These are the steps that we have until now:

- Understanding the data / pre-visualizations

- Features extraction / Data engineering

- Final visualizations

- Split the data in training, test and validation set

- Balance the data and data preparation for each algorithm (down-sampling)

- Modeling

            - Random forest

            - Logistic regression

            - Lambda MART

- Use validation set to choose the hyperparametrs with cross-validation

- Accuracy calculation (test set) / comparison of the models

1. We would like to have feedback of the steps.

 The general process looks good

2. Do we have to do a normalization of the numerical data? If so, between which steps would it work best? And what type of normalization?

 What you find, reference and argue will be important for your report

I haven’t seen any type of normalization, like mean centering or so, only data engineering like:

For random forest:

* all strings need to be converted
* any missing data needs to be filled
* Chen, Chao, Andy Liaw, and Leo Breiman. "Using random forest to learn imbalanced data." University of California, Berkeley (2004).

For Logistic regression:

* don't need to normalize your data for logistic regression
* Sun, Yanmin, et al. "Cost-sensitive boosting for classification of imbalanced data." Pattern Recognition 40.12 (2007): 3358-3378.

For Lambda MART:

Also:

* Dummy all categorical variables?

Actually we have over 80% categorical variables presented in numeric format.

And some of them have more than 100 categories. It's inefficient to set them into dummy variables. While numeric format will apparently create errors.

Try only to use the most frequent and "other" values in each column to dummy code. Also you can try to use a hashed model matrix.

Regarding steps: refer to answer 3 (data leakage).

3. Regarding missing values, do we have to fill them after of before splitting the data? We think that for example if we do the mean of a variable to fill the missing values of this variable, it would be more accurate to do it before (with less bias), but at the same time if we do it before, the test and validation set, which are supposed to be independent from the train set, then they won’t.

There's a lot of strategies to deal with NaN values, I'll be assessing your justification for selection  
Type of leak occurs when you aren't careful distinguishing training data from validation data. For example, this happens if you run preprocessing (like fitting the Imputer for missing values) before calling train\_test\_split. Validation is meant to be a measure of how the model does on data it hasn't considered before. You can corrupt this process in subtle ways if the validation data affects the preprocessing behavoir.. The end result? Your model will get very good validation scores, giving you great confidence in it, but perform poorly when you deploy it to make decisions.

<https://www.kaggle.com/dansbecker/data-leakage/code>

4. We want to leave out the hotels with missing values (new hotels), because we cannot cluster them. Is that correct or we are missing data there, and therefore we should fill the missing values with, for example, the worst case scenario?

Again, a few approaches (the section on business understanding should reveal what the top winners did on this one I believe)

I would do two hotel clustering:

* C1: with no search data information, thus with all the hotels (new ones included)
* C2: with all the hotel data, thus no new hotels included.

I would base the ranking on the C2, but once in a while (randomly) include a result from C1, to include the new hotels.

5. We used a 15% for the validation, 15% for the test and 70% for the training set, is that correct? We are not sure if we can do this for the lambda MART algorithm, because it’s a learning-to-rank model.

You can certainly have a training, test and validation set for a lambdamart model  
Common proportions are 70%/30% training/validation

Yes, we can do that for lambda mart (PAPER lambda mart)

6. The data balance has to be different for every algorithm that we want to apply?

Again, a few approaches (the section on business understanding should reveal what the top winners did on this one I believe)

You can oversample your minority class examples by simply duplicating them, or you can use the [SMOTE](https://www.jair.org/media/953/live-953-2037-jair.pdf) algorithm ([DMwR](http://cran.r-project.org/web/packages/DMwR/DMwR.pdf) package in R, function SMOTE), that generates synthetic minority class examples while downsampling the majority category at the same time. Since you have a pretty high number of cases, downsampling should not lead to too much concept loss, but of course, you'll still be losing a bit of information, which is not ideal.

Stochastic Gradient Tree Boosting is a little bit less sensitive to relative class imbalance than Random Forests (since the focus is gradually put on misclassified cases). With Random Forests, two [strategies](http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf) have been developed to cope with class imbalance, they involve resampling and weighting (PAPER OF RANDOM FOREST)

<https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset>

### 3) Try Resampling Your Dataset

You can change the dataset that you use to build your predictive model to have more balanced data.

This change is called sampling your dataset and there are two main methods that you can use to even-up the classes:

1. You can add copies of instances from the under-represented class called over-sampling (or more formally sampling with replacement), or
2. You can delete instances from the over-represented class, called under-sampling.

These approaches are often very easy to implement and fast to run. They are an excellent starting point.

In fact, I would advise you to always try both approaches on all of your imbalanced datasets, just to see if it gives you a boost in your preferred accuracy measures.

You can learn a little more in the the Wikipedia article titled “[Oversampling and undersampling in data analysis](https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis)“.

#### Some Rules of Thumb

* Consider testing under-sampling when you have an a lot data (tens- or hundreds of thousands of instances or more)
* Consider testing over-sampling when you don’t have a lot of data (tens of thousands of records or less)
* Consider testing random and non-random (e.g. stratified) sampling schemes.
* Consider testing different resampled ratios (e.g. you don’t have to target a 1:1 ratio in a binary classification problem, try other ratios)

### 4) Try Generate Synthetic Samples

A simple way to generate synthetic samples is to randomly sample the attributes from instances in the minority class.

You could sample them empirically within your dataset or you could use a method like Naive Bayes that can sample each attribute independently when run in reverse. You will have more and different data, but the non-linear relationships between the attributes may not be preserved.

There are systematic algorithms that you can use to generate synthetic samples. The most popular of such algorithms is called SMOTE or the Synthetic Minority Over-sampling Technique.

As its name suggests, SMOTE is an oversampling method. It works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances.

Learn more about SMOTE, see the original 2002 paper titled “[SMOTE: Synthetic Minority Over-sampling Technique](http://www.jair.org/papers/paper953.html)“.

There are a number of implementations of the SMOTE algorithm, for example:

* In R, the [DMwR package](https://cran.r-project.org/web/packages/DMwR/index.html) provides an implementation of SMOTE.

7. For the down-sampling we are thinking about leaving a part of unbooked and unclicked values out, or randomly select the ones that we want to keep in. But for the lambda MART we are not sure about this part.

What you find, reference and argue will be important for your report

(Answer 6)

Other remarks:

* Cross-Validation in R

<http://www.milanor.net/blog/cross-validation-for-predictive-analytics-using-r/>

* general process and justifications

<https://www.dataquest.io/blog/kaggle-tutorial/>

* Why random forest does not work well? -> Data leakage, variable election

<https://www.kaggle.com/c/expedia-hotel-recommendations/discussion/20831>

* PCA: option to reduce number of variables

<https://www.r-bloggers.com/computing-and-visualizing-pca-in-r/>

* There is a leak in the data and it has to do with the orig\_destination\_distance attribute.

Correct, cluster assignments can change. For example, in winter a hotel is cheap and not-popular and will be assigned to cluster A, and in summer the same hotel will become expensive and popular and will be assigned to cluster B. In addition to yearly seasonality, we can observe weekly seasonality as well (hotels are cheaper and not popular in weekdays, and more expensive and popular during weekends).

I think our data is without the leaking, that is why we have a different dataset without clusters.