# Описание задачи и набора данных

В качестве изучаемого набора данных был выбран набор из сайта Kaggle.com, предоставленный компанией AirBnb. Ссылка на задачу: <https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings>

В данном наборе содержались данные о пользователях, такие как ФИО, данные об используемых приложениях, устройствах, и названия зарубежных стран, проживание в которых забронировано впервые. Данные представляют собой таблицы формата csv.

Задача – определить страну, куда забронирует билет пользователь, еще не совершавший выездок за границу.

Основные файлы:

Train\_users\_2.csv – таблица данных о пользователях; включает в себя: id, дату создания аккаунта, дата первого бронирования, пол, возраст, предпочтения в языках, используемое устройство, используемое приложение, страна назначения бронирования. По этим данным будет производиться обучение.

Test\_users.csv – данные о пользователях, еще не совершавших бронирование. Именно их поведение мы и должны предсказать.

# Преобразование данных

Считывание данных производилось при помощи библиотеки **pandas** и метода **pandas.read\_csv(“имя файла»).**

Данные представляют собой таблицу пользователей и информацию о них. Размер тренинговых данных – порядка 210 тысяч элементов, тестовых – около 60 тысяч.

Из тренингового набора были вычленены данные о посещенных странах, а также идентификаторы пользователей.

Для более точного предсказания было решено объединить тренинговые и тестовые наборы данных, предварительно удалив из тренинговых столбец посещенных стран. Таким образом мы можем учитывать выборку по большему числу полей.

В ходе анализа набора данных было обнаружено, что id и дата первой брони не влияют на результат. Поэтому данные поля были удалены. Также были обнаружены ошибки в поле возраста пользователей (у некоторых данный параметр был равен 1, у других – 2014). У таких пользователей значение возраста было изменено на -1 при помощи метода **numpy.where().** Кроме того, поля **date\_account\_created** и **timestamp\_first\_active** были разбиты на отдельные столбцы по годам, мясацам и дням.

Другие поля данных, как (**'gender', 'signup\_method', 'signup\_flow', 'language', 'affiliate\_channel', 'affiliate\_provider', 'first\_affiliate\_tracked', 'signup\_app', 'first\_device\_type', 'first\_browser'),** были переведены в числовой формат при помощи метода **pandas.get\_dummies()** и заменой столбца со строковыми значениями на строки с числовыми значениями.

# Машинное обучение

Машинное обучение проводилось двумя способами: по алгоритму «случаеного леса» (Random Forest) и алгоритму градиентного бустинга (Gradient Boosting).

«Случайный Лес» — алгоритм, который для полученных данных случайным образом создает множество деревьев принятия решений и потом усредняет результаты их предсказаний. Алгоритм построения дерева очень быстр, поэтому не составляет большого труда сделать столько деревьев, сколько будет нужно. Он почти не требует конфигурации, что упрощает его использование на начальном уровне. При этом он дает довольно точные результаты.

Идея градиентного бустинга состоит в построении набора последовательно уточняющих друг друга элементарных моделей. Каждая последующая элементарная модель обучается на ошибках набора из предыдущих элементарных моделей, ответы моделей взвешенно суммируются.

Для реализации данного алгоритма в python использовалась библиотека **xgboost**. Для разбиения имен стран использовалась библиотека sklearn и класс LabelEncoder().

При этом было рассмотрено два варианта предсказания решений:

1. Нахождение точного значения
2. Нахождения списка значений

В первом случае производится выбор страны, куда пользователь попадет с наибольшей вероятностью

Во втором случае для каждой страны вероятность ее посещения пользователем вычисляется отдельно. Далее берется несколько стран с наибольшей вероятностью. Они и будут включены в конечную выборку.

# Оценка точности и времени

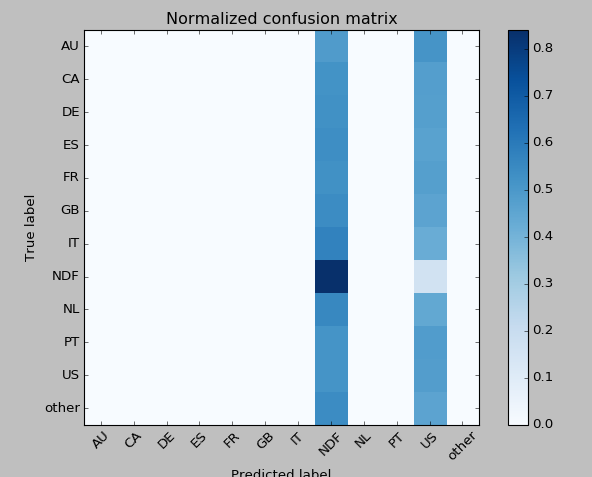
Для оценки точности был взят только набор тренинговых данных с последующим разбиением, 40% от объема было взято за тренинговые данные, 60% - за тестовые.

Итого: 85000 экземпляров на обучение и 120000 на предсказание.

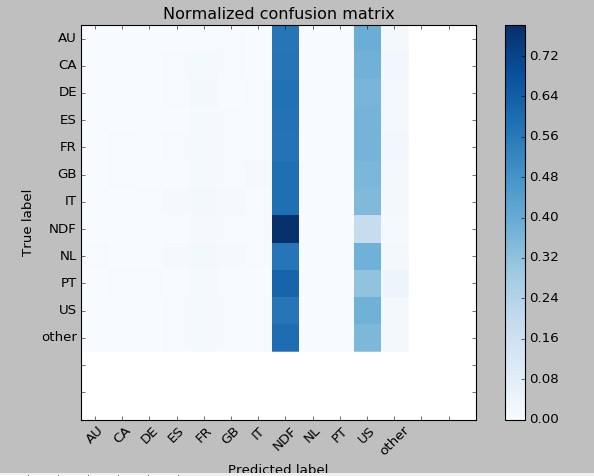
Ниже приведена таблица точности и времени для xgboost и Random Forest

|  |  |  |
| --- | --- | --- |
|  | precision (%) | time (s) |
| XGBOOST | 63,11% | 7,66 |
| XGBOOST prob | 96,45% | 0,13 |
| Random Forest | 56,87% | 1,14 |
| Random Forest prob | 93,83% | 2,18 |

Матрица выбора для xgboost:



Для Random Forest:



# Выводы

В ходе анализа точности работы алгоритма было определено, что для более точного предсказания порой достаточно брать не одно значение, наиболее вероятное при определенном наборе данных, на несколько значений, удовлетворяющих условиям в наибольшей степени. Другое дело, какие условия были наложены задачей, возложенной на программиста. Но в контексте данной задачи следующее допущение позволяет получить большую точность.

Xgboost является отличным инструментом для машинного обучения благодаря простоте для начинающих пользователей. По скорости обучения и точности он превосходит другие алгоритмы.

В свою очередь, настройка xgboost достаточно трудоемка, поэтому Random Forest является отличной альтернативой, предоставляемой в библиотеке scikit-learn.

**Precision (xgboost): 96%**

**Precision (random forest): 93%**

**Time(xgboost): 1 second**

**Time(random forest): 2 seconds**