for a total of approximately 37 millions points. We only consider a random subset of it (the dataset has already been randomized by Expedia).
The data has 24 features as it is possible to see in **Table 1**. To compute the duration of stay we used the check-in date (srch_in) and the check-out date (srch_co). The target variable is called stay. After that we proceeded to remove the srch_co column from the dataset. We use only 100.000 samples from the whole dataset due to computational reasons.

Column name	Description	Variable type
date_time	Timestamp	timestamp
site_name	ID of the Expedia point of sale (i.e. Expedia.com, ...)	categorical
posa_continent	ID of continent associated with site_name	categorical
user_location_country	The ID of the country the customer is located	categorical
user_location_region	The ID of the region the customer is located	categorical
user_location_city	The ID of the city the customer is located	categorical
orig_destination_distance	Physical distance between a hotel and a customer.	continuous
user_id	ID of user	categorical
is_mobile	1 when a user connected from a mobile device, 0 otherwise	boolean
is_package	1 if the click/booking was generated as a part of a package	boolean

channel	ID of a marketing channel	categorical
is_booking	1 if a booking, 0 if a click	boolean
srch_ci	Checkin date	timestamp
srch_co	Checkout date	timestamp
srch_adults_cnt	The number of adults specified in the hotel room	categorical
srch_children_cnt	The number of children specified in the hotel room	categorical
srch_rm_cnt	The number of hotel rooms specified in the search	categorical
srch_destination_id	ID of the destination where the hotel search was performed	categorical
srch_destination_type_id	Type of destination	categorical
hotel_continent	Hotel continent	categorical
hotel_country	Hotel country	categorical
hotel_market	Hotel market	categorical
cnt	Number of similar events in the context of the same user session	discrete
hotel_cluster	ID of a hotel cluster	categorical

Table 1: Summary information about the dataset

2 Data exploration

We start by analyzing the distribution of the target variable. Stay has 15 classes. Each class corresponds to how many days the customer spent in the hotel. Duration longer than 15 days are cut to 15, since they are extremely rare. Hence the 15 days class is representative of duration ranging from 15 to 50 days. As it is possible to see in **Figure 1** the distribution is heavily skewed towards short stay duration. The target classes are heavily unbalanced, we explain how we balance them in the feature engineering section.

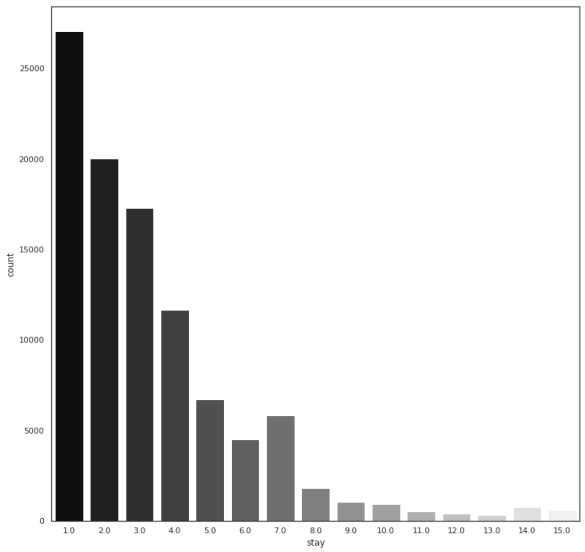

Figure 1: Distribution of the duration of days spent in the hotel

Then we explore the correlation between variables. As it is possible to see in **Figure 2**. The 3 most correlated features with the target are:

- is_package: positively correlated, binary variable;

- orig_destination_distance: positively correlated, continuous variable;
- hotel_continent: positively correlated, categorical variable with 6 categories.

The is_package variable is 1 if the click/booking was generated as part of a package, 0 otherwise. It is correlated with stay because usually when a travelling package is bought, the duration of the stay is already set, giving the customer less flexibility: we expect customers who bought a package not to stay for e.g. 1 day or 9 days, rather we expect them to stay for a weekend or for a week.

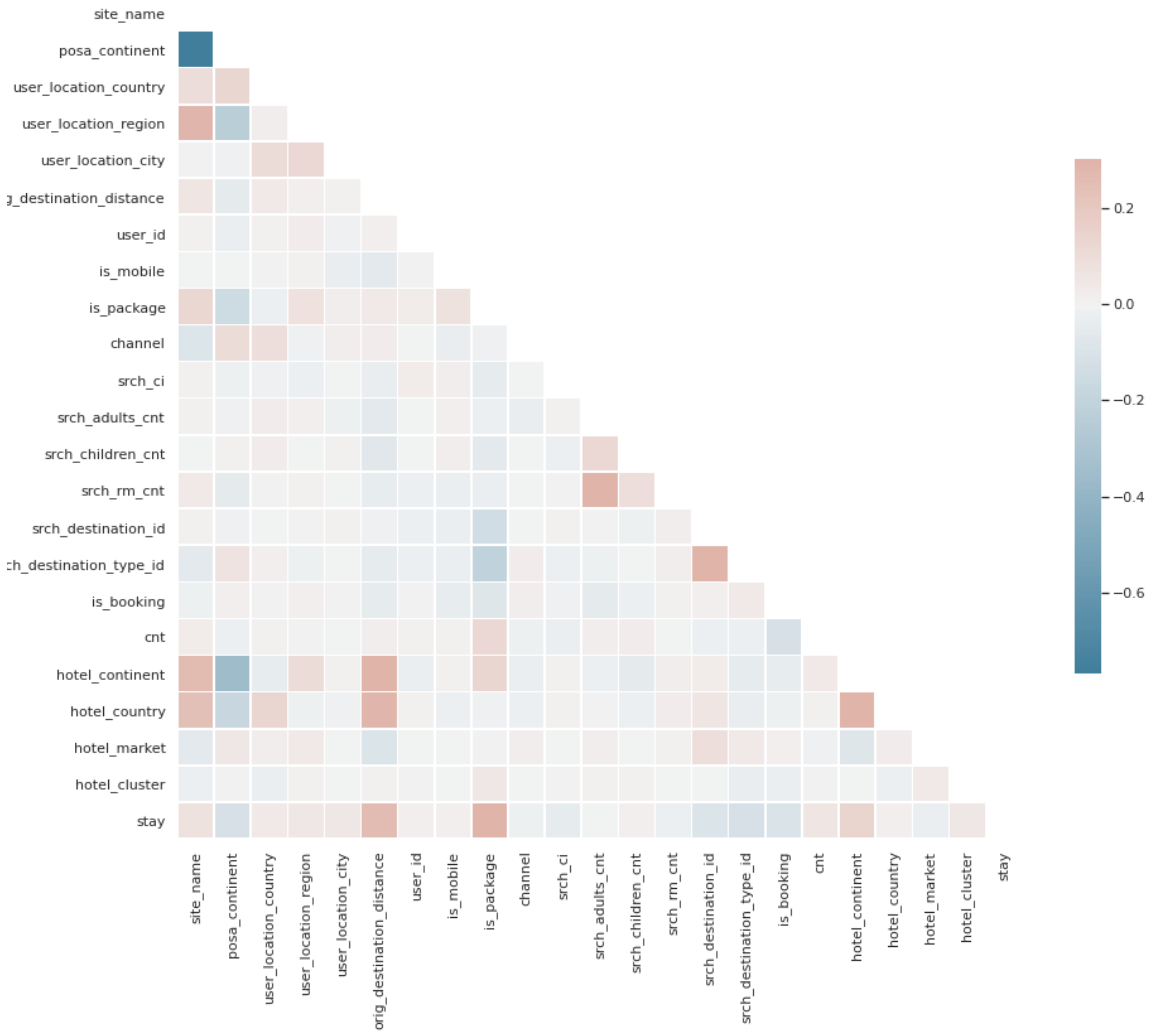

Figure 2: Correlation among features

Indeed, the distribution of people travelling with a a package is very different from the distribution of people travelling without one (**Figure 3**).
The variable orig_destination_distance is the physical distance between the customer and the hotel at the time of the search. Since the distribution of stay is skewed towards short duration, the distribution of orig_destination_distance will be as well because people don't travel e.g. 8000km to stay one day (**Figure 4**). Moreover, it is reasonable to expect that on average the customers booking hotels far away from where they live will stay for longer The variable hotel_continent is positively correlated both with stay and with orig_destination_distance.

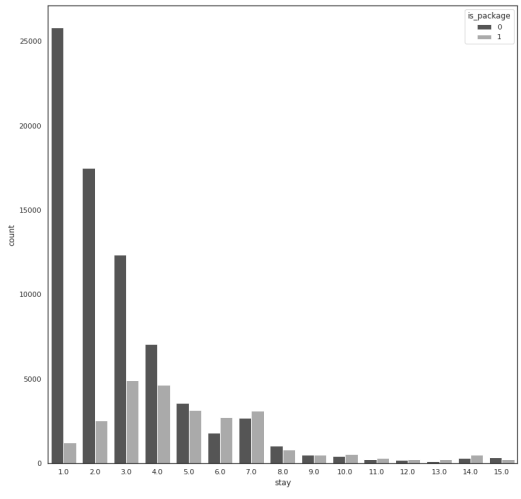

Figure 3: Distribution of stay with and without package

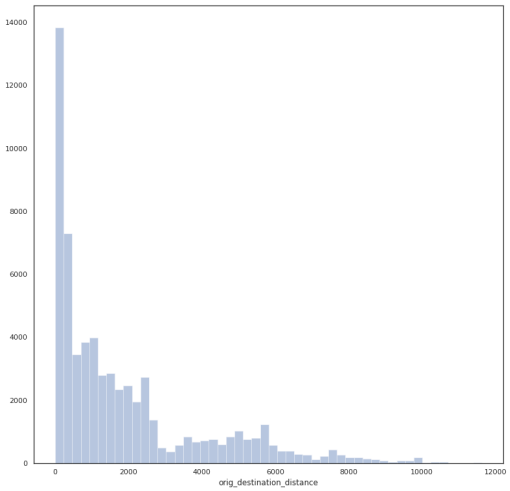

Figure 4: Distribution of orig destination distance

The continents having the highest number of tourists are Europe and North America and Asia, while the continents having less are Oceania, Africa and south America (continent 1 is missing, we suppose it to be Antarctica). However, we don't have the mapping among continent id and continents. The continent with the most short to medium stay customers is continent 2, while continent has more medium-long stay **Figure 5** (we segmented stay into 4 different classes to make the plot readable). To further explore this relation we look at **Figure 6**. The distribution of orig_destination_distance is very different depending on the hotel continent and on the stay duration. As an example, we can see that continent 2 is at the same time the most touristic, the one where people are more likely to travel short distances and the one where people prefer to stay 1-3 days.

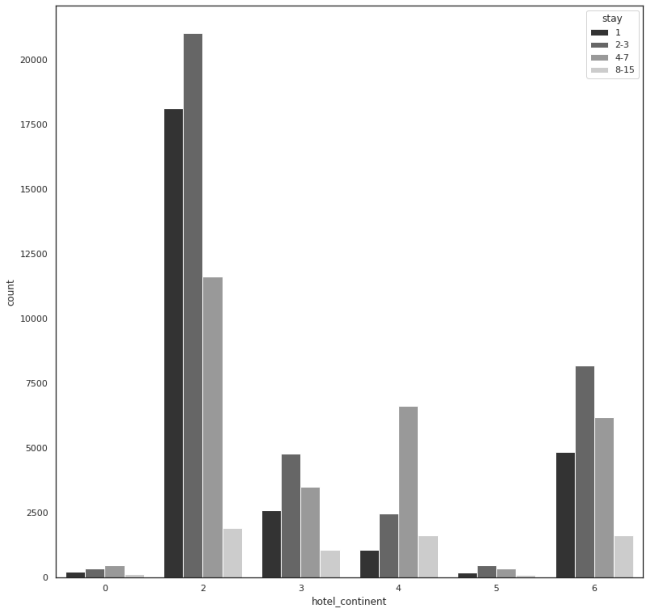

Figure 5: Distribution of hotel continent conditional to stay

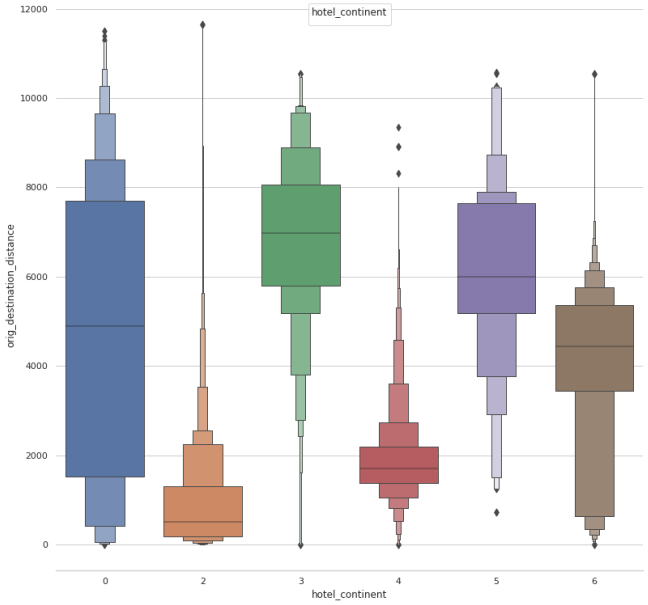

Figure 6: Distribution of orig destination distance conditional to stay

3 Feature engineering

The first step after the data visualization is features engineering. It plays an extremely important role in how the models learn, and in what they really learn. The models that we are going to use rely heavily on the consistency and quality of the data provided, therefore in this chapter we are going to describe how we have approached the problems of our dataset and how we have attempted to solve them. To start the analysis we display in **table 2** an overview of the datatset, showing the main pieces of information about the different variables.

	Column	**Non-Null Count**	**Dtype**
0	*date_time*	100008 non-null	object
1	*site_name*	100008 non-null	int64
2	*posa_continent*	100008 non-null	int64
3	*user_location_country*	100008 non-null	int64
4	*user_location_region*	100008 non-null	int64
5	*user_location_city*	100008 non-null	int64
6	*orig_destination_distance*	65069 non-null	float64
7	*user_id*	100008 non-null	int64
8	*is_mobile*	100008 non-null	int64
9	*is_package*	100008 non-null	int64
10	*channel*	100008 non-null	int64
11	*srch_ci*	100008 non-null	int64
12	*srch_adults_cnt*	100008 non-null	int64
13	*srch_children_cnt*	100008 non-null	int64
14	*srch_rm_cnt*	100008 non-null	int64
15	*srch_destination_id*	100008 non-null	int64
16	*srch_destination_type_id*	100008 non-null	int64
17	*is_booking*	100008 non-null	int64
18	*cnt*	100008 non-null	int64
19	*hotel_continent*	100008 non-null	int64
20	*hotel_country*	100008 non-null	int64
21	*hotel_market*	100008 non-null	int64
22	*hotel_cluster*	100008 non-null	int64
23	*stay*	100008 non-null	float64

Table 2: Summary content information about the subset of the original dataset

As we mentioned in the introduction, our target was built by subtracting the check-out and check-in dates. The check-out column was removed while we only kept information about the month from the check-in feature, which is now called srch_in and it's a categorial variable. The dataframe contains a total of 23 features and is composed by a subset of 150,000 samples randomly extracted from the original dataset. We split our dataset in train, validation and test set. The testing set contains 50,000 samples (33% of the total) and the validation set contains 15,000 samples (10% of the total). There is also a column containing missing data, which will be taken care of in the next paragraph.
This section is organized as follows: first we take care of the missing data, then we proceed to explain the encoding, normalization and scaling. Finally we analyze the distribution of target classes.

3.1 Missing data

Missing data represents a serious problem for every machine learning algorithm, because having a lot of missing values for a feature causes the data to be noisy and impacts the model performance negatively. We impute the missing data in the column *orig_destination_distance* using k-nearest

neighbor algorithm. The information contained in this column is obviously the distance between the customer and the hotel at the time of the booking. Since it is a numerical continuous feature, a possible choice could be using the mean to replace missing values. However considering the plots of the **Figure 5** and of the **Figure 6** of the previous chapter we can see that the distributions *orig_destination_distance* are very different depending on the hotel continent and on the stay duration. Using the mean would cause a substantial loss of information and would not take into account the intrinsic variability if the feature. Clearly the imputation was carried out after the split of the dataset into train, dev and test set, in order not to have information leakage.

3.2 Encoding

After having replacing the missing values we have encoded the categorical features. We have adopted different strategies depending on the column in question, choosing the most suitable one. For some features we adopted the simple one-hot encoding. For most of the features, one hot encoding was not a feasible strategy and would have caused exponential columns explosion, since features had up to 3000 different categories. For those columns we have used a more sophisticated strategy, which is Leave One Out encoding. It encodes the value with the mean of the target variables for all the samples, but the value itself, containing the same value for the categorical feature variable which is to be encoded, all without causing response leakage. We report now a summary table (**table 3**) with our choices:

Columns	**Encoding**
'site_name', 'user_location_country', 'user_location_region', 'user_location_city', 'user_id', 'srch_destination_id', 'hotel_country', 'hotel_market', 'hotel_cluster'	Leave One Out
'posa_continent', 'channel', 'srch_destination_type_id', 'hotel_continent'	One Hot

Table 3: Encoding Strategy for each column

3.3 Normalization

Scaling and normalization is an important phase to improve numerical stability and the accuracy of the model. After analyzing our dataset we decide not to heavily modify our columns since most of them have been encoded and are ordinal variables. Thus, we have only scaled *orig_destination_distance* by removing the mean and scaling to unit variance.

3.4 Classes Balancement

We have already shown the distribution of the target classes in the **Figure 1**, and they look very unbalanced. This is good because confirms that data reflects reality, since most people travel only for a short period of times and not for 15 days. We decide neither to under-sample the majority classes nor to over-sample minority classes, since we didn't want to lose information or to add noise to the samples. We simply merge stay ≥ 15 days in a unique class, and we have opted for a cost-sensitive learning strategy, using class weights in each model to address unbalancement.

4 Model selection

Once the data has been preprocessed we feed it to various models and assess their performance according to different metrics we think are important in order to have a model yielding to good predictions. Each metric has been chosen for a particular reason which will be discussed in the relative paragraph. The five models we have tested are:

- Decision Tree Classifier
- Random Forest Classifier
- K-Nearest Neighbor with K = 5
- Support Vector Machine with rbf kernel
- Multi Layer Perceptron Network

The architecture of the deep network is composed by 4 stacked dense layers, with relu activation function to model non-linearity patterns. We use dropout of 0.25 in each layer and early stopping with a patience of 10 epochs to prevent overfitting. Moreover we have chosen experimentally after several attempts, to use a small batch size and a learning rate of 10^{-4}. After having analyzed the results obtained testing the models with the development set, we choose the most promising one and tune its hyper-parameters more accurately collect benchmark using the final test set.

4.1 Cross-validation

The first technique we have adopted to evaluate the goodness of our model is cross validation. We split the training set in 5 fold. Each fold is selected once to be the testing set and the model is trained on the remaining four. Since some target classes are very rare, k-fold will help us reducing thge risk of selecting an "unlucky" training set. We have done this for all the models except for the Multi Layer Perceptron network due to computational reasons. For it we used fixed split in train and validation set. We report in the table below the results:

Model	**N° Folds**					**Mean**	**Max**	**95% Confidence Int.**
	1	2	3	4	5			
Tree	0.679	0.681	0.676	0.683	0.682	0.680	0.682	± 0.005
Random Forest	0.743	0.746	0.734	0.748	0.741	0.742	0.748	± 0.011
K-NN	0.584	0.579	0.567	0.574	0.577	0.576	0.584	± 0.012
SVM	0.383	0.390	0.392	0.394	0.385	0.389	0.394	± 0.010
MLP	0.344							

Table 4: Weighted F1 Score with cross validation

The table reports the weighted F1 score for each model and for each fold, then we consider the mean and the maximum values in the entire folds. Analyzing the table we can see that the Random Forest is the model with best performances, while deep model is the poorest preforming one.

4.2 Learning curve

Since we are only using a small subset of the whole dataset, it is important to see if the performance is plateauing due to the models themselves or if more data could lead to better results. This is done with the learning curve shown in **Figure 7**. This curve describes the variations of the learning process of each model with respect to the amount of data provided. As it is possible to see, once

again random forest is the best model, and also decision tree and k-nearest neighbor perform well. Other models have generally a poor performance, regardless of the amount of training data provided. Additionally, the F1 score hasn't reached yet a plateau therefore more data would definitely improve the performance.

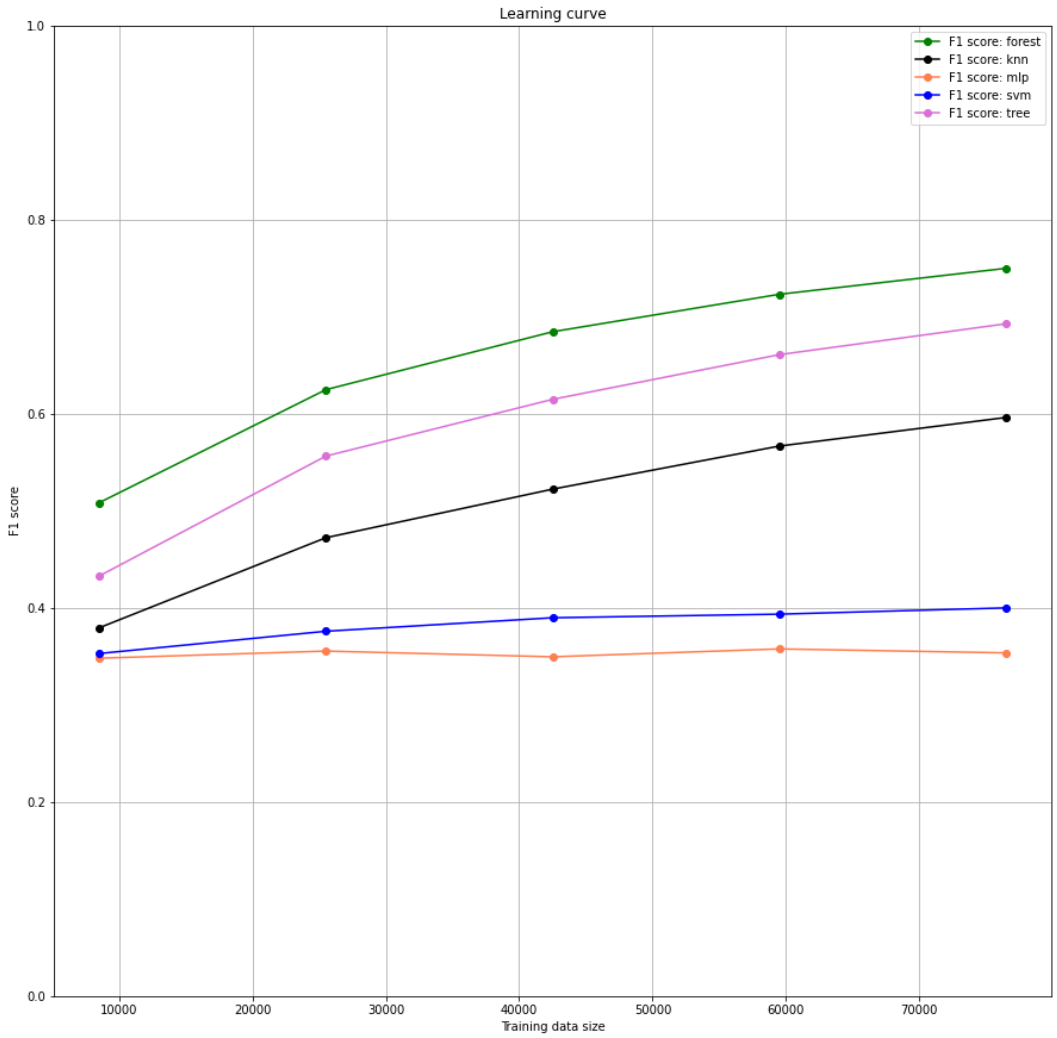

Figure 7: Learning curve for each considered model

4.3 Confusion matrix

Another technique used to evaluate the performance of a model is the confusion matrix, in which we can count the well classified samples. We can also know how model miss-classify respect the various classes. We report in the next page (**Figure 8**) the results that we obtain testing each model with the development set. Once again random forest, decision tree and knn have less sparse confusion matrix than the mlp and the svm and perform generally very well

Figure 8: Confusion Matrices on development set for each considered model

4.4 Jensen - Shannon distance

Another metric we have used to assess the performance of different models is the Jensen - Shannon distance. The Jensen Shannon distance between two distributions p and q is $\sqrt{\frac{KL(p||m)+KL(q||m)}{2}}$ where $KL(\cdot||\cdot)$ is the Kullback-Leibler divergence and m is the point-wise mean of p and q.
We have used this metric because we want to assess "how bad" the model mistake is. Miss-classifying 1 day with 2 days is definitely not as bad as miss-classifying e.g. 1 day with 15 days. This is not due to classes unbalancement, since even if all the classes had the same weights, the magnitude of the difference between the two aforementioned errors would still be the same. In **Figure 9** e show an graphical representation of our argument. The classes 2 to 5 all have the same weight, however predicted distribution 1 is closer (in a Jensen-Shannon way) to the true distribution even if test 1 and test 2 both contains 5 errors (highlighted by the black arrows). The F1 score for those two predictions would be the same. It is then necessary to use the Jensen - Shannon distance
In **Table 5** the distances between the predictions of different models on the validation set and the truth are displayed. The Tree and the Forest both have really low distances.

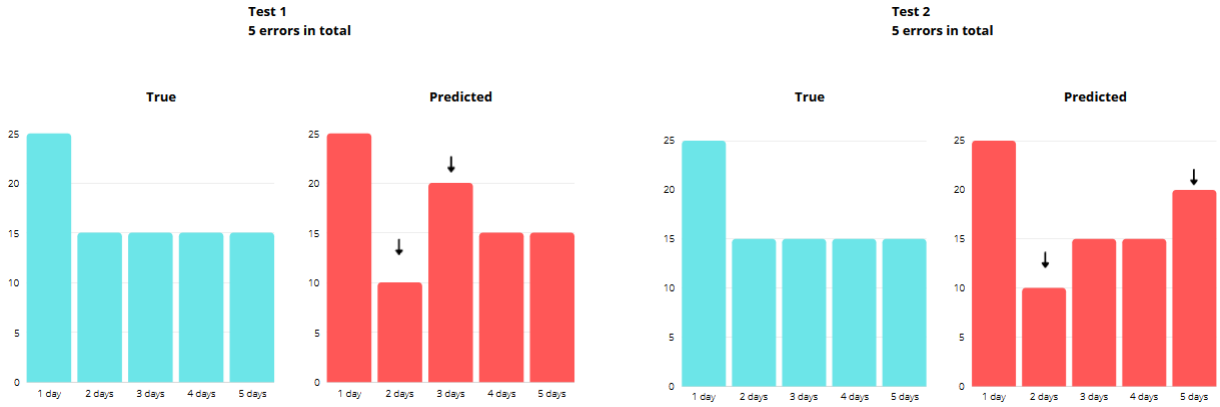

Figure 9: Example of distance between distributions

Model	Jensen - Shannon distance
MLP	0.14
Tree	0.01
KNN	0.07
SVM	0.04
Forest	0.02

Table 5: Jensen - Shannon distance for each model respect validation set

4.5 Classification report

The last and most classic evaluation metric is the classification report. The precision, recall and f1 score values are reported for each target class, also the average and weighted results are printed. The results are always consistent with what we have seen so far.

Classification report - tree :

	precision	recall	f1-score	support
1.0	0.79	0.76	0.78	4099
2.0	0.70	0.72	0.71	3075
3.0	0.67	0.69	0.68	2587
4.0	0.64	0.65	0.64	1756
5.0	0.60	0.59	0.59	1018
6.0	0.62	0.61	0.62	667
7.0	0.70	0.66	0.68	862
8.0	0.58	0.60	0.59	256
9.0	0.56	0.59	0.57	162
10.0	0.61	0.57	0.59	126
11.0	0.51	0.57	0.54	75
12.0	0.50	0.57	0.53	56
13.0	0.38	0.47	0.42	57
14.0	0.67	0.61	0.64	108
15.0	0.60	0.58	0.59	98
accuracy			0.69	15002
macro avg	0.61	0.62	0.61	15002
weighted avg	0.69	0.69	0.69	15002

Classification report - forest :

	precision	recall	f1-score	support
1.0	0.80	0.85	0.82	4099
2.0	0.75	0.75	0.75	3075
3.0	0.72	0.74	0.73	2587
4.0	0.72	0.72	0.72	1756
5.0	0.75	0.64	0.69	1018
6.0	0.72	0.66	0.69	667
7.0	0.76	0.73	0.74	862
8.0	0.73	0.66	0.69	256
9.0	0.74	0.62	0.68	162
10.0	0.72	0.63	0.68	126
11.0	0.62	0.56	0.59	75
12.0	0.68	0.57	0.62	56
13.0	0.49	0.46	0.47	57
14.0	0.75	0.69	0.71	108
15.0	0.77	0.66	0.71	98
accuracy			0.75	15002
macro avg	0.71	0.66	0.69	15002
weighted avg	0.75	0.75	0.75	15002

Classification report - knn :

	precision	recall	f1-score	support
1.0	0.68	0.74	0.71	4099
2.0	0.57	0.60	0.59	3075
3.0	0.55	0.56	0.56	2587
4.0	0.53	0.53	0.53	1756
5.0	0.57	0.48	0.52	1018
6.0	0.56	0.47	0.51	667
7.0	0.64	0.52	0.58	862
8.0	0.62	0.52	0.57	256
9.0	0.57	0.42	0.48	162
10.0	0.61	0.53	0.57	126
11.0	0.51	0.45	0.48	75
12.0	0.55	0.41	0.47	56
13.0	0.60	0.37	0.46	57
14.0	0.64	0.52	0.57	108
15.0	0.74	0.52	0.61	98
accuracy			0.60	15002
macro avg	0.60	0.51	0.55	15002
weighted avg	0.60	0.60	0.60	15002

Classification report - svm :

	precision	recall	f1-score	support
1.0	0.66	0.55	0.60	4099
2.0	0.37	0.39	0.38	3075
3.0	0.33	0.30	0.31	2587
4.0	0.28	0.34	0.31	1756
5.0	0.22	0.27	0.25	1018
6.0	0.30	0.25	0.28	667
7.0	0.40	0.28	0.33	862
8.0	0.22	0.40	0.29	256
9.0	0.22	0.33	0.26	162
10.0	0.27	0.40	0.32	126
11.0	0.21	0.41	0.28	75
12.0	0.30	0.48	0.37	56
13.0	0.29	0.39	0.33	57
14.0	0.50	0.48	0.49	108
15.0	0.27	0.51	0.35	98
accuracy			0.39	15002
macro avg	0.32	0.39	0.34	15002
weighted avg	0.41	0.39	0.40	15002

Classification report - mlp :

	precision	recall	f1-score	support
1.0	0.59	0.59	0.59	4099
2.0	0.33	0.25	0.28	3075
3.0	0.28	0.23	0.26	2587
4.0	0.22	0.26	0.24	1756
5.0	0.16	0.23	0.19	1018
6.0	0.16	0.28	0.20	667
7.0	0.33	0.12	0.18	862
8.0	0.10	0.21	0.13	256
9.0	0.00	0.00	0.00	162
10.0	0.03	0.02	0.03	126
11.0	0.02	0.09	0.04	75
12.0	0.00	0.00	0.00	56
13.0	0.09	0.18	0.12	57
14.0	0.31	0.48	0.38	108
15.0	0.16	0.22	0.18	98
accuracy			0.33	15002
macro avg	0.18	0.21	0.19	15002
weighted avg	0.34	0.33	0.33	15002

Figure 10: Classification report on development set for each model

5 Model tuning and evaluation

Based on the results collected from the benchmarks on the validation set, and after all the considerations explained, we think the random forest is the most suitable model for this task. The reasons of the poor performance of the support vector machine is mostly related to the fact that the data is very noisy, and therefore the margins can't be selected effectively. Regarding the KNN, we think the issues ar due to the sparsity of the data, and hence it's hard to cluster the rarest classes. The MLP has the worst performance of all models, this haoppens because neural networks ususally perform bad on tabular data.
Tree models are the best for this problem, with the forest (as expected) outperforming the individual tree. being an ensable method, the random forest classifier is robust to noise, and since it's trees are build using bootstrapping and by selecting a random vector of features for each node, it is also robust to variations in the distribution of class labels. Having chosen this model, we proceed to do a computationally intensive grid search to find the best hyper-parameters. The grid is as following:

Parameters	**Values**		
n_estimators	10	100	200
criterion	entropy	gini	
max_depth	10	20	None

Table 6: Grid Search settings

After a long run the best combination of parameters that we discovered is: *'criterion': 'gini', 'max_depth': None, 'n_estimators': 200*. The mean of the f1 weighted score with cross validation on the training set is **0.753**.
We benchmark our final model with test set, that represent a 33% respect the 150000 initial samples, for a total of 50000 samples.There follows the classification report and the confusion matrix:

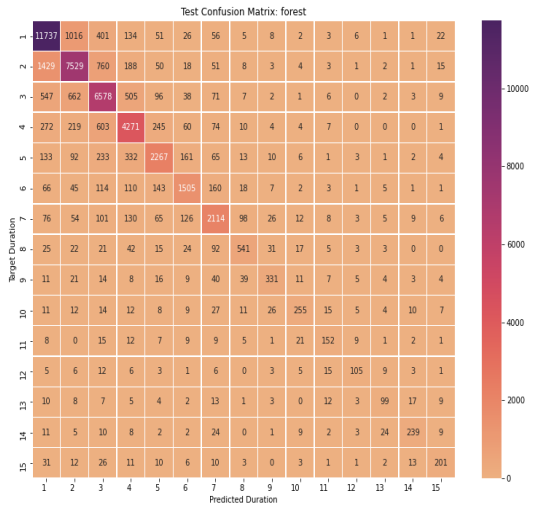

Figure 11: Final confusion matrix

Classification report:

	precision	recall	f1-score	support
1.0	0.82	0.87	0.84	13469
2.0	0.78	0.75	0.76	10062
3.0	0.74	0.77	0.75	8527
4.0	0.74	0.74	0.74	5770
5.0	0.76	0.68	0.72	3323
6.0	0.75	0.69	0.72	2181
7.0	0.75	0.75	0.75	2833
8.0	0.71	0.64	0.68	841
9.0	0.73	0.63	0.68	523
10.0	0.72	0.60	0.66	426
11.0	0.63	0.60	0.62	252
12.0	0.71	0.58	0.64	180
13.0	0.61	0.51	0.56	193
14.0	0.79	0.68	0.73	349
15.0	0.69	0.61	0.65	330
accuracy			0.77	49259
macro avg	0.73	0.67	0.70	49259
weighted avg	0.77	0.77	0.77	49259

Figure 12: Final classification report

The F1 weighted score on the test set is 77%, even higher than the best score on the training set obtained during the grid search. We want to emphasize that the test set has been split at the very beginning of the work and all the precautions to avoid target information leak have been taken. Therefore we can consider the final score on the unknown data accurate and truthful.

6 Explanability

6.1 Game theory and machine learning

In the Proposed methods section we mentioned that we used SHAP for explanability. In this section we briefly introduce the SHAP library to give some context, and then we proceed to use it to interpret our model.
SHAP stands for Shapley additive explanations and is a game theoretic approach to explain the output of any machine learning model. *Given a cooperative game* (that is a game with competitions among groups of players) the Shapley value *assigns a unique distribution among the players of a total surplus* (payout/prize) *generated by the coalition of all players.* In machine learning we consider each feature value of an instance as a player in a game (the model) where the prediction is the payout, hence *"assigning a unique distribution among the players of a total surplus generated by the coalition of all players"* means generating a criteria on how to fairly distribute the prize among the features.

6.2 Interpreting the model

For graphical reasons, we considered a subset of the problem having only 8 classes, where all instances of classes 8 to 15 of the original problem are grouped into a single class, which is class 7. Moreover for classes 0 to 6, we have that *class n = the customer stay is n+1* so e.g. class 0 corresponds to 1 day stay.
In **Figure 13** we can see the average impact of features on model output. The features are ordered by the most important to the least important. For each feature, the individual class bar length is proportional to that feature importance for the class. For example, we have that user_id is overall the most important feature, however it's not as important as srch_destination_id when it comes to predict class label 0 (1 day of stay). At the same time, we can see that user_id is approximately equally important for class 2 to 7, but it is much more important for classes 0 and 1 than for any other class. the top 3 classes that on average get the biggest "chuck" of each feature are classes 0, 1 and 7. We then give a more in depth analysis of the model predictions for classes 0 and 7 (since classes 0 and 1 are very similar we skip class 1).

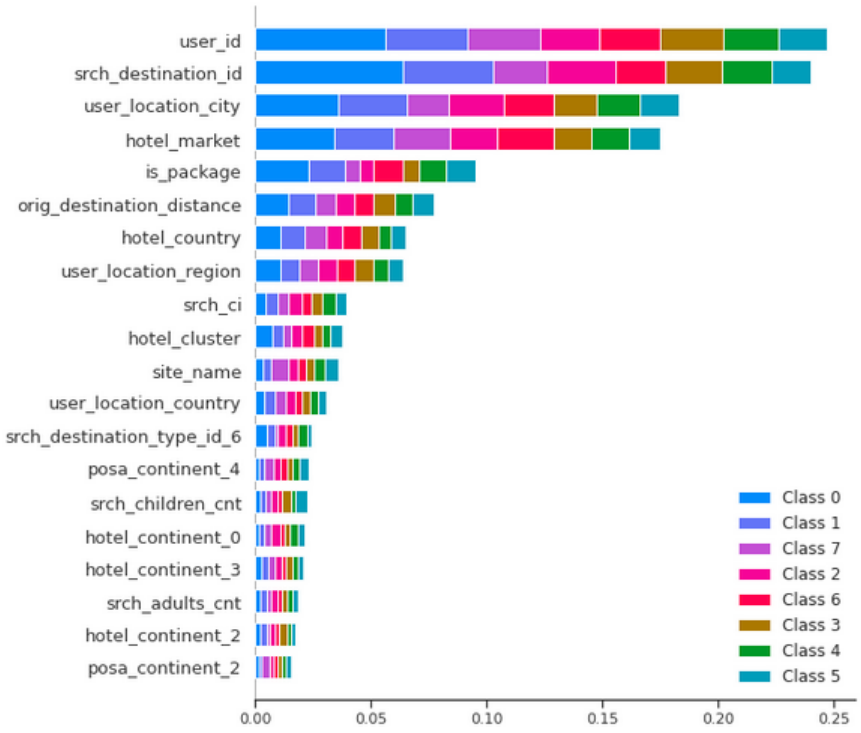

Figure 13: Global features contribution

In **Figure 14** it is possible to see the distribution of data belonging to the category stay 1 day for different features. The features are ordered by importance for that class label, meaning that "locally" (for 1 day stay) srch_destination_id is more important than user_id, despite the opposite being true globally. The higher the SHAP values for a feature, the higher is that the feature contribution to a positive prediction for the label. As an example to explain how to read the plot, we consider the feature is_package: when the feature value is high (red color), the feature has a negative SHAP value (x-axis). This means that high value of the feature contributes to "drift" the prediction away from the 1 day stay , whereas low values of the feature (blue color) contributes to "bring" the prediction towards 1 day stay. In this case the feature is binary so the separation of colors is abrupt: is_package = 0 (low values = blue color) means that the data instance is likely to belong to the category 1 day stay, while is_package = 1 (high values = red color) means that the data instance is less likely to belong to the category 1 day stay and more likely to belong to another category. Hence, customers without a package are expected to stay one day. Consider now orig_destination_distance: shorter distances (blue, positive shap value) mean that the customer is more likely to stay 1 day, as supposed at the beginning of the analysis.
Concerning **Figure 15**, it is interesting to notice that is_package is not as important for this features as it was expected to be from the correlation analysis done in the data exploration part: it's the 9^{th} feature by importance, despite it being the most correlated with stay. Moreover, the fact that high and low values for the feature are mixed we can see having a package is not "distinctive" for users staying more than 7 days.

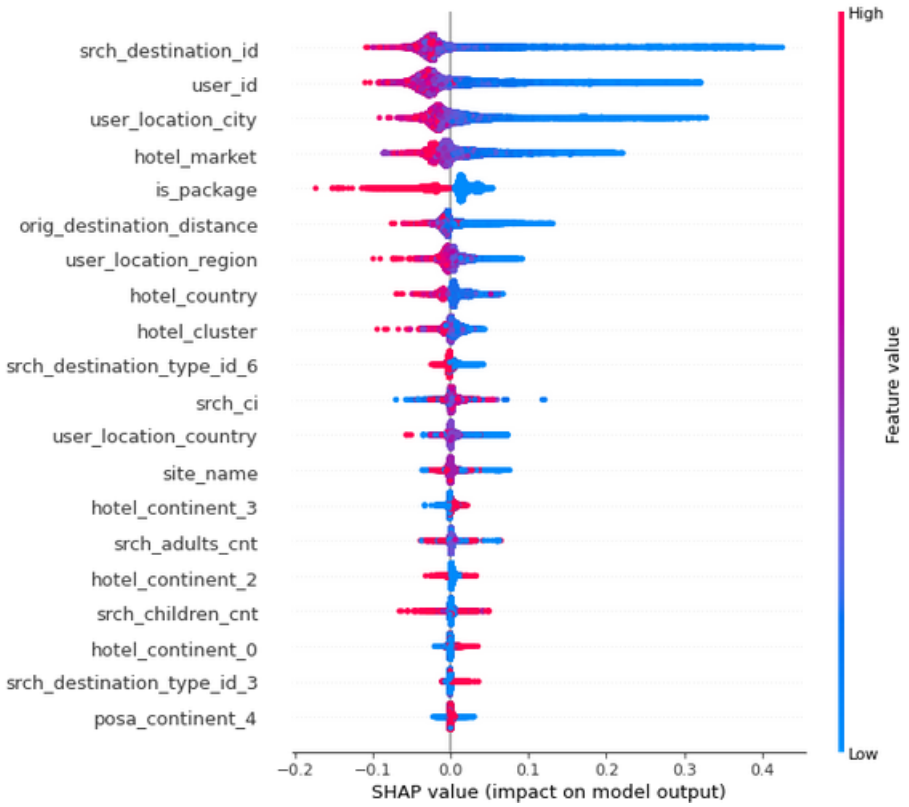

Figure 14: Local feature contribution: 1 day stay

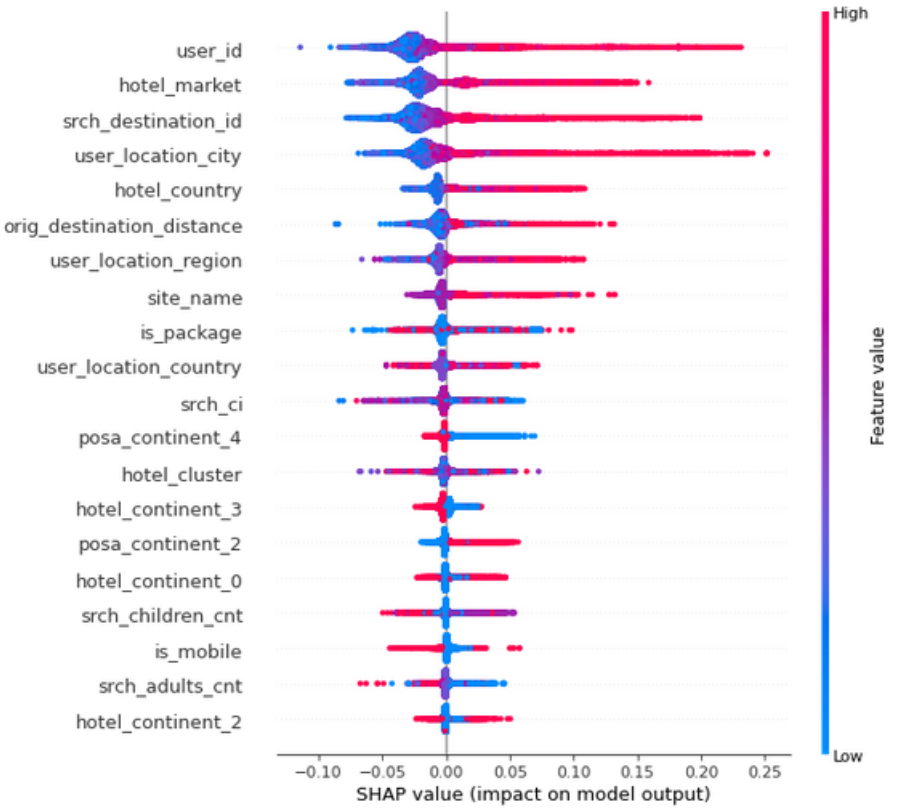

Figure 15: Local feature contribution: 8-15 days stay

7 Final considerations

Coming up with our own original problem and working on it having no external benchmark for the results was very interesting, instructive and challenging. Whenever we got a result we asked ourselves: "can we do better?". Having no way to compare our results with others has pushed us to continuously improve and to constantly search for clever ways to squeeze more percentage points of performance from our models. However, the size of the complete dataset is massive, and using more than what we have used goes beyond our computational means (it was too much even for google colab). The next step would be to use much more data, since as it is possible to see from the learning curve plot, adding more data would definitely help in improving models performances.