1 - Exploratory Data Analysis

Sparkommender - Exploratory Data Analysis

```
import org.apache.spark.sql.{DataFrame, Row, SQLContext}
import org.apache.spark.sql.functions._

val sqlContext = new SQLContext(sc)

val csv = "com.databricks.spark.csv"
val csvOptions = Map("delimiter" -> ",", "header" -> "true", "inferSchema" -> "true")
```

```
val trainFile = "/Users/radek.ostrowski/git/expedia-kaggle/data/train.csv"
val testFile = "/Users/radek.ostrowski/git/expedia-kaggle/data/test.csv"
val destinationsFile = "/Users/radek.ostrowski/git/expedia-kaggle/data/destinations.csv
```

```
  3.8G  train.csv         - the training set
  264M  test.csv          - the test set
  132M  destinations.csv  - hotel search latent attributes
```

```
val trainDf = sqlContext.read.format(csv).options(csvOptions).load(trainFile).cache()
```

```
trainDf.printSchema
```

```
display(trainDf, maxPoints=5)
```

Let's look at three main components and what describes them: hotels, users and their interaction.

Hotels

Our main goal is to predict the last column `hotel_cluster`

```
trainDf.select("hotel_cluster").distinct.count
```

```
100
```

Hotels are described by their location

```
val hotelsDf = trainDf.select("hotel_cluster","hotel_continent","hotel_country","hotel_
```

```
display(hotelsDf, maxPoints=5)
```

Two most important features are `hotel_market` which describes a city and `srch_destination_id` which describes a specific destination within that city.

```
hotelsDf.select("hotel_market").distinct.count
```

```
2118
```

```
hotelsDf.select("srch_destination_id").distinct.count
```

```
59455
```

Additionally we have a file containing latent description of search destinations

```
val destinationsDf = sqlContext.read.format(csv).options(csvOptions).load(destinationsF
```

`srch_destination_id` plus 149 numeric columns

```
display(destinationsDf, maxPoints=5)
```

Users

```
val usersDf = trainDf.select("user_id","posa_continent","user_location_country","user_l
```

```
display(usersDf, maxPoints=10)
```

Most important features are `user_id` which uniquely identifies the users and `user_location_city` which is their location at the moment of search or booking.

```
usersDf.select("user_id").distinct.count
```

1198786

There are nearly 1.2 million users

```
usersDf.select("user_location_city").distinct.count
```

50447

Users and Hotels Interaction: Searches and Bookings

We also have various data related to user searches and the actual bookings of the hotel clusters.

```
val actionDf = trainDf.select("is_booking", "date_time", "srch_ci", "srch_co", "is_mobi
```

```
display(actionDf, maxPoints=5)
```

Feature `is_booking` distinguishes between a search and a booking

```
actionDf.select("is_booking").where("is_booking = 1").count
```

3000693

There are 3 million bookings

```
actionDf.select("is_booking").where("is_booking = 0").count
```

34669600

And nearly 35 million of clicks

Other important features are the time of search/booking `date_time` and check-in and check-out times `srch_ci`, `srch_co`

Note, that during the Kaggle Expedia competition a data leak was discovered where one could just match on `orig_destination_distance` and `user_location_city` and get a near perfect score for a big chunk of the data. It's not exploited here.

Build: | **buildTime**-*Sat Mar 12 18:40:47 UTC 2016* | **formattedShaVersion**-*0.6.3-61efca56c2a45613ec609f350dc4911b166b3f28-SNAPSHOT* | **sbtVersion**-*0.13.8* | **scalaVersion**-*2.11.7* | **sparkNotebookVersion**-*0.6.3* | **hadoopVersion**-*2.7.2* | **jets3tVersion**-*0.7.1* | **jlineDef**-*(jline,2.12)* | **sparkVersion**-*1.6.1* | **withHive**-*true* | **withParquet**-*true* |.