Proceedings of the IASTED International Conference
Modelling, Identification and Control (MIC 2017)
February 20 - 21, 2017 Innsbruck, Austria

DATA ANALYTICS FOR ONLINE TRAVELLING RECOMMENDATION SYSTEM: A CASE STUDY

Alessio Petrozziello and Ivan Jordanov
University of Portsmouth, Portsmouth, U.K.
Alessio.Petrozziello@port.ac.uk, Ivan.Jordanov@port.ac.uk

ABSTRACT
Nowadays, the online travel agencies (OTAs) provide the main service for booking holidays, business trips, accommodations, etc. As in all online services where users, items, and decisions are involved, there is a necessity for a Recommender System (RS) to facilitate the navigation of catalogues and websites. For a travel RS, the use of a pure collaborative filtering approach is not feasible because the user-item matrix is way too sparse. For this reason, a content-based filtering is investigated in this work, focusing on one of its main problems: missing features. An initial exploratory analysis is used to identify a class of poorly ranked properties (e.g., Vacation Rentals (VR)). To deal with the missingness in the data, several state-of-the-art imputation methods (K-NN, Random Forests, and Gradient-Boosted Trees) are investigated and their performance critically analysed and tested. These techniques are applied following dataset preprocessing that includes cleaning, feature scaling, and standardization. In addition to that, a k-fold cross validation is used to validate the imputation results and reduce the possibility of overfitting. Three similarity measures (Jaccard, Weighted Hamming and Fuzzy-C-Means rankings) based on engineered non-historical features (amenities and geographical position) are analysed and employed for determining the best proxy for unavailable features.

KEY WORDS
Data Analytics; Recommender Systems; Missing Features; K-NN; Random Forests; Fuzzy-C-Means; Big Data.

1. Introduction

As in all online services where users choices, items, and decisions are involved, there is a necessity for a Recommender System (RS). Since the online travel agencies (OTAs) provide the main service of booking holidays, business trips and, accommodations a correct recommendation is what the user can benefit mostly from the system (money-wise and satisfactory-wise). Navigating through big catalogues is a tedious and time-consuming activity and selecting the best deal among many similar offers is not a trivial task. For a RS based on the users past experience (Collaborative Filtering (CF)), all users' information and interactions with the catalogue are recorded in a *user-item* matrix in order to learn behaviours and popularity trends to give new recommendations. Since travel RS allows the user to navigate the catalogue and book hotels without being logged in the system, the use of a CF approach is not feasible because the user-item matrix is way too sparse (and sometimes even the user information is not available at all). For this reason, a Content-Based Filtering (CBF) approach is considered in this work (where the recommendation is given on items similarity), focusing on one of its main problems: missing features. An important event in the market of holidays lodging is represented by the explosive popularity of *private renting* (Vacation Rentals (VR)) gained in the past few years [1]. New VR (e.g., apartments, condos, apart-hotels, etc.) have been recently introduced in the OTAs (~40K records), and more than 160K records are expected to be added by the end of the year. The primary problem with this massive influx of new properties is the lack of related historic data which results in their unfair ranking in the system. For the new VR, the lack of features is not only historical (e.g., historical prices, purchases, popularity, etc.), but also in absents of basic characteristics (e.g., star rating and guest rating). For example, *star rating* (which is given for public properties by an *Institutional body*) is not regularized for private lodgings. On top of that, the *guest rating* will be missing for some time after the first appearance of the property, until recording at least a few users' rating. Different missing data imputation techniques [2-4] and ad-hoc feature engineering can be used to enhance the CBF, allowing more diversity and fairer ranking of new items. This investigation includes analysis of the VR market. Experiments and results of the missing features imputation and item similarities prediction are also reported. The rest of the paper is organized as follows: Section 2 describes the catalogue datasets, and the employed missing data imputation methods are reported in Section 3. Session 4 discusses the carried out empirical study and the experiments results are critically analysed and discussed in Session 5. Finally, in Section 6 conclusion and future development are given.

2. Datasets

The collected data is coming mainly from three sources:

- *Amenities*: includes the characteristics of all the properties (e.g., Wi-Fi, pool, TV, etc.);
- *Destinations*: records of all points of interest (e.g., cities, landmarks, airports, train and metro stations, etc.);
- *Properties*: comprises all the relevant information in the system (e.g., property ID, name, latitude, longitude, etc.).

The *Amenities* dataset is divided into four categories: *Dining* (e.g., restaurants, bar, etc.), *Room* (e.g., TV, Wi-Fi, etc.), *Property* (e.g., reception, elevator, etc.) and *Recreation* (e.g., spa, pool, etc.) amenities.

The cleaning of this dataset is performed in two steps. The first one filters out all the amenities that are not provided by the property or those requiring a fee (e.g., "No Free Parking", "No Free Water", "No Free Wi-Fi"). The second filter removes, for each category, the most frequently listed amenities (e.g., "Free Wi-Fi" appears in 190K out of 290K properties) and the least frequently listed ones (roughly getting rid of the top 2% and bottom 15% of the list). The obtained cleansed subset contains around 5 million records (16 amenities on average for each property) and 500 unique amenities. From the *Properties* dataset, only the active ones on the catalog are considered (~290K) and referred as *active* dataset. D*estinations* table contains all points of interest belonging to the following categories *landmark*, *airport*, *metro station* or *train station*. This set is cross-joined with the *active* properties in order to calculate the distances between each property and destination (Haversine distance [5]). The resultant set contains 200 million records of property-destination pairs.

3. Missing Data Imputation Techniques

3.1 Baselines

The most common techniques used as baselines for comparison and analysis of predictive models are Mean and Median substitution [6]. The Mean (Median) substitution replaces the missing values with the mean (median) of the same attribute of the set without missing values. Despite being fast and of straightforward implementation, these techniques are only used here to for an initial sanity check and comparative purposes because they are generally rejected by the scientific community.

3.2 K-NN Imputation

In the K-Nearest Neighbours Imputation (KNNI) the missing values are imputed applying the mean, mode or median of the K most similar patterns, found by minimizing the Euclidean Distance between a pattern with missing values and the complete subset [2]. The KNNI approach comprises three steps: take only the datasets rows without missing data and use this subset as a dataset to select the nearest neighbours; choose a distance metric and compute the nearest neighbour between each pattern with missing data and the complete subset; impute the data, using the mean or the mode of the chosen neighbours. The only parameter to be selected is the number of neighbours K and the authors in [2] argue that the method is fairly insensitive to its choice. In all the simulations carried out in this work, we used a value of K = 10. The K-Nearest Neighbours has some advantages: the method can predict both, categorical variables (the most frequent value among the KNN) and continuous variables (the average among the KNN); it has easy interpretability and straightforward implementation; furthermore, it only imputes values in the original range of values of the complete set. Disadvantages of the approach are the low scalability to big dataset, ($O(N^2)$ comparison needed) and the robustness of the results is questionable (e.g., compared to ensemble methods).

3.3 Random Forest Regression Imputation

Random forests (RF) regression [3] is an evolution of the regression trees approach where multiples models are used together (ensemble) to predict the value of the substituted variable. Furthermore, the combination of multiple decision trees helps to reduce the risk of overfitting. Due to their flexibility, scalability, and robustness, the RF are considered one of the most successful machine learning models for classification and regression tasks [7]. RF have a wide range of benefits: they can easily handle categorical, continuous, discrete and boolean features, they are not very sensitive to feature scaling, and can capture non-linearities and feature interactions without any additional effort in the data preparation. This method trains a set of decision trees separately, increasing the parallelization and scalability while adding some randomness in order to ensure that each tree is different from the others. On the test set, the prediction of each tree is combined to reduce the variance, improving the performance metrics. The randomness is usually injected through two techniques: bootstrapping from the original dataset at each iteration; or using only a subset of features for each tree. To make a prediction on a new instance of the test set, all the trees are aggregated, usually predicting as final value an average of the predictions over all trees. The RF algorithm used in this work from in the *ML Spark* library [8].

3.4 Gradient-Boosted Trees Regression Imputation

Gradient-Boosted Trees (GBTs) [4] are an ensemble of decision trees that iteratively train single trees in order to minimize a given loss function. On each iteration, the algorithm uses the current set of models (ensemble) to predict the value of each training sample which is compared to the observed label. The dataset is re-labeled to give more importance to the training samples with low prediction accuracy, hence, in the next iteration, the algorithm will put more effort in correcting those problematic instances. The re-labelling process is carried through a loss function and the final aim is to minimize it in successive iterations using the training set.

The two main loss functions for regression are the Squared Error $\left(\sum_{i=1}^{N}(o_i - p_i)^2\right)$ and the Absolute Error

($\sum_{i=1}^{N} |o_i - p_i|$), where o_i is the observed label and p_i the predicted one for a given pattern *i*.

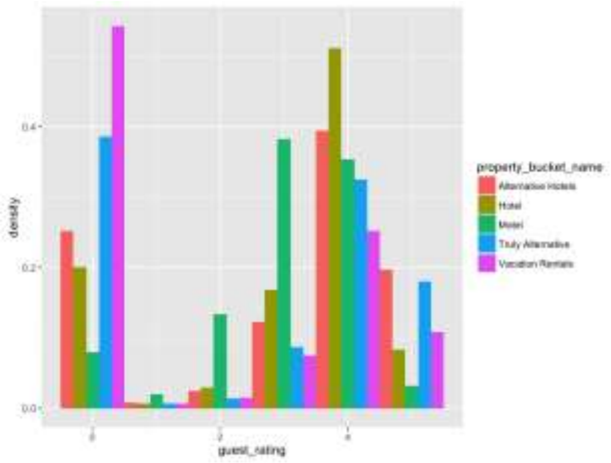

Figure 1 Distribution of *guest rating* (ranging 1 to 5 with 0 used to represent missing ratings) across all property types (e.g., 'Hotel', 'Vacation Rentals', 'Motels', etc.)

Similar to RF, GBTs can handle a variety of features (e.g., categorical, continuous, discrete and boolean), no additional data scaling is needed and they are able to capture non-linear patterns and feature relationships. Since the GBTs can overfit during the training process, a validation set should be used in order to mitigate the possibility of memorizing the data instead of learning to generalise. The training is stopped when the improvement in the validation error is less than a certain tolerance. Usually, the validation error decreases initially with the training error and increases later in the learning when the model starts to overfit (while the training error continues to decrease). The GBTs algorithm used in this work is from the *ML Spark* library [8].

4. Empirical Study

4.1 Research Objectives

- Are the VR unfairly ranked by the Recommender System? Following the exploratory analysis of the available data, we found that only 2% of all VR have *median* rank in the top 10 positions and a missing data rate of 50% (*Figure 1*) for the guest rating and 19% for the star rating (*Figure 2*).
- Can the Co-Clicks similarity be approximated with a non-historical similarity metric for new properties? The Co-Clicks similarity are compared with three other non-historical similarity metrics (i.e., Jaccard Similarity and Weighted Hamming Distance on amenities, and Fuzzy-C-Mean clustering centroid distance on geographical features). The metrics and the results of their implementation are described and analysed in Section 5.1.
- Can the guest and star ratings be imputed using non-historical features? The missing features are imputed with two baselines (i.e., Mean and Median Imputation) and three state-of-the-art techniques for missing data imputation (K-NN, RF and GBTs). The application of these methods is discussed in Section 5.2, followed by the analysis of the results.

4.2 Evaluation and Validation

A variety of metrics for comparing and evaluating data imputation and predictive models can be found in the literature [9]. Among them, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are the most popular. The MAE is argued to be more accurate and informative than the MSE [10], hence, it is used in this work to assess the results from the imputation of guest and star ratings. The MAE is easy to interpret, reflecting on average how many error stars are there between the observed and the predicted value (e.g., MAE = 0.5 means an average error of a half star). One disadvantage of this measure is its inability to specify whether the rating is underestimated or overestimated. When two lists (rankings) need to be compared (e.g., in RS), the use of Mean Average precision at X (MAP@X) is recommended [11]. The MAP@X measures the distance of a particular item between the predicted rank and its position in the target list: $\text{MAP@X} = \frac{1}{N}\sum_{m=1}^{\text{N}} P(m)$, where N is the number of items and *P(m)* is the precision at cutoff X calculated as:

$$P(m) = \begin{cases} \frac{1}{|o_m - p_m| + 1} & if \quad |o_m - p_m| \leq \text{X}, \\ 0 & otherwise \end{cases},$$

with o_m and p_m as observed and predicted item ranks respectively. For validation purposes, the dataset is split in training and test sets (70% - 30%), then using the training set we apply a k-fold cross validation (k = 10). The test set is generated using uniform sampling without repetitions, and the rest of the data is left as training and validation sets. Furthermore, the *ML Spark* pipelines [8] are used to ensure correctness and replicability of the experiments.

4.3 Big Data Tools

During the development of this research different frameworks for distributed computing on big data are used. The Hadoop framework, along with some of its successors (e.g., *Spark*), permitted the management (Sqoop, Oozie), exploration (Hive) and the implementation of scalable and distributed algorithms (*Spark*). Hive [12] is an open source data warehouse solution built on top of Hadoop which supports queries expressed in HiveQL (an SQL-like declarative language, which is translated in map-reduce jobs executed on Hadoop). The query language supports primitive types, collections, arrays, maps and nested compositions of primitive types. Hive also includes a system catalog, namely *Hive-Metastore*, containing schemas and statistics useful for data exploration and query optimization. We use it in this work mainly for data pre-processing (e.g., data filtering, aggregation, and basic statistics). Apache *Spark* [13] is an open source framework for distributed computing (driver-workers paradigm) written in *Scala* [14]. It is composed of a main core package and four components: *SparkSQL*, *ML Spark*, *SparkStreaming* and *GraphX*. The main advantage of *Spark* is that, differently from the Map-Reduce paradigm it is not built on acyclic data flow model. The main data structure is the 'resilient distributed dataset' (RDD) which is distributed and stored in the fastest part of the memory (e.g., RAM) and can be easily accessed multiple times when needed for machine learning iterative algorithms.

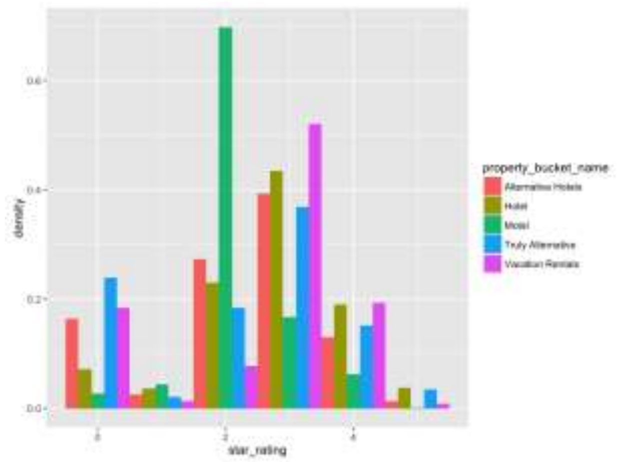

Figure 2 Distribution of *star rating* (ranging 1 to 5 with 0 used to represent missing ratings) across all property types (e.g., 'Hotel', 'Vacation Rentals', 'Motels', etc.)

Spark supports two types of operations: transformations and actions. The transformations are 'lazy' and an *execution plan* is created when they are involved [13]. They are only computed when an *action* requires a result to be returned to the driver program (e.g., print, collect, count, etc.). This design enables *Spark* to run more efficiently, since only the last result (required by the action) is returned to the main node of the cluster, reducing network bottlenecks. The fault tolerance is achieved by keeping track of the "lineage" of each RDD (the sequence of operations that produced it), so that it can be reconstructed in the case of data loss. *ML Spark* is a distributed machine learning framework built as additional module on top of *Spark*. Due to its distributed in-memory architecture (using RDDs) it is considered 9 times faster than the disk-based implementations of Mahout [8]. Many machine learning algorithms and statistical procedures are implemented in the package and can easily be used through the *Spark pipelines*.

5. Experimentation and Results

5.1 Co-Clicks Similarity prediction

Table 1 Scaled co-clicks similarity for five items within 0 and 1, where 1 means most co-clicked items and 0 - never co-clicked items.

	Item 1	Item 2	Item 3	Item 4	Item 5
Item 1	-	0.98	0.67	0.42	0.84
Item 2	-	-	0.35	0.88	0.10
Item 3	-	-	-	0.40	0.32
Item 4	-	-	-	-	1.00
Item 5	-	-	-	-	-

The *Co-Clicks* is an item-item measure which takes into account the properties co-clicked by the users in a specific time frame. Let us assume a group of 1 million users are navigating through the London properties catalog over a year. Every time a user clicks on multiple items during the search, the number counting the co-clicked pairs will increase by one. All the values are scaled within [0,1] range for estimating the similarity (*Table 1*). The main problem of this similarity measure, based on *implicit feedback* from the user, is that for each new item added to the catalogue, there will be no historical data available resulting in 0 similarity with all other items. The first objective of this research is to find an approximation function (based on non-historical features) to be used as a proxy of the *Co-Clicks* (when they are not available). To achieve this, two sets of engineered features based on *amenities* and *geographical position* are investigated. Two experiments are carried out using the *amenities*: the first one implements the Jaccard similarity metric and the second one - the Weighted Hamming Distance. The Jaccard similarity (within [0,1] interval, where 0 means completely dissimilar and 1 when identical) is defined as:

$$sim(i,j) = \frac{\cap\,(A,B)}{\cup\,(A,B)}\ ,$$

where A and B are two sets of amenities related to properties i and j. Jaccard similarity only takes into account the number of shared amenities, without giving any preference to the amenity popularity. On the other hand, the Weighted Hamming Distance (within [0, inf] range, smaller the value the greater the similarity) takes into account the number of properties in which the amenity is available:

$$sim(i,j) = XOR(A,B)\ \cdot\ W^{T}\ ,$$

where W is the amenity weight vector. Each element of W is calculated as Inverse Document Frequency (idf):

$$w_k = idf(k, D) = \log\left(\frac{N}{|d \in D : k \in d|}\right).$$

In this equation D represents all amenity sets, N is the total number of items (properties), and $|d \in D : k \in d|$ is the number of properties where the amenity k is available. For the geographical features, a Fuzzy-C-Means [15] (or soft clustering) algorithm is used to group the properties and subsequently the Euclidean Distance of their membership functions to calculate their similarities. In order to cluster the properties, a three stage *Spark* pipeline is built: *vectorization*; *standardization;* and *clustering*. The *vectorization* transforms a *Dataframe* with one column per feature, in a one column *Dataframe* containing a vector of features. The 0-mean and unit-variance *standardization* is then applied to each feature in the vector and the *clustering* groups the properties based on a given parameter *c* (number of clusters). The choice of *c* is empirically tested with values between 5 and 100 with incremental step of 5. The final choice resulted in *c* = 10 for two main reasons: the MAP@X accuracy was not significantly improving for values greater than 10; and the clusters were meaningful (e.g., each cluster containing properties in the city centre, or near commute connections, or hot landmarks, or even airports). For the Co-Clicks similarity prediction, the MAP@x (x = 5) is used as accuracy measure. The results reported in *Table 2* show the accuracy of the Jaccard similarity, Weighted Hamming distance and Fuzzy-C-Means rankings against the Co-Clicks similarity with rank = 1 and rank ≤ 5 (the first five are chosen because they are at the top of the screen). As it can be seen from *Table 2*, the accuracy using the amenities as similarity measure is very low (11% and 25% for the Jaccard similarity and 9% and 22% for the Weighted Hamming Distance respectively). The poor results can be explained by the correlation between the Co-Clicks and the position of the properties in the raking (two closely ranked items will have higher probability of being co-clicked in the same session), while the *amenities* similarity (either Jaccard or Weighted Hamming Distance) only correlates properties based on the number and type of matching amenities.

Table 2 MAP@5 accuracy scored by the three proposed similarity measures, compared only on the pairs of properties with Co-Clicks rank = 1 and Co-Clicks ≤ 5. The second column shows the probability of a property ranked 1 by the Co-Clicks similarity, to be ranked in the first 5 positions by the other similarity metrics. The third column shows the accuracy of a property ranked in the first 5 positions by the Co-Clicks similarity and all the other similarity metrics.

Method	Rank = 1	Rank ≤ 5
Jaccard Similarity Ranking	0.11	0.25
Weighted Hamming Ranking	0.09	0.22
Fuzzy-C-Means Ranking	**0.35**	**0.60**

For the Fuzzy-C-Means ranking, the achieved accuracy is much higher (35% and 60% for the two ranking groups respectively) and this could be related to the fact that usually the listed properties are geographically close (in a given search session the shown properties are from the queried city area). While the *Co-Clicks* still gives a better insights of the properties similarity, implicitly reflecting the users' behaviour and choice; the geographical clustering offers a good approximation with a 35% of correctly ranked items in first position (rank = 1), and 60% for the top 5 rankings, when the Co-Clicks similarity information is missing (due to the lack of historical data for the new properties).

5.2 Guest Rating and Star Rating Imputation

Two of the most frequently used features when selecting a property from a catalogue are the users rating and the quality of the property (e.g., star rating). Very often this information is missing when a new property is added to the website and both the user and the RS are most likely to underestimate the property. In order to provide fairer ranking, as a second objective of this research, the two ratings are imputed using available non-historical features (e.g., amenities and geographical position). From the initial dataset we take out all the instances with missing guest or star rating, which results in a complete subset (with no missing values). This set is further split into 70% training and 30% testing subsets. Then the guest and star ratings are removed artificially from the test subset. This testing subset with missing values will be used for assessment of imputation methods performance. Two baselines (Mean and Median) and three state-of-the-art approaches for missing data imputation are used in this work: K-Nearest neighbours (K-NN), Random Forests regression and Gradient Boosted Trees regression (GBTs). Firstly, the two baselines are used to calculate the Mean (Median) of the *guest rating* (*star rating*) on the training set and substituted in the test set. Secondly, the model based imputation approaches are employed. For each instance from the test set (all with missing ratings), K-NN is used to calculate the k (k = 10) most similar neighbours from the training subset. For this purpose, the Euclidean Distance on the *weighted amenities* and *geographical features* is used to find the k nearest neighbours. The imputed value then is the average of these k *guest ratings* (*star ratings*). For the RF and GBTs, four stage *Spark* pipeline is set up to ensure replicable experiments: *vectorization*, *standardization*, *cross validation*, and *regression*. The first two steps have already been described in Section 5.1. The *cross validation* step splits the complete dataset (without missing ratings) into a set of folds which are used as a training and validation subsets. E.g., with 10 folds, the cross validator generates 10 (training, validation) dataset pairs, each of which uses 9/10 of the data for training and 1/10 for validation iterating through them during the training. The *regression* step involves the two algorithms described in Sections 3.3 (RF) and 3.4 (GBTs) from the *ML Spark* library. Table 3 contains results from applying the above imputation techniques on *guest/star ratings*. As expected, the two baselines (Mean and Median) achieved the lowest accuracy with MAE of 1.21 and 1.01 for the guest rating, and 0.70 and 0.75 for star rating. From the implemented state-of-the-art algorithms, K-NN produced the worst accuracy (0.70 and 0.50 for star and guest rating respectively). This approach also appeared to be very slow, because of the large number of samples in the training set (~200K samples). The GBTs technique with standardized features achieved the best result (0.36 for star rating and 0.34 for guest rating), followed by the RF with the same setup (0.39 and 0.47 for star/guest rating respectively). The same experiments were repeated removing the amenities from the dataset in order to assess the importance of this feature. While the use of amenities as a feature failed the prediction of the *Co-Clicks* similarity (*Table 2*), their use in the ratings imputation improved the overall accuracy (Table 3). These contradictory results can be explained by the fact that the amenities partially define the quality of the property (e.g., property with amenities such as *pool* and *spa* has higher likelihood to be rated as a 5 star). On the other hand, the geographical position seems to be useful for determining the guest rating (e.g., properties close to stations, airports or famous landmarks have greater probability of receiving a higher rating).

Table 3 Guest Rating and Star Rating imputation errors. The MAE is used as error function to compare the imputation results of two baselines (Mean and Median imputation) and three state-of-the-art approaches (K-NN, RF, and GBTs). The experiments are performed with (v) or without (-) amenities (Am.) and standardisation (St.).

			Guest Rating		Star Rating	
Method	Am.	St.	MAE	SD	MAE	SD
Mean	v	-	1.21	1.17	0.70	0.65
Median	v	-	1.01	0.99	0.75	0.71
K-NN	v	-	0.70	0.66	0.50	0.45
RF	v	-	0.40	0.37	0.42	0.40
	-	-	0.44	0.40	0.50	0.46
	v	v	0.39	0.35	0.47	0.44
GBTs	v	-	0.37	0.34	0.39	0.36
	-	-	0.40	0.37	0.44	0.42
	v	v	**0.34**	**0.31**	**0.36**	**0.33**

6. Conclusion

The missing data problem for travel Recommender Systems is investigated giving particular focus on the VR market. After an initial exploratory analysis, the dataset of properties at hand showed a 50% and 19% missingness for guest rating and star rating respectively. Furthermore, the properties similarity, namely the Co-Clicks, is always missing when a new property is added in the catalog (because there is no historical data of users co-clicks with other properties). To deal with that, non-historical features (amenities and geographical position) are introduced, analysed and employed to determine the best proxy of the *Co-Clicks* (when not available). Results from the applied three similarity measures (Jaccard, Weighted Hamming and Fuzzy-C-Means rankings) showed the Fuzzy-C-means to be the best approximation metric with 35% of correctly ranked items in first position, and 60% for the top 5 rankings, indicating a correlation between the co-clicked properties and their position on the map. The applied data imputation techniques for mitigating the guest and star ratings missingness (two baselines: Mean and Median; and three state-of-the-art models: K-NN, RF, and GBTs), resulted in recommending the GBTs as the most suitable model for this task, achieving the lowest MAE (0.36 and 0.34 for guest and star rating respectively), followed by the RF and KNN. As expected the baselines scored the lowest accuracy, with an error greater than 1 for guest rating and 0.75 for the star rating. Furthermore, the importance of amenities as a feature for this task was assessed repeating the same experiment using only the geographical features. While the use of amenities failed the prediction of the *Co-Clicks* similarity (with an accuracy between 9% and 25%), their use in the ratings imputation always improved the overall accuracy. Future work would analyse the behaviour of the RS when the engineered and imputed features are used during the recommendation process for VR properties. In addition to that, new error functions will be tested in order to assess and distinguish the current under/over estimation of the imputed ratings by the model based techniques.

References

[1] G. Zervas, D. Proserpio, and J. Byers, Therise of the sharing economy: Estimating the impact of airbnb on the hotel industry, *Boston U. School of Management Research Paper*, 2016, 2013-2016.

[2] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, Missing value estimation methods for dna microarrays, *Bioinformatics*, *17*(6), 2001, 520–525.

[3] L. Breiman, Random forests, *Machine learning*, *45*(1), 2001, 5–32.

[4] J. Ye, J.-H. Chow, J. Chen, and Z. Zheng, Stochastic gradient boosted distributed decision trees, *in Proceedings of the 18th ACM conference on Information and knowledge management, ACM*, 2009, 2061–2064.

[5] C. Robusto, The cosine-haversine formula, *The American Mathematical Monthly*, *64*(1), 1957, 38–40.

[6] F. Sarro, A. Petrozziello, and M. Harman, Multi-objective software effort estimation, *in Proceedings of the 38th International Conference on Software Engineering, ACM*, 2016, 619–630.

[7] M. Fernàndez-Delgado, E. Cernadas, S. Barro, and D. Amorim, Do we need hundreds of classifiers to solve real world classification problems, *J. Mach. Learn. Res*, *15*(1), 2014, 3133–3181.

[8] X. Meng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S.

Owen, et al., Mllib: Machine learning in apache spark, *JMLR, 17*(34), 2016, 1–7.

[9] X.-Y. Pan, Y. Tian, Y. Huang, and H.-B. Shen, Towards better accuracy for missing value estimation of epistatic miniarray profiling data by a novel ensemble approach, *Genomics, 97*(5), 2011, 257–264.

[10] C. J. Willmott and K. Matsuura, Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance, *Climate research, 30*(1), 2005, 79–82.

[11] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li, Ranking measures and loss functions in learning to rank, *in Advances in Neural Information Processing Systems*, 2009, 315–323.

[12] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, Hive: a warehousing solution over a map-reduce framework, *Proceedings of the VLDB Endowment, 2*(2), 2009, 1626–1629.

[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, Spark: cluster computing with working sets., *HotCloud, 10,* 2010, 1–7.

[14] M. Odersky, L. Spoon, and B. Venners. *Programming in scala* (Artima Inc, 2008).

[15] N. R. Pal, K. Pal, J. M. Keller, and J. C. Bezdek, A possibilistic fuzzy c-means clustering algorithm, IEEE transactions on fuzzy systems, 13(4), 2005, 517– 530.