# Product Matching system to find matching products of Zalando and Aboutyou

Danyal Quazi
*School of Computing*
*Dublin City University*
Dublin, Ireland
danyal.quazi2@maildcu.ie

***Abstract*—A prototype of a Product Matching with all the necessary steps which are required like text pre-processing, model fitting, and evaluation. The project is an attempt to match products offered by the company Zalando to the same products offered by Aboutyou using attributes like the product title, the color, and the product description. A dataset of 102884 products is used. In order to compare these attributes, various similarity scores were used as the features to see the effect on the output variable which in this case is match or no-match. Various ML algorithms are tested using various accuracy matrices to find the most accurate model. The aim is to maximize the F1 score. XGB obtained the highest F1 score which was then used to find the matching products from a dataset containing entirely different products.**

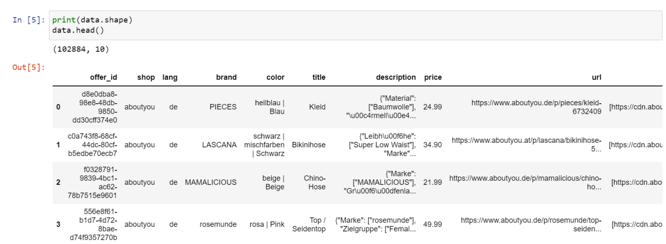
***Index Terms*—Product Matching, Machine Learning Models, Preprocessing, XGB (Extreme Gradient Boosting ), F1 Score, Zalando, Aboutyou, Similarity Score, Confusion Matrix**

## I. Introduction

With the increase in the popularity of online shopping, there are now billions of products sold online. Most of these products are sold by numerous stores or websites. Every customer wants to buy a particular product at the lowest price possible with the best deal. Similarly, for the sellers, it is very crucial to understand where that seller stands in the market by comparing their prices for a product to the prices offered by the competitors. Due to this, a need for a technique by which the same or similar products can be matched arises. The task is simply finding the matching products using the data provided.

The most accurate technique is if a human does this job, but looking at the volume and velocity of data that needs to be processed, it's just not practical. There is a need for a system that can efficiently classify the matching products when provided with data. To produce this assignment, Dublin City University collaborated with Zalando, a major online apparel company. The company aims to offer competitive rates in each of its dynamic market situations to save clients time comparing costs and to increase revenue. Zalando needs to identify exact product matches across all relevant European competitors in order to do so for its hundreds of thousands of individual products.

The dataset is in the form of a parquet file. There are two datasets provided one containing all the offers by two stores Zalando and Aboutyou. This dataset contains 102884



```
In [5]: print(data.shape)
        data.head()
(102884, 10)
Out[5]:
```

| | offer_id | shop | lang | brand | color | title | description | price | url | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | d8e0dba8-98e8-48db-9850-dd30cff374e0 | aboutyou | de | PIECES | hellblau \| Blau | Kleid | {"Material": ["Baumwolle"], "\u00c4rmel\u00e4... | 24.99 | https://www.aboutyou.de/p/pieces/kleid-6732409 | [https://cdn.abou |
| 1 | c0a743f8-68cf-44dc-80cf-b5edbe70ecb7 | aboutyou | de | LASCANA | schwarz \| mischfarben \| Schwarz | Bikinihose | {"Leibh\u00f6he": ["Super Low Waist"], "Marke"... | 34.90 | https://www.aboutyou.at/p/lascana/bikinihose-5... | [https://cdn.abou |
| 2 | f0328791-9839-4bc1-ac62-78b7515e9601 | aboutyou | de | MAMALICIOUS | beige \| Beige | Chino-Hose | {"Marke": ["MAMALICIOUS"], "Gr\u00f6\u00dfenla... | 21.99 | https://www.aboutyou.de/p/mamalicious/chino-ho... | [https://cdn.abo |
| 3 | 556e8f61-b1d7-4d72-8bae-d74f9357270b | aboutyou | de | rosemunde | rosa \| Pink | Top / Seidentop | {"Marke": ["rosemunde"], "Zielgruppe": ["Femal... | 49.99 | https://www.aboutyou.de/p/rosemunde/top-seiden... | [https://cdn.abo |

Fig. 1. A snap of the data.

rows each with a unique offer and 10 columns containing the attributes of the product or the offer which can greatly help in describing the product. The columns are offer_id, shop, lang, brand, color, title, description, price, url, image_urls. The second file contains a subset of the first dataset which contains only the offer ids of Zalando and aboutyou which match with each other contained in separate columns we can use these results to check the accuracy of the proposed ML model for the classification. There is another dataset that contains offers from Zalando and aboutyou in the same form as that of the first dataset but the matches of these offers are unknown.

The challenge is to use text data intelligently for this assignment using features such as product titles, colors, and descriptions. The project is divided into multiple sections like Preprocessing, Data transforming, Similarity calculation, Fitting ML algorithms, Evaluation and results, Working on test data, Conclusion.

## II. Data Cleaning

Text pre-processing is one of the most crucial steps in making any Machine Learning system. It can greatly influence the results and accuracy of the Model. Preprocessing is the process of removing or converting all the irrelevant data into something manageable and useful. There are multiple steps involved in text processing.

### *A. none type clean*

Before any pre-processing task, we will have to remove the none type from the text. None type is simply empty values or NULL values. For python, the none type is a data type of its own and will treat it differently. This will result in errors because in our case we will be dealing with the string data type. In order to deal with this problem, all the none type characters need to be converted to string values without changing the meaning of the document. We can achieve this by simply replacing the NULL values with an empty string "".

### B. converting to lower

The dataset contains strings in both upper cases and lower case characters. This will not necessarily give errors but can cause some other problems like lower similarity scores and skipping a few strings etc. The functions for similarity score will treat upper case and lower case characters differently hence it is best practice to convert text to one form.

### C. removing the pipe

The column containing the colors for a product in the aboutyou has multiple colors separated by a pipe '|' and needs to be replaced by an empty string to avoid errors while getting the similarity scores. A simple .replace() function is used to replace the'|'.

```
In [63]: df.head()
```

Out[63]:

<table>
<tr><th></th><th>zalando</th><th>aboutyou</th><th>zal_col</th><th>zal_title</th><th>zal_des</th><th>a_you_col</th><th>a_you_title</th><th>a_you_des</th><th>match</th></tr>
<tr><td>0</td><td>90990814-3cf7-4263-bd15-cbc9c71880be</td><td>17507e80-3ff4-4804-a32c-bb216adfde75</td><td>beige</td><td>vmellen top t-shirt basic</td><td>main_supplier_code K70240 $ name_suffix birch ...</td><td>weiß | hellbraun | Weiß</td><td>handtasche 'cessily'</td><td>{"Gr\u00f6\u00dfe (Volumen)": ["Klein (< 25 l)...</td><td>0</td></tr>
<tr><td>1</td><td>ee69a476-787d-4a4f-86ae-8ee45439170e</td><td>bae879ea-4397-4c2a-b3e2-6a6239a0601a</td><td>schwarz</td><td>vmava h neck langarmshirt</td><td>name_suffix black patternunifarben materi...</td><td>blau | Blau</td><td>baggy pants</td><td>{"Marke": ["VERO MODA"], "Gr\u00f6\u00dfenlauf...</td><td>0</td></tr>
<tr><td>2</td><td>1f9cae18-3aed-41ab-b146-884e655eec12</td><td>df4de1c1-a28f-4eda-9706-7cec70efbc9a</td><td>camel</td><td>pcboss 3/4 blazer</td><td>skirt_details Schulterpolster $ name_suffix ap...</td><td>pastellrot | Rot</td><td>blazer 'boss'</td><td>{"Zielgruppe": ["Female"], "\u00c4rmell\u00e4n...</td><td>1</td></tr>
<tr><td>3</td><td>995dedc0-9c54-4f50-b8e4-b17034a42571</td><td>36c7c18a-7526-446d-b065-c978380a26c2</td><td>braun</td><td>adellyn minirock</td><td>skirt_details Unterrock name_suffixbrown ...</td><td>chamois | umbra | Braun</td><td>rock</td><td>{"Zielgruppe": ["Female"], "L\u00e4nge": ["Kur...</td><td>1</td></tr>
<tr><td></td><td>50568527-</td><td>a6666f68-</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Fig. 2. A snap showing the color seperated with pipe.

## III. Data Transforming

This section is a part of data preparation and involves all the steps of converting the dataset into a form that can be used easily for further processes like similarity computation and fitting ML models.

The very first transformation done is splitting the offers into two sets ie Zalando and Aboutyou. By splitting the dataset based on the shop it gets easier to compare these two shops. Once the dataset is split, the offer_id column is set as the index which can help in accessing the data using the offer_id

```
In [26]: zalando.loc['b33f55d6-0149-4063-8b63-3eeae63562a2']

Out[26]: shop                                                      zalando
         lang                                                           de
         brand                                                  Swarovski
         color                                                silberfarben
         title                                         CREATIVITY  Halskette
         description    main_supplier_code K85009 $ name_suffix silver...
         price                                                   59.867273
         url            https://www.zalando.de/lookup/article/4SW51L01...
         image_urls     [https://img01.ztat.net/article/46ee8503931f49...
         Name: b33f55d6-0149-4063-8b63-3eeae63562a2, dtype: object
```

Fig. 3. Accessing an offer using an offer-id.

The dataset containing the matches is used to prepare the dataset for training and testing. The dataset already contains the matched ids. The three important attributes used in this project are then added to the matched dataset. A column is added named 'Match' which will contain either 0 or 1 representing match and no-match respectively. As the matching dataset contains all the offers which are matched, value 1 is added to the match column. The dataset containing the matched values is now complete but the offer_ids for non-matching offers still need to be added to the dataset. This is achieved by first dropping the matched values from both datasets Zalando and Aboutyou, and dropping the un-needed columns so that it contains only important features. This data is then shuffled and placed in a new dataset to form two columns like the matching set. A column 'Match' is added and the value 0 is initiated for all the cells.

This new dataset containing all unmatched offers is then joined to the matched dataset. The dataset now contains both, matched and unmatched offers with the label as 0 or 1. Finally, the dataset is again shuffled. The dataset now contains 9 columns as shown in the following picture.

```
In [56]: final_df.tail()
```

Out[56]:

<table>
<tr><th></th><th>zalando</th><th>aboutyou</th><th>zal_col</th><th>zal_title</th><th>zal_des</th><th>a_you_col</th><th>a_you_title</th><th>a_you_des</th><th>match</th></tr>
<tr><td>40899</td><td>2f4d18cf-7c4a-4987-8530-4cb772887c4f</td><td>5a25ac51-b472-481d-9da6-14db78fac2ec</td><td>bordeaux</td><td>VMGLORY ROLLNECK BLOUSE Strickpullover</td><td>name_suffix port royale patternunifarben ...</td><td>braun | Braun</td><td>Bluse / Tunika</td><td>{"Marke": ["BURBERRY"], "Zielgruppe": ["Female...</td><td>0</td></tr>
<tr><td>40900</td><td>6a5c7629-fdce-40d3-8a05-daf7b2130532</td><td>93adc96c-7136-4fe7-ad9b-761f56f83f13</td><td>pink</td><td>Jeansjacke</td><td>main_supplier_code K87700 $ skirt_details Eing...</td><td>beige | Beige</td><td>Tasche 'ALBURY'</td><td>{"Zielgruppe": ["Female"], "Gr\u00f6\u00dfe (V...</td><td>0</td></tr>
<tr><td>40901</td><td>86c461cf-e038-46ed-b805-8bbe207f30e0</td><td>8fa916d0-30d0-4d3e-bd7f-701bb8792581</td><td>schwarz</td><td>VMVILLA O NECK SLIT Strickpullover</td><td>name_suffix black patternunifarben materi...</td><td>schwarz | Schwarz</td><td>Kleid 'VILLA'</td><td>{"Zielgruppe": ["Female"], "\u00c4rmell\u00e4n...</td><td>1</td></tr>
<tr><td>40902</td><td>304f98ad-49d2-463b-8d3d-501ed80425b4</td><td>fc3e1a45-e2da-4b85-9616-0b03beb68ffc</td><td>blau</td><td>RAGO T-Shirt print</td><td>main_supplier_code K71252 $ skirt_details elas...</td><td>lila | Lila</td><td>Top / Seidentop</td><td>{"Marke": ["rosemunde"], "Zielgruppe": ["Femal...</td><td>0</td></tr>
<tr><td>40903</td><td>0b3a53bc-0043-41b1-a396-fa816609c824</td><td>6d90e44a-5f3b-4b30-9d12-1ce0d47eda0d</td><td>schwarz</td><td>SUCCESS ARMATURÉ Body</td><td>main_supplier_code K85249 $ skirt_details läng...</td><td>mint | Grün</td><td>Hose</td><td>{"Material": ["Viskose"], "L\u00e4nge": ["Lang...</td><td>0</td></tr>
</table>

Fig. 4. A figure showing the prepared data to calculate similarity

## IV. Calculating Similarity

The attributes like title, color, and description are all strings. An ML model can not work on text data as there is no way of directly understanding the effect of these string attributes on the output variable which in our case is 'match'. To find a relation between two values, we need to have a data type that can increase or decrease which can affect the output variable positively or negatively, for instance, integer or float.

One way we can achieve this is by calculating various similarity scores of the three attributes we have selected which shows how similar the two texts are. The similarity score of matching products will be high as the greater number of matching terms in the two texts. As the same or matched products will have higher similarity scores compared to unmatched products and will be labeled as 1 in the matched column, an ML algorithm can be trained on this data.

Some of the similarity scores used to get the features are listed below: Levenshtein Distance

- Damerau Levenshtein Distance
- Hamming Distance
- Jaro Similarity
- Jaro Winkler Similarity
- Match Rating Comparison

- Ratio
- Partial Ratio
- Token Sort Ratio
- Token Set Ratio
- Matching Numbers
- Matching Numbers Log
- Log Fuzz Score
- Log Fuzz Score Numbers

The figure below shows the calculated similarity scores. These scores can be used as features for the ML models to train. The data set is then divided into training and testing sets using the train_test_split() function.

```
In [80]: df.head()
Out[80]:
```

| match | levenshtein_distance_title | ... | jaro_winkler_similarity_des | match_rating_comparison_des | ratio_des | partial_ratio_des | token_sort_ratio_des | token_set_ratio_ |
|---|---|---|---|---|---|---|---|---|
| 0 | 22 | ... | 0.621800 | 0 | 31 | 33 | 37 | |
| 0 | 20 | ... | 0.666942 | 0 | 32 | 34 | 40 | |
| 1 | 15 | ... | 0.630542 | 0 | 30 | 34 | 39 | |
| 1 | 12 | ... | 0.612906 | 0 | 31 | 36 | 41 | |
| 0 | 28 | ... | 0.627584 | 0 | 30 | 33 | 40 | |

Fig. 5. A screenshot of similarity scores

The Correlations of similarities obtained are shown below in figure 6.

```
In [84]: training_data[training_data.columns[1:]].corr()['match'][:].sort_values(ascending=False)

Out[84]: match                                  1.000000
         w_ratio_title                          0.785425
         token_set_ratio_title                  0.725097
         token_sort_ratio_title                 0.654437
         log_fuzz_score_title                   0.619311
         partial_ratio_title                    0.586283
         token_set_ratio_des                    0.578200
         token_set_ratio_col                    0.566459
         w_ratio_col                            0.564298
         partial_ratio_col                      0.560497
         ratio_title                            0.548327
         uq_ratio_col                           0.547500
         ratio_col                              0.546744
         uq_ratio_title                         0.541008
         q_ratio_col                            0.540242
         q_ratio_title                          0.538310
         token_sort_ratio_col                   0.535543
         jaro_winkler_similarity_col            0.470361
         jaro_similarity_col                    0.452750
         match_rating_comparison_col            0.443024
         log_fuzz_score_numbers_col             0.409078
         log_fuzz_score_col                     0.409078
         log_fuzz_score_des                     0.404568
         matching_numbers_des                   0.384140
         matching_numbers_log_des               0.381429
         jaro_winkler_similarity_title          0.335656
         jaro_similarity_title                  0.333084
         log_fuzz_score_numbers_des             0.324080
         token_sort_ratio_des                   0.307839
         hamming_distance_des                   0.278448
         log_fuzz_score_numbers_title           0.269121
         damerau_levenshtein_distance_des       0.252317
         levenshtein_distance_des               0.252175
```

Fig. 6. A screenshot of Correlation

## V. Model Selection

The data is now divided into four parts, x_train: the training features, y_train: the training output which is the column 'match', x_test: the input which will be used to validate the trained model, and y_test: the 'match' column which will be used to test the model. For the training of the models, both input and output variables will be provided to the algorithm where it can train by understanding the effect of the input variables on the dependent variable. Once the model is trained, it will be made to predict the match variable on the data which is unknown. The predicted values can later be evaluated using many accuracy matrices discussed in the evaluation section. Multiple ML algorithms are trained on this data. Once trained, all the models can classify products as match or no-match represented by a 0 or 1. The models trained are listed below: [2]

- DummyClassifier
- K-Neighbors Classifier
- XGB Classifier (Extreme Gradient Boosting)
- Decision Tree Classifier
- Random Forest Classifier
- AdaBoost Classifier
- Gradient Boosting Classifier
- Perceptron
- MLP (Multi-layer Perceptron)
- XGBClassifer tuned

The detailed results and scores are discussed in the results and evaluation section. The best performing model in this project is the XGB Classifier which was selected to classify the matches from the testing data. XGBoost is a distributed gradient boosting library. It uses the Gradient Boosting framework to implement machine learning algorithms. XGBoost is a parallel tree boosting algorithm that solves a variety of data science issues quickly and accurately. [1]

## VI. Testing and Evaluation

In this section, the results obtained from the ML algorithms are discussed. The goal is to maximize the F1 score which is measured by taking the harmonic mean of the classifier's precision and recall. The model that is selected for the prediction of the test set which is XGB which has the highest F1 score. Other accuracy measures used are also shown in the figure. The scores of XGB are highlighted showing the highest accuracy with an f1 score of 0.955.

| | model | accuracy | mae | precision | recall | f1 | roc | run_time | tp | fp | tn | fn |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | DummyClassifier_stratified | 0.533817 | 0.466183 | 0.368716 | 0.361020 | 0.364827 | 0.498344 | 0.0 | 4908 | 2813 | 1643 | 2908 |
| 1 | KNeighborsClassifier | 0.930737 | 0.069263 | 0.921047 | 0.889475 | 0.904985 | 0.922266 | 0.08 | 7374 | 347 | 4048 | 503 |
| 2 | XGBClassifier | 0.966835 | 0.033165 | 0.956790 | 0.953637 | 0.955211 | 0.964126 | 0.21 | 7525 | 196 | 4340 | 211 |
| 3 | DecisionTreeClassifier | 0.932529 | 0.067471 | 0.910836 | 0.906834 | 0.908831 | 0.927254 | 0.01 | 7317 | 404 | 4127 | 424 |
| 4 | RandomForestClassifier | 0.955590 | 0.044410 | 0.941968 | 0.938036 | 0.939998 | 0.951986 | 0.1 | 7458 | 263 | 4269 | 282 |
| 5 | AdaBoostClassifier | 0.951923 | 0.048077 | 0.940797 | 0.928807 | 0.934763 | 0.947178 | 0.06 | 7455 | 266 | 4227 | 324 |
| 6 | GradientBoostingClassifier | 0.958198 | 0.041802 | 0.947077 | 0.939793 | 0.943421 | 0.954419 | 0.26 | 7482 | 239 | 4277 | 274 |
| 7 | Perceptron | 0.901239 | 0.098761 | 0.817094 | 0.945287 | 0.876528 | 0.910281 | 0.0 | 6758 | 963 | 4302 | 249 |
| 8 | MLP | 0.936441 | 0.063559 | 0.947756 | 0.876950 | 0.910979 | 0.924228 | 0.05 | 7501 | 220 | 3991 | 560 |
| 9 | XGBClassifer tuned | 0.955183 | 0.044817 | 0.921069 | 0.961547 | 0.940873 | 0.956489 | 0.03 | 7346 | 375 | 4376 | 175 |

Fig. 7. A screenshot of all the scores obtained

By looking at the confusion matrix, we can see how well the trained model performed on the validation set. It predicted 7346 as 0 or no-match and 4376 as 1 or matched product. It predicted only 500 comparisons incorrectly.

## VII. Working on the Testing data

The last step of the project is implementing the selected trained model on the testing data of which the results or matches are unknown. The data set contains a set of entirely

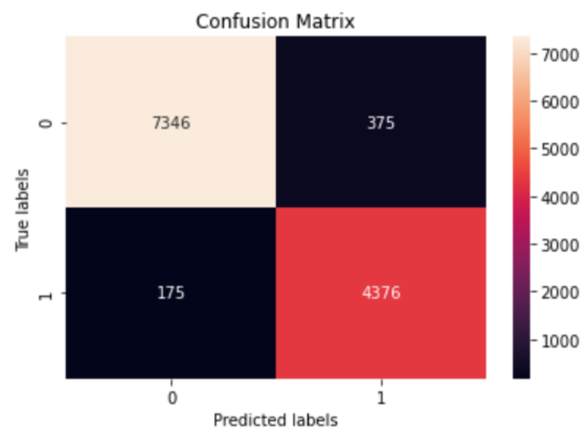



Fig. 8. A screenshot of the confusion matrix

different products which are not repeated. The data is processed using the same techniques as mentioned previously in the paper like dealing with none type, converting the text to lower case, and getting rid of the symbol like the '|' in the color column of the aboutyou data. Similar to the training data, the products are separated based on the shop.

The real challenge is to create a dataset on which we can implement the later steps like similarity calculation and predicting the output variable match. The first or the most obvious arrangement can be comparing each product from the Zalando dataset to every aboutyou product which results in a dataset too large to execute even the simplest calculation. The next option was to compare each Zalando product to every Aboutyou product which belongs to the same brand. This greatly reduced the size of the dataset. This dataset was used to create features using the similarity scores but again the execution of the task was taking too long it took around 8 hrs but was still incomplete and resulted in memory error, especially the description part. There was an option of dropping the description column and building the model based on only two features, the title, and the color but that will again result in a drop in the accuracy of the model.

To further reduce the size of the dataset, the products were compared which had the same brand and same color. This is done assuming that the same product but of different colors are considered two separate products. This resulted in a dataset containing about 12000 rows. This dataset was provided as the input to the trained model to predict the match column. Finally, the product comparisons with 1 in the match column were extracted, and later the column 'match' was dropped to leave only two columns: Zalando and Aboutyou containing only the matched products. This data set is then converted to a parquet file.

## VIII. Conclusions and Future Work

A Product Matching system was implemented by training the XG Boost algorithm using the similarity scores as the features. The model performed well when predicting the values for the testing set. The goal was to increase the F1 score. The model was able to reach an F1 score of 0.95. The matches for the testing set containing completely different products were predicted using the trained model.

A lot of work can be done to improve these results the model seems to be performing well but the formation of the testing data can be improved. Another feature that can greatly help in comparison is if a new feature is extracted for the product category which can then be used for comparing the products belonging to the same category. This can be done by using the description. An attempt was made to translate the description to English in order to extract more features but again the lack of resources the execution was very slow. Better matches can easily be obtained if products are grouped on the basis of the brand and executed on a better system with more resources and time.

## References


[1] XGBoost Documentation,https://xgboost.readthedocs.io/en/stable/

[2] How to create a product matching model using XGBoost, Matt Clarke, Saturday, March 13, 2021, Practical Data Science, https://practicaldatascience.co.uk/machine-learning/how-to-create-a-product-matching-model-using-xgboost