Azure Solution for Hotel Recommender Engine

NCE Skills Stream two – Analytics & AI

Metodi Simeonov



Data Analysis

Table of contents

Data Analysis 1

Correlation Testing 2

Feature Engineering 3

Azure ML architecture 4

Web Service deployment and consumption 5

The IDE platform of choice used for the data analysis was R Studio. By applying various statistical methods it was determined that the total number of hotels was 1478 distributed across 6 European cities: Vienna, Amsterdam, Barcelona, London, Milan and Paris. The single most contributing city in the dataset was London with 262301 reviews. The mean and median of the hotel score are nearly identical at 8.4 indicating symmetrical distribution.

|  |
| --- |
| 1. City                  Country       Additional\_Number\_of\_Scoring 2. Amsterdam: 57214   Austria       : 38939   Min.   :   1.0 3. Barcelona: 60149   France        : 59928   1st Qu.: 169.0 4. London   :262301   Italy         : 37207   Median : 341.0 5. Milan    : 37207   Netherlands   : 57214   Mean   : 498.1 6. Paris    : 59928   Spain         : 60149   3rd Qu.: 660.0 7. Vienna   : 38939   United Kingdom:262301   Max.   :2682.0 9. Review\_Date     Average\_Score 10. 8/2/2017 :  2585   Min.   :5.200 11. 9/15/2016:  2308   1st Qu.:8.100 12. 4/5/2017 :  2284   Median :8.400 13. 8/30/2016:  1963   Mean   :8.397 14. 2/16/2016:  1940   3rd Qu.:8.800 15. 7/5/2016 :  1904   Max.   :9.800 16. (Other)  :502754 17. Hotel\_Name 18. Britannia International Hotel Canary Wharf       :  4789 19. Strand Palace Hotel                              :  4256 20. Park Plaza Westminster Bridge London             :  4169 21. Copthorne Tara Hotel London Kensington           :  3578 22. DoubleTree by Hilton Hotel London Tower of London:  3212 23. Grand Royale London Hyde Park                    :  2958 24. (Other)                                          :492776 25. Reviewer\_Nationality    Negative\_Review 26. United Kingdom           :245246    No Negative:127890 27. United States of America : 35437     Nothing   : 14295 28. Australia                : 21686     Nothing   :  4236 29. Ireland                  : 14827     nothing   :  2225 30. United Arab Emirates     : 10235     N A       :  1037 31. Saudi Arabia             :  8951     None      :   984 32. (Other)                   :179356    (Other)    :365071 33. Review\_Total\_Negative\_Word\_Counts Total\_Number\_of\_Reviews 34. Min.   :  0.00                    Min.   :   43 35. 1st Qu.:  2.00                    1st Qu.: 1161 36. Median :  9.00                    Median : 2134 37. Mean   : 18.54                    Mean   : 2744 38. 3rd Qu.: 23.00                    3rd Qu.: 3613 39. Max.   :408.00                    Max.   :16670 41. Positive\_Review   Review\_Total\_Positive\_Word\_Counts 42. No Positive  : 35946   Min.   :  0.00 43. Location    :  9222   1st Qu.:  5.00 44. Everything  :  2284   Median : 11.00 45. location    :  1677   Mean   : 17.78 46. Nothing     :  1243   3rd Qu.: 22.00 47. The location:  1126   Max.   :395.00 48. (Other)      :464240 49. Total\_Number\_of\_Reviews\_Reviewer\_Has\_Given Reviewer\_Score 50. Min.   :  1.000                            Min.   : 2.500 51. 1st Qu.:  1.000                            1st Qu.: 7.500 52. Median :  3.000                            Median : 8.800 53. Mean   :  7.166                            Mean   : 8.395 55. ays\_since\_review 56. 1 days :  2585 57. 322 day:  2308 58. 120 day:  2284 59. 338 day:  1963 60. 534 day:  1940 61. 394 day:  1904 62. (Other):502754  65. 3rd Qu.:  8.000                            3rd Qu.: 9.600 66. Max.   :355.000                            Max.   :10.000 |

**Figure 1.** Summary statistic of the dataset

Correlation testing

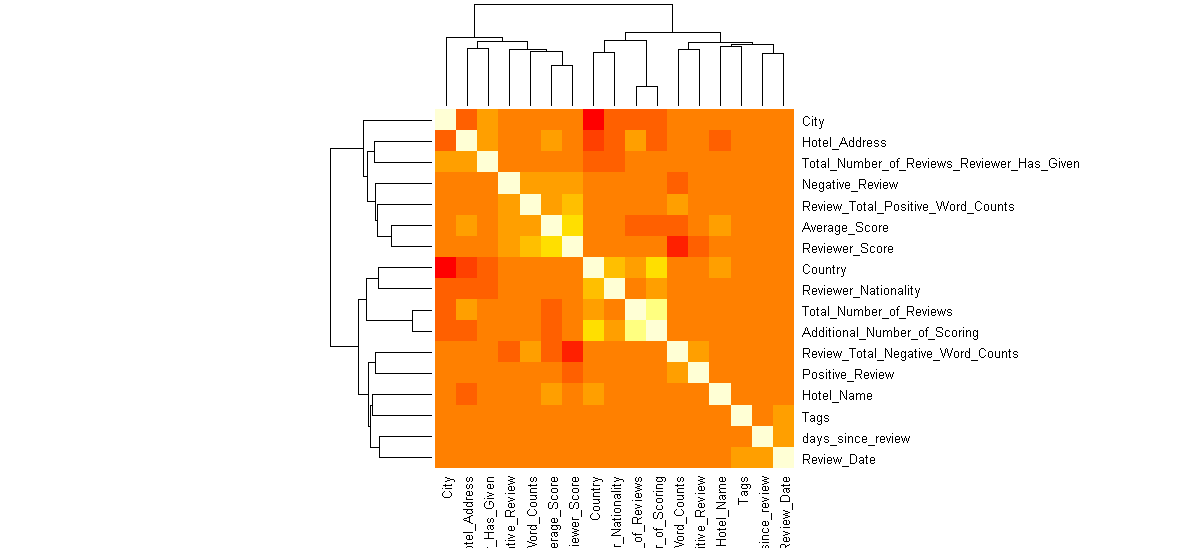
Performing Pearson’s correlation testing provided several insights into the data structure and the link between the different features of the dataset. Applying the “cor.test” function to few of the columns shed some interesting findings.

1. > cor.test(dat$Reviewer\_Score, as.numeric(dat$City))
3. Pearson's product-moment correlation
5. data:  **dat$Reviewer\_Score and as.numeric(dat$City)**
6. t = 1.3828, df = 515740, p-value = 0.1667
7. alternative hypothesis: true correlation **is** **not** equal to 0
8. 95 percent confidence interval:
9. -0.000803698  0.004654661
10. sample estimates:
11. cor
12. 0.001925496
14. > cor.test(dat$Reviewer\_Score, as.numeric(dat$Country))
16. Pearson's product-moment correlation
18. data:  **dat$Reviewer\_Score and as.numeric(dat$Country)**
19. t = -22.755, df = 515740, p-value < 2.2e-16
20. alternative hypothesis: true correlation **is** **not** equal to 0
21. 95 percent confidence interval:
22. -0.03439581 -0.02894291
23. sample estimates:
24. cor
25. -0.0316696
26. > cor.test(dat$Reviewer\_Score, as.numeric(dat$Reviewer\_Nationality))
28. Pearson's product-moment correlation
30. data:  **dat$Reviewer\_Score and as.numeric(dat$Reviewer\_Nationality)**
31. t = 25.103, df = 515740, p-value < 2.2e-16
32. alternative hypothesis: true correlation **is** **not** equal to 0
33. 95 percent confidence interval:
34. 0.03220762 0.03765933
35. sample estimates:
36. cor
37. 0.03493373

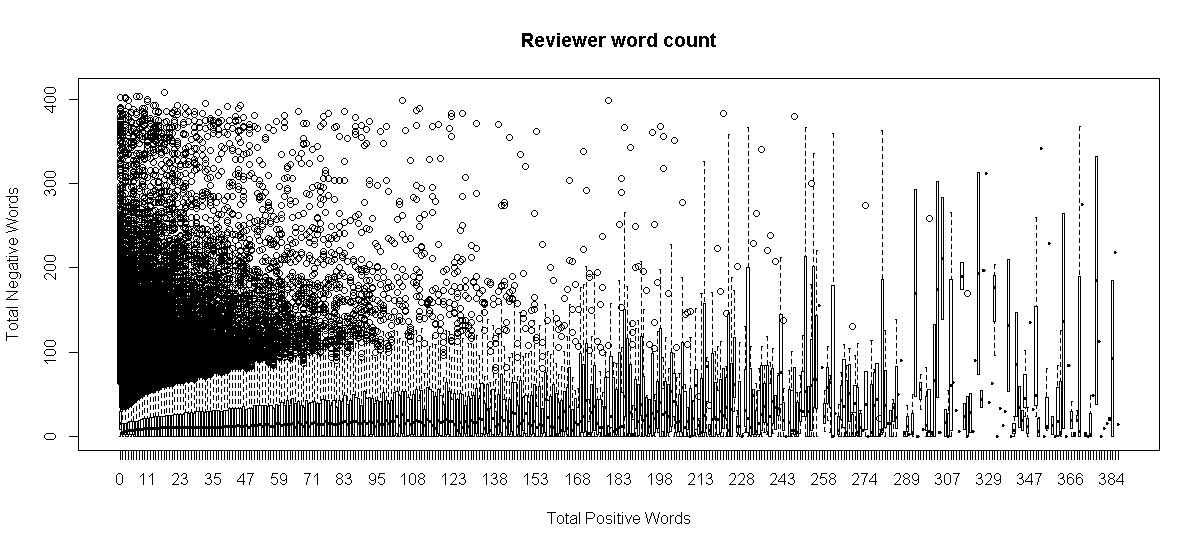
**Figure 2.** Pearson’s correlation between different columns from the dataset

Hypothesis outcome from visualizing the data:

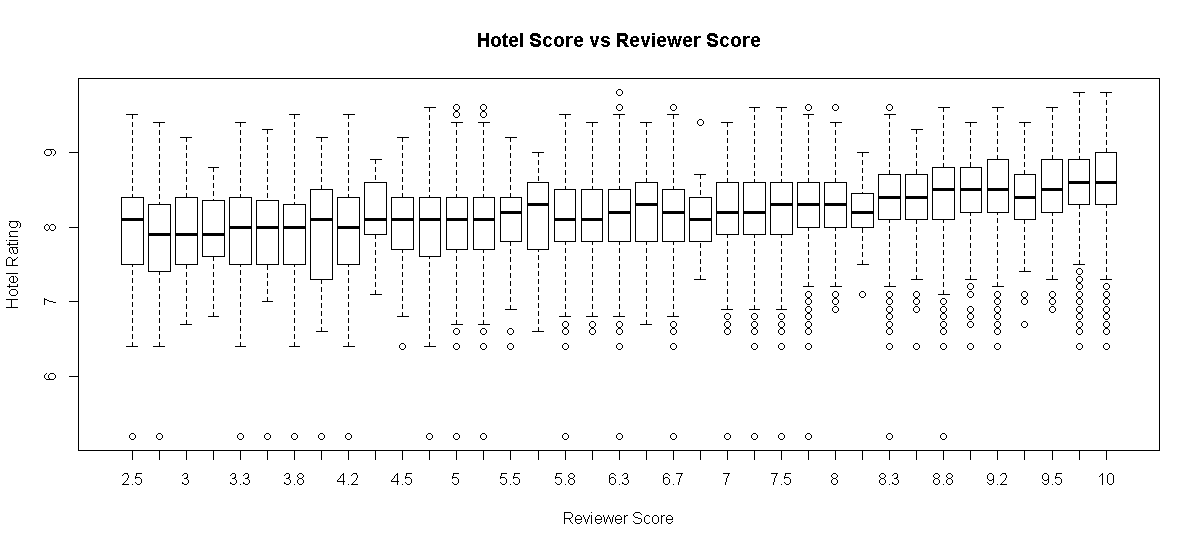
* Negative Review correlates with Total negative Word Counts. People who are dissatisfied tend to express it with more words.
  + Negative review words more on average than the positive ones
* Average Score correlated highly with Total Number of Reviews
* Total number of reviews given correlates with Country and Nationality



**Figure 3.** Heath map distribution of the correlations in the dataset



**Figure 4.** Boxplot distribution of the reviewer word count – positive vs negative



**Figure 5.** Boxplot distribution of the hotel score vs reviewer score

**Feature Engineering**

The descriptive statics provided valuable insights into the data, however creating the predictive engine in Azure Machine Learning required specific feature engineering to enrich the dataset and enhance the model. The main module of AML model was Score Matchbox Recommender. The goal of creating a recommendation system is to recommend one or more "items" to "users" of the system. In this scenario, our item is a hotel name and a user is a person with specific item preferences.

Out of the three types of recommender engines, we are using “Item Recommendation” and “Similar Items”. To recommend items for users, we provide a list of users and items as input. From this data, the model uses its knowledge about existing items and users to generate a list of items with probable appeal to each user. We can customize the number of recommendations returned, and set a threshold for the number of previous recommendations that are required in order to generate a recommendation.

The scored dataset returned by Score Matchbox Recommender lists the recommended items for each user. The first column contains the user identifiers.

Additional columns are generated, depending on the value we set for Maximum number of items to recommend to a user. Each column contains a recommended item (by identifier). The recommendations are ordered by user-item affinity, with the item with highest affinity put in column, Item 1.

Before we ingest the data into AML we need to perform few consecutive steps:

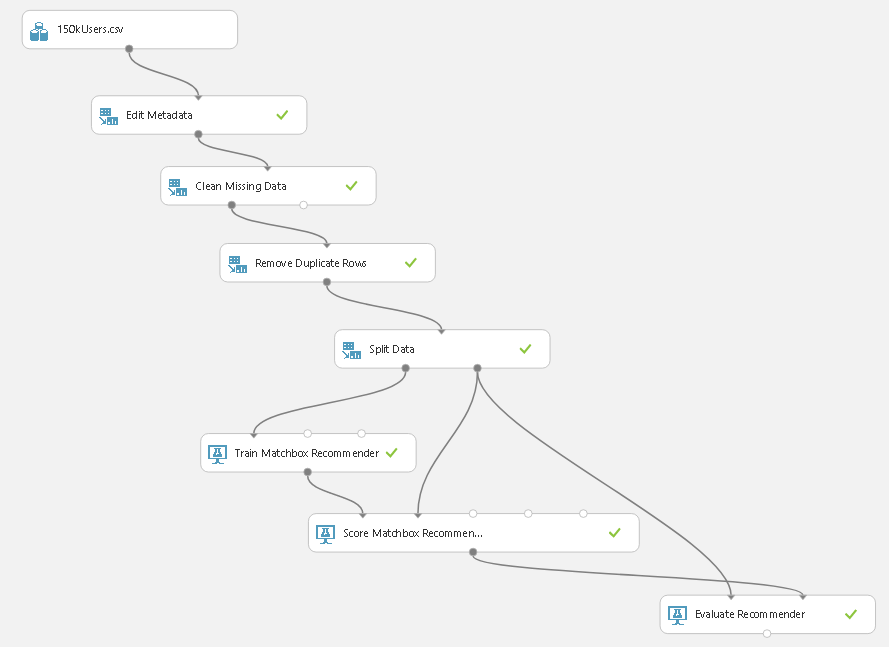
* First, we need to make an assumption related to the number of unique users. This is critical step as without unique user ID our recommendation engine will not work. During the initial statistical analysis, we found out that 154640 users have given one review only and the remaining 361098 users have on an average review count of 10. These numbers by themselves are only indicative and we cannot conclude the real number of unique users. Therefore, we create a new column into the data frame, which we call “UserID” and use R code to randomly ingest numbers from 1 to 192030 in all of the 515739 rows in the data frame. By doing this we ensure that, each user has rated at least 2 hotels. Without this our recommendation engine will not work. The R code used for this operation is below:

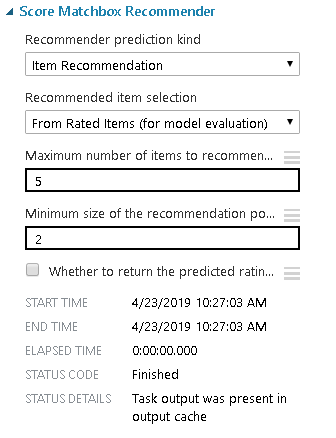
1. table(dat$Total\_Number\_of\_Reviews\_Reviewer\_Has\_Given)
2. users <- seq(**from**=1, to=192031, length.out = 192030)
3. users <- as.integer(users)
4. dat$UserID <-sample(192030, size = nrow(dat), replace = TRUE)

* After we successfully ingested and distributed 192030 unique ID’s across the data, we need to drop the columns that are not required for the recommendation engine. We are using R code again for this task. Sample below:
* dat$Total\_Number\_of\_Reviews\_Reviewer\_Has\_Given <- NULL
* dat$long <- NULL
* dat$lat <- NULL
* dat$Tags <- NULL
* dat$Positive\_Review <- NULL
* dat$Negative\_Review <- NULL
* etc…
  + The final form of the data set consist of only three columns – UserID, Hotel Name and Reviewer Score.
* Once we are satisfied with the content of our new data frame we then export it in a new CSV file using this R code:
* write.csv(dat,"C:/Data/Hotel Recommender//Hotel\_Revies\_for\_AML.csv", row.names = FALSE)

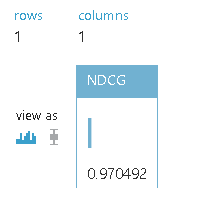
**Azure Machine Learning Model**

For this specific model, we have created two separate machine learning training experiments. The first one aimed at predicting Item Recommendations is configured the following way:

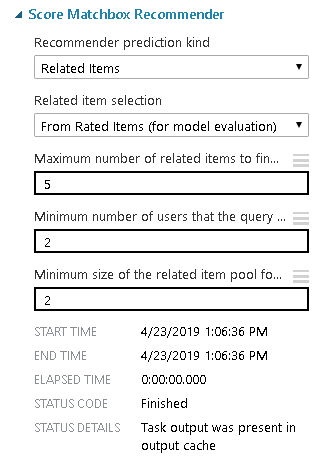




The NDCG (normalized discounted cumulative gain) score of the model turned out to be high, which is a solid indication of the accuracy of the model. Because it is impossible to know the actual "ground truth" for the recommended items, Evaluate Recommender uses the user-item ratings in the test dataset as gains in the computation of the NDCG. The value returned by the Evaluate Recommender module of 0.9704 is very close to the maximum achievable one.



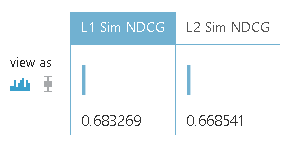
The second machine learning training experiment is aimed at “Related Items”. The module workflow uses the same logic as the first model with slightly different configuration.



This time around, the Evaluate Recommender computes the average normalized discounted cumulative gain (NDCG), based on Manhattan (L1 Sim NDCG) and Euclidean (L2 Sim NDCG) distances, and returns both values in the output dataset. Because there is no actual ground truth for the related users, Evaluate Recommender uses the following procedure to compute the average NDCGs.

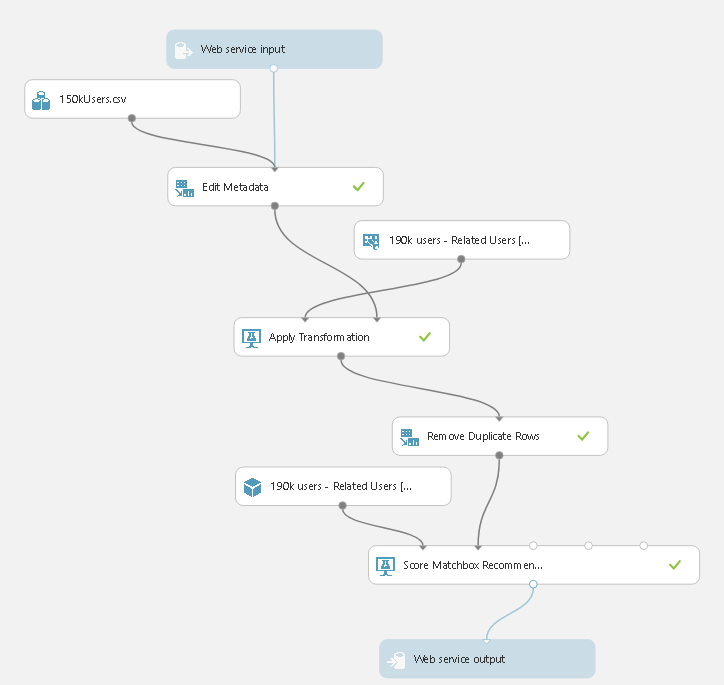
For each user of interest in the scored dataset:

1. Find all items in the test dataset, which have been rated by both the user of interest and the related user under consideration.
2. Create two vectors from the ratings of these items: one for the user of interest, and one for the related user under consideration.
3. Compute the gain as the similarity of the resulting two rating vectors, in terms of their Manhattan (L1) or Euclidean (L2) distance.
4. Compute the L1 Sim NDCG and the L2 Sim NDCG, using the gains of all related users.
5. Average the NDCG values over all users in the scored dataset.

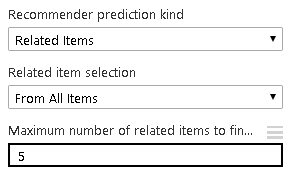
In other words, gain is computed as the similarity (normalized Manhattan or Euclidian distances) between a user of interest (the entry in the first column of scored dataset) and a given related user (the entry in the n-th column of the scored dataset). The gain of this user pair is computed using all items for which both items have been rated in the original data (test set). The NDCG is then computed by aggregating the individual gains for a single user of interest and all related users, using logarithmic discounting. That is, one NDCG value is computed for each user of interest (each row in the scored dataset). The number that is finally reported is the arithmetic average over all users of interest in the scored dataset (i.e. its rows). 

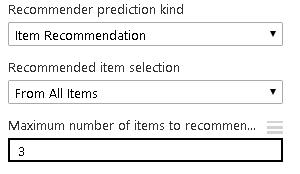
For Related Items model, the NDCG score is sufficiently high indicating good accuracy of the predictions.

Once we are satisfied with the accuracy of the models we proceed to creating a Predictive Experiment.



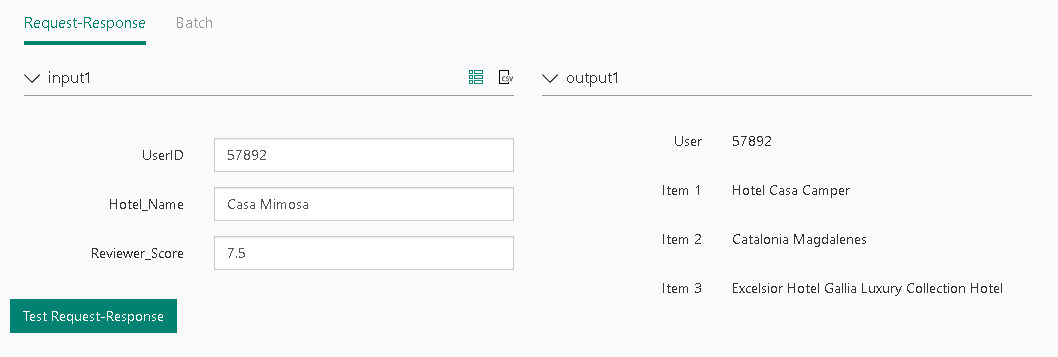
Here we change the configuration settings of both Score Matchbox Recommenders to use the entire dataset.- From All Items.

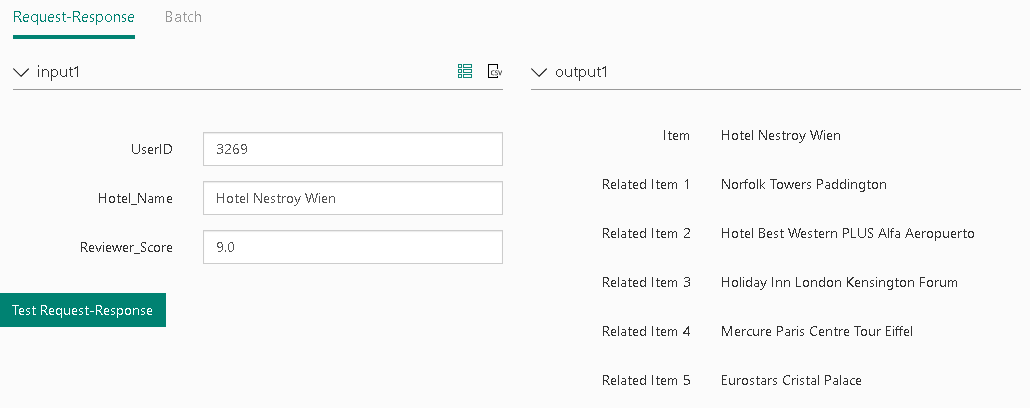




**Web Service deployment and consumption**

We deploy both predictive experiments using the same logic and parameters – request response service (as we expect single item request) with API key integration.





The web service consumption can happen through Excel Online using the easily available Azure Machine Learning Add-on or using predefined code for C#, Python, Java and R integration into a specific GUI.