**Discussion Hopper Assignment**

**Isabelle Lefebvre**

**Problem Definition**

For a given combination of fly dates (departure, return) and search date (*t\_0),* we want to establish if the user should:

- Wait: There exists a *t*, where *t > t\_0* for which *t* has a better price

- Book now: There is no *t, where t > t\_0,* for which *t* has a better price

We need to define "better price". Two potential candidates are:

1. *min(price)*: If we expect at least one flight with a cheaper rate than today's cheapest rate, we advise to wait.

2. *avg(price)*: If we expect to see, on average, cheaper prices than today's average rate, we advise to wait.

I chose candidate (1), *min(price),* as I expect a cheaper rate on a date in the future inspires more user confidence in the product’s decision. In using *avg(price),* the average prices could go down, with no new cheaper rate. This decision could be validated by conducting user research and identifying which definition inspires most trust in Hopper (and therefore could drive customer lifetime value).

**Approach Chosen**

Consider one observation to be a combination of a search date, departure date and return date (this is the level at which we need predictions). Using those observations, I decided to build a series of machine learning models, to predict “book” or “wait” as a classification problem, based on features we know at the search result stage. I then chose the best model (based on accuracy), and use it to make predictions. Below the pros and cons of this approach

Pros:

* No cold start problem

Cons:

* Does not necessarily make use of the dependency between prices over time, to inform future prices
* This violates the iid assumption required for some those models (i.e. logistic regression)

As the dataset is imbalanced (78% of all advices are to wait), I first adjusted the dataset by oversampling the minority class, in order to obtain a dataset equally distributed between the two labels.

**Results**

The model with the best performance is a random forest model, with 500 trees and a maximum depth of 12. It achieves 83% the test set (and 85% on the train set). The most important features are the number of days ahead of departure, the departure date, as well as the number of stops on the outgoing flight.

**Next Steps**

As next steps, I would like to explore:

* Further optimizing the model selection process, by tuning the models more extensively
* Adding more features, such as external economics features, as well as interaction features, additional transformations of existing features (i.e. incorporating information on past searches, when available).
* Building a model that predicts the date a user should wait until (when the advice is to wait). The current version of Hopper already shows this, and it makes for a better user experience.
* Investigate if multilevel models can help drive higher accuracy.
* Create an accurate mapping for cabin classes and incorporate it in the model as an ordinal feature