# Product Matching between Zalando and Aboutyou: CA684 Machine Learning Assignment 2022

Khizer Ahmed Biyabani
*School of Computing*
*Dublin City University*
Dublin, Ireland
khizer.biyabani2@mail.dcu.ie

***Abstract*—Consumers may now buy things from tens of thousands of online stores. The completeness of product descriptions and the taxonomies used to organize the items, on the other hand, varies amongst e-shops. Approaches for product integration on the Web are needed to improve the user experience, such as enabling for easy comparison of offers from multiple suppliers. The following report presents a method for implementing a product matching algorithm for matching products for the fashion company Zalando with that of the Aboutyou using different classification approach in machine learning.**

***Index Terms*—Product Matching, Text pre-processing, Text Similarity Distance Measures, Levenshtein distance, Damerau–Levenshtein distance, Hamming Distance, Token Set Ratio, Fuzzy string matching, Jaro and Jaro-Winkler, DummyClassifier, KNeighborsClassifier, XGBClassifier, DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, Perceptron, MLP, Confusion Matrix, F1 Score.**

## I. Introduction

The e-commerce sector, such as Amazon, Flipkart, and many other leading online purchasing sites, has recently experienced a huge upswing. With the rise in popularity of online selling platforms, each and every one of these online retailers require a technique for comparing their items to those of their competitors.

Zalando is a well-known European online fashion and leisure platform that brings together customers, businesses, and partners. Many of Zalando's rivals sell the same brand of items; for example, 'Aboutyou' is a fast-growing fashion platform and one of Zalando's competitors.

The following article presents a method for comparing various items between Zalando and Aboutyou, and it matches the common products between the two stores. Detailed information regarding the product data for the two stores for which the match must be discovered may be found throughout the text. To anticipate the match, a number of approaches have been developed, one of which is the text-similarity measure between the product color, title, and description of each of the two shops. We may determine the proximity or extent of each product's relationship using these similarities, as discussed in the next sections of this report.

We then use the similarity metrics to determine whether there is a match and train various machine learning classification models such as Dummy Classifier, KNearest Neighbours, and many more. The performance of each model is then evaluated in order to determine which model is the best fit. Finally, we make predictions about the matches in the provided test set, which is our final result.

### *A. About the Dataset*

The dataset was given by Zalando and it was a licensed dataset, the dataset contains the following files:

1) offer_train.parquet : This file is the training set. It cotains the product offers of each of the zalando and aboutyou products. This dataset is neede fo the training of the model. It contains 102884 product offers and 10 attributes or columns of each. Out of which 61980 are zalando offers and 40904 are anoutyou offers.
2) offer_test.parquet : The final test set contains the offers of which matches has to be predicted using our trained model. It is same as the training set, it has 106741 values in which 70105 and 36636 are zalando and aboutyou offers respectively.
   Each of the above datasets contain the following attributes:
   - offer_id : unique identifier for an offer of a product (i.e. a product x shop combination, where we don't know the product component).
   - shop: "zalando", "aboutyou".
   - lang: "de" (German).
   - brand: brand of each of the shop for example 'Nike' it is different for each of the shop.
   - color: Color of the product e.g. "blue" - note that there could be more than one and different 'shop's might have different 'brand' nomenclature ("ocean", "light blue", "...") and may have more than one color (ordering matters).
   - title: The actual title of the products e.g. "White Nike tennis top".
   - description: a long product description that can may contain material composition, size and different attributes for the. products, cleaning instructions, etc.
   - price: price in euro without any discount.
   - url: url of the product description page.

- image_urls: list of product images such as stock photo, with model, lifestyle photo, or close up.

3) match_training.parquet : This file contains the ground truth matches present in the training dataset.
The dataset has the following attributes:
- zalando: offer_id from “zalando” shop.
- aboutyou: offer_id from “aboutyou” shop.
- brand: unique identifier for the brand representing the match.

## II. Proposed Methodology

### A. System Architecture

The following diagram gives the summary of the entire process adopted :

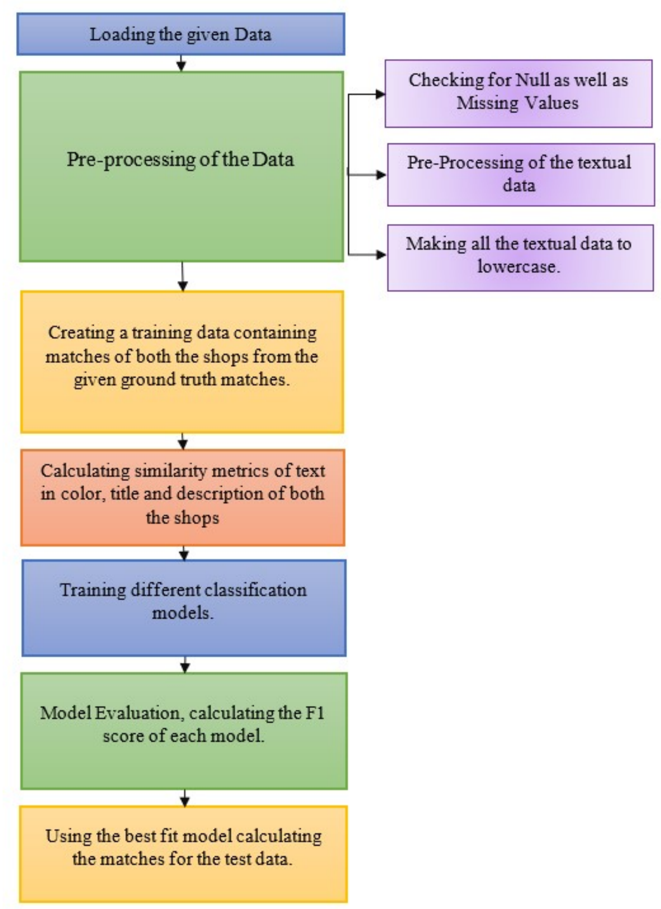
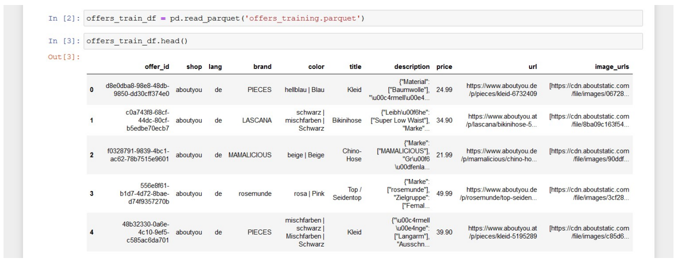

Loading the given Data
Pre-processing of the Data
Checking for Null as well as Missing Values
Pre-Processing of the textual data
Making all the textual data to lowercase.
Creating a training data containing matches of both the shops from the given ground truth matches.
Calculating similarity metrics of text in color, title and description of both the shops
Training different classification models.
Model Evaluation, calculating the F1 score of each model.
Using the best fit model calculating the matches for the test data.

Fig. 1. System Architecture: The process flow diagram

The further sections discuss in detail about each and every step in the above flow diagram.

### B. Loading the Dataset:

With the help of pandas library of python we can read the parquet file of our datasets. The following snippet in figure 2 shows the first five rows of the training set.



```
In [2]: offers_train_df = pd.read_parquet('offers_training.parquet')

In [3]: offers_train_df.head()
Out[3]:
```

| | offer_id | shop | lang | brand | color | title | description | price | url | image_urls |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | d8e0dba8-98e8-48db-9850-dd30cff374e0 | aboutyou | de | PIECES | hellblau \| Blau | Kleid | {"Material": ["Baumwolle"], "\u00c4rmell\u00e4... | 24.99 | https://www.aboutyou.de/p/pieces/kleid-6732409 | [https://cdn.aboutstatic.com/file/images/06728... |
| 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.aboutstatic.com/file/8ba09c163f54... |
| 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.aboutstatic.com/file/images/90ddf... |
| 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.aboutstatic.com/file/images/3cf28... |
| 4 | 48b32330-0a6e-4c10-9ef5-c585ac6da701 | aboutyou | de | PIECES | mischfarben \| schwarz \| Mischfarben \| Schwarz | Kleid | {"\u00c4rmell\u00e4nge": ["Langarm"], "Ausschn... | 39.90 | https://www.aboutyou.at/p/pieces/kleid-5195289 | [https://cdn.aboutstatic.com/file/images/c85d6... |

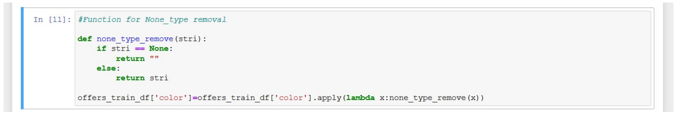
Fig. 2. First five rows of the training data.

### C. Pre-processing of Data

Data pre-processing is the process of converting raw data into a format that can be understood. Data in the real world is frequently partial, inconsistent, redundant, and loud. Data preparation entails a number of stages that aid in the conversion of unprocessed data into a usable state.

This mainly includes the data cleaning, the raw data that we have is usually corrupt and this data needs to processed before giving as an input to the machine learning algorithm. Both the training and test data contains certain anomalies, such as NoneTypes or there are certain values missing in various columns of the dataset.

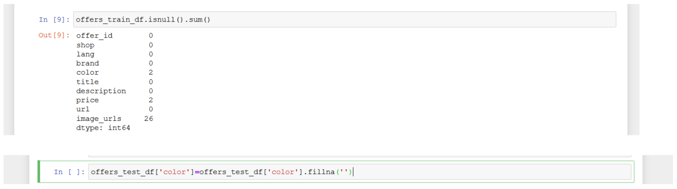
*1) Removing NoneTypes::* In Python, the data type of an object is NoneType when the object has no value. The NoneType object can be started with the keyword None. There is a function created to clean the NoneTypes from the daatset as shown below:



```
In [11]: #Function for None_type removal

         def none_type_remove(stri):
             if stri == None:
                 return ""
             else:
                 return stri

         offers_train_df['color']=offers_train_df['color'].apply(lambda x:none_type_remove(x))
```

Fig. 3. Python function for NoneType Removal

*2) Treating Null or Missing Values::* As you can see in the image below there certain missing or null values present in the dataset, we can find that the color column contains 2 missing values, these values can be replaced with an empty string (“”). The missing values in the image_urls, are also there but as we are working on the text so we can ignore these values.



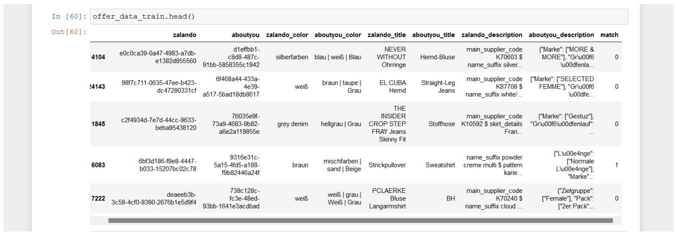
```
In [9]: offers_train_df.isnull().sum()

Out[9]: offer_id       0
        shop           0
        lang           0
        brand          0
        color          2
        title          0
        description    0
        price          2
        url            0
        image_urls    26
        dtype: int64

In [ ]: offers_test_df['color']=offers_test_df['color'].fillna('')
```

Fig. 4. Treating Missing Values

*3) Changing the text to lowercase::* For fast processing of text for color, title and description we standardise the case of the text to lower, this make the text uniform for the shop offers.

## *D. Preparing the Training data using Ground Truth Matches*

As we have the ground-truth matches for the training from that we can prepare a dataset that contains the matched offers of both zalando and aboutyou along-with the attributes of each of these product offers, i.e. the color, title and description. These matched offers we can combine with the unmatched offers. Also, there is column called 'match' with value 1 for a matched offer and 0 for an unmatched offer. This dataset looks like the following in the image below:



```
In [60]: offer_data_train.head()
Out[60]:
```

| | zalando | aboutyou | zalando_color | aboutyou_color | zalando_title | aboutyou_title | zalando_description | aboutyou_description | match |
|---|---|---|---|---|---|---|---|---|---|
| 4104 | e0c0ca39-0a47-4983-a7db-e1382d855560 | d1effbb1-c8d8-487c-91bb-5858355c1942 | silberfarben | blau \| weiß \| Blau | NEVER WITHOUT Ohrringe | Hemd-Bluse | main_supplier_code K70603 $ name_suffix silver... | {"Marke": ["MORE & MORE"], "Gr\u00f6\u00dfenla... | 0 |
| 24143 | 98f7c711-0635-47ee-b423-dc47280331cf | 6f468a44-433a-4e39-a517-5bad18db8617 | weiß | braun \| taupe \| Grau | EL CUBA Hemd | Straight-Leg Jeans | main_supplier_code K87708 $ name_suffix white/... | {"Marke": ["SELECTED FEMME"], "Gr\u00f6\u00dfe... | 0 |
| 1845 | c2f4934d-7e7d-44cc-9633-beba95438120 | 76035e9f-73a9-4683-9b82-a6e2a119855e | grey denim | hellgrau \| Grau | THE INSIDER CROP STEP FRAY Jeans Skinny Fit | Stoffhose | main_supplier_code K10592 $ skirt_details Fran... | {"Marke": ["Gestuz"], "Gr\u00f6\u00dfenlauf": ... | 0 |
| 6083 | 6bf3d186-f8e8-4447-b033-15207bc02c78 | 9316e31c-5a15-4fd5-a188-f9b82446a24f | braun | mischfarben \| sand \| Beige | Strickpullover | Sweatshirt | name_suffix powder creme multi $ pattern karie... | {"L\u00e4nge": ["Normale L\u00e4nge"], "Marke"... | 1 |
| 7222 | deaeeb3b-3c58-4cf0-8390-2676b1e5d9f4 | 738c128c-fc3e-48ed-93bb-1641e3acdbad | weiß | weiß \| grau \| Weiß \| Grau | PCLAERKE Bluse Langarmshirt | BH | main_supplier_code K70240 $ name_suffix cloud ... | {"Zielgruppe": ["Female"], "Pack": ["2er Pack"... | 0 |

Fig. 5. The matched and unmatched offers training set

## *E. Measuring text similarities and preparing the Training set*

Now, the dataset shown above in figure 5 contains only the textual information but the machine learning model needs some numerical measures with help of which it can classify the match as 0 or 1. Hence we calculate the text similarity score each title, color and description of the offers of both the shops. There are several similarity measures [4] that has been used in this assignment some of which include:

*1) Levenshtein distance:* The Levenshtein distance is a measure of word similarity. The distance between two words indicates how many modifications are required to change one word into another. Editing can be accomplished using one of three methods: Insertion, deletion and substitution. [1]

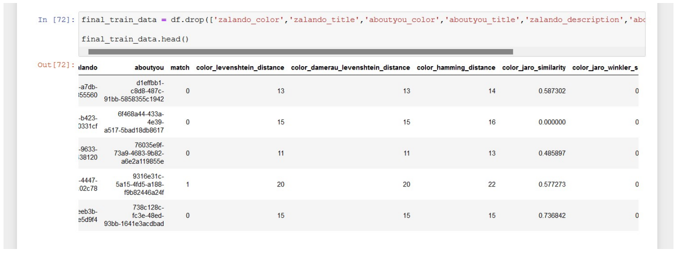
Another version of this is the 'Damerau–Levenshtein distance' which is the smallest number of operations (consisting of insertions, deletions, or replacements of a single character, or transposition of two adjacent characters) necessary to turn one word into the other.

*2) Hamming Distance:* It is the distance is a measurement of how different two bitstrings of the same dimension are. The classic definition of a Hamming distance is as follows: given two bitstrings (more broadly, strings of symbols) of the same dimension, the Hamming distance is the smallest number of symbol modifications required to turn one string into the other. [2]

*3) Token Set Ratio:* The strings are tokenized and preprocessed in token sort ratio by converting to lower case and removing punctuation. After that, the strings are alphabetically sorted and put together. Following that, the Levenshtein distance similarity ratio between the strings is determined.

There are other Fuzzy string matching techniques like matching the sub-string of the two strings which is the 'Partial Ratio' and various other methods have been used to find the similarity score. [3] Apart from that, other similarity measures such as Jaro and Jaro-Winkler similarity, and even matching numbers in both string is also used. Python's 'JellyFish' and 'Fuzzywuzzy' has inbuilt libraries for all these similarity scores.

Finally, the dataset created with the text similarities and dropping all the unnecessary columns is shown in the figure below:

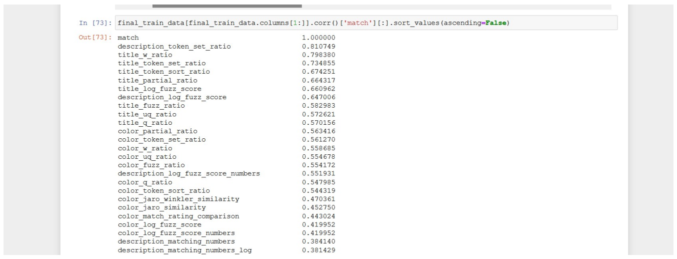


```
In [72]: final_train_data = df.drop(['zalando_color','zalando_title','aboutyou_color','aboutyou_title','zalando_description','abo

         final_train_data.head()
Out[72]:
```

| lando | aboutyou | match | color_levenshtein_distance | color_damerau_levenshtein_distance | color_hamming_distance | color_jaro_similarity | color_jaro_winkler_s |
|---|---|---|---|---|---|---|---|
| -a7db-55560 | d1effbb1-c8d8-487c-91bb-5858355c1942 | 0 | 13 | 13 | 14 | 0.587302 | 0 |
| -b423-0331cf | 6f468a44-433a-4e39-a517-5bad18db8617 | 0 | 15 | 15 | 16 | 0.000000 | 0 |
| -9633-38120 | 76035e9f-73a9-4683-9b82-a6e2a119855e | 0 | 11 | 11 | 13 | 0.485897 | 0 |
| -4447-02c78 | 9316e31c-5a15-4fd5-a188-f9b82446a24f | 1 | 20 | 20 | 22 | 0.577273 | 0 |
| eb3b-e5d9f4 | 738c128c-fc3e-48ed-93bb-1641e3acdbad | 0 | 15 | 15 | 15 | 0.736842 | 0 |

Fig. 6. The Final Training set: Contains different text similarity measures.

The correlation with each of the similarity measures can be seen in the following figure. It is evident that maximum similarity measures are positively correlated with the match.

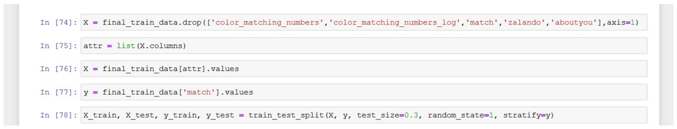


```
In [73]: final_train_data[final_train_data.columns[1:]].corr()['match'][:].sort_values(ascending=False)
Out[73]: match                                       1.000000
         description_token_set_ratio                 0.810749
         title_w_ratio                               0.798380
         title_token_set_ratio                       0.734855
         title_token_sort_ratio                      0.674251
         title_partial_ratio                         0.664317
         title_log_fuzz_score                        0.660962
         description_log_fuzz_score                  0.647006
         title_fuzz_ratio                            0.582983
         title_uq_ratio                              0.572621
         title_q_ratio                               0.570156
         color_partial_ratio                         0.563416
         color_token_set_ratio                       0.561270
         color_w_ratio                               0.558685
         color_uq_ratio                              0.554678
         color_fuzz_ratio                            0.554172
         description_log_fuzz_score_numbers          0.551931
         color_q_ratio                               0.547985
         color_token_sort_ratio                      0.544319
         color_jaro_winkler_similarity               0.470361
         color_jaro_similarity                       0.452750
         color_match_rating_comparison               0.443024
         color_log_fuzz_score                        0.419952
         color_log_fuzz_score_numbers                0.419952
         description_matching_numbers                0.384140
         description_matching_numbers_log            0.381429
```

Fig. 7. The Correlation between Similarity Measures and the Match

## *F. Building the Classification Model*

After preparing the training dataset with similarity scores of each of the offers from zalando and aboutyou. The training data is then splitted up into train and test set. The train set for training the model and the test set for the validation purpose and results evaluation.



```
In [74]: X = final_train_data.drop(['color_matching_numbers','color_matching_numbers_log','match','zalando','aboutyou'],axis=1)

In [75]: attr = list(X.columns)

In [76]: X = final_train_data[attr].values

In [77]: y = final_train_data['match'].values

In [78]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
```

Fig. 8. Splitting the Training set into train and validation or test set

There several classification algorithms in machine learning that can be used. This assignment uses the following classifiers to classify the matches:

1) DummyClassifier
2) KNeighborsClassifier
3) XGBClassifier
4) DecisionTreeClassifier
5) RandomForestClassifier
6) AdaBoostClassifier
7) GradientBoostingClassifier

8) Perceptron
9) MLP

Each of the above models have been fitted with the training and the testing or validation set and the best model is chosen among them.

## III. Evaluation and Results

There are several evaluation metrics that has been used and based on which each classification model has been compared. [4]

1) The Confusion Matrix is a performance metric for machine learning classification tasks with two or more classes as output. As illustrated in the figure below, it's a table with a mix of expected and actual data.

| | | **ACTUAL VALUES** | |
|---|---|---|---|
| | | **MATCH (POSITIVE)** | **NO MATCH (NEGATIVE)** |
| **PREDICTED VALUES** | **MATCH (POSITIVE)** | TRUE POSITIVE (TN) | FALSE POSITIVE (FP) |
| | **NO MATCH (NEGATIVE)** | FALSE NEGATIVE (FN) | TRUE NEGATIVE (TN) |

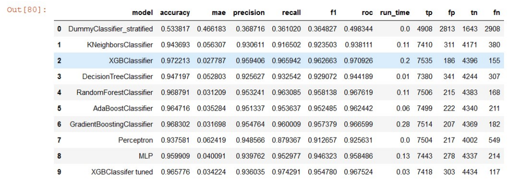
Fig. 9. Confusion Matrix: General Form

We can determine Recall, Precision from the above matrix.

2) Recall : Recall is the proportion of relevant documents that has been retrieved.
$Recall = \frac{\text{True Positive retrieved}}{\text{True Positive + False Negative}}$
3) Precision: It is the proportion of documents retrieved that is relevant.
$Recall = \frac{\text{True Positive retrieved}}{\text{True Positive + False Positive}}$
One must note that both precision and recall are inversely proportional to each other.
4) F1 Score: It gives the combined idea about precision and recall.
$F1 - Score = 2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}}$

Now that we have a better understanding of the evaluation metrics, we can use them to assess the prediction outcomes of each of our categorization models. The evaluation of each of the above-mentioned trained classification models is summarized in the table shown in figure 10.

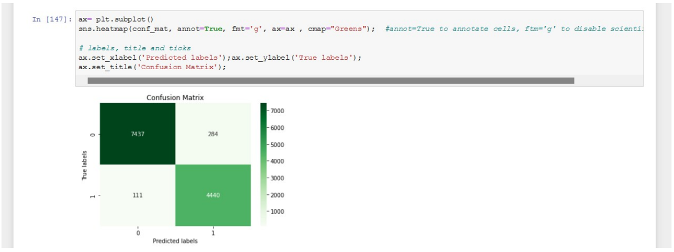
It can be observed that XGBooster Classifier outperforms the other algorithms, with F1 score of 96.59% than the other algorithms. Hence, XGB Classifier can be used to predict the matches in the given testing dataset. The confusion matrix shown below explains the predictions done by the XGB Classifier, it clearly shows that only 400 products have been predicted wrong, which us quite a good result.



Out[80]:

| | 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 | 0.943693 | 0.056307 | 0.930611 | 0.916502 | 0.923503 | 0.938111 | 0.11 | 7410 | 311 | 4171 | 380 |
| 2 | XGBClassifier | 0.972213 | 0.027787 | 0.959406 | 0.965942 | 0.962663 | 0.970926 | 0.2 | 7535 | 186 | 4396 | 155 |
| 3 | DecisionTreeClassifier | 0.947197 | 0.052803 | 0.925627 | 0.932542 | 0.929072 | 0.944189 | 0.01 | 7380 | 341 | 4244 | 307 |
| 4 | RandomForestClassifier | 0.968791 | 0.031209 | 0.953241 | 0.963085 | 0.958138 | 0.967619 | 0.11 | 7506 | 215 | 4383 | 168 |
| 5 | AdaBoostClassifier | 0.964716 | 0.035284 | 0.951337 | 0.953637 | 0.952485 | 0.962442 | 0.06 | 7499 | 222 | 4340 | 211 |
| 6 | GradientBoostingClassifier | 0.968302 | 0.031698 | 0.954764 | 0.960009 | 0.957379 | 0.966599 | 0.28 | 7514 | 207 | 4369 | 182 |
| 7 | Perceptron | 0.937581 | 0.062419 | 0.948566 | 0.879367 | 0.912657 | 0.925631 | 0.0 | 7504 | 217 | 4002 | 549 |
| 8 | MLP | 0.959909 | 0.040091 | 0.939762 | 0.952977 | 0.946323 | 0.958486 | 0.13 | 7443 | 278 | 4337 | 214 |
| 9 | XGBClassifer tuned | 0.965776 | 0.034224 | 0.936035 | 0.974291 | 0.954780 | 0.967524 | 0.03 | 7418 | 303 | 4434 | 117 |

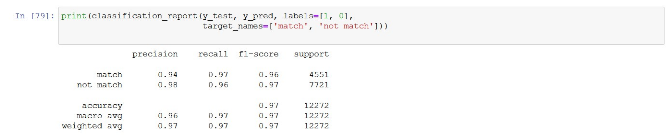
Fig. 10. Confusion Matrix resembling evaluation of each classification model.



```
In [147]: ax= plt.subplot()
          sns.heatmap(conf_mat, annot=True, fmt='g', ax=ax , cmap="Greens");  #annot=True to annotate cells, ftm='g' to disable scienti

          # labels, title and ticks
          ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
          ax.set_title('Confusion Matrix');
```

Fig. 11. Confusion Matrix for XGBoost Classifier

Also, we can note that the the F1 score for match is 96% and that for no match is 97% percent from the classification report below.



```
In [79]: print(classification_report(y_test, y_pred, labels=[1, 0],
                                     target_names=['match', 'not match']))

                    precision    recall  f1-score   support

             match       0.94      0.97      0.96      4551
         not match       0.98      0.96      0.97      7721

          accuracy                           0.97     12272
         macro avg       0.96      0.97      0.97     12272
      weighted avg       0.97      0.97      0.97     12272
```

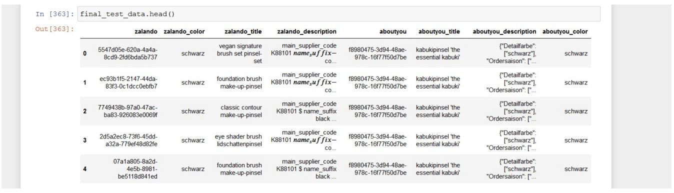
Fig. 12. Classification Report for XGBoost Classifier

Thus, we can use XGBooster to predict the matches for the given test set.

## IV. Working on the Test set: Predicting the Unseen Matches

As we mentioned in section I-A we have a offers_test.parquet file for which the matches has to be predicted. The dataset is loaded the same as above for the training set and it is pre-processed and the similarity for each of the offers in the test set.

The following is the snippet of the test dataset we loaded:



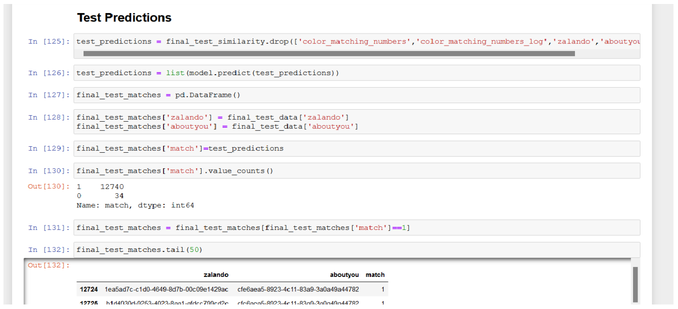
In [363]: `final_test_data.head()`

Out[363]:

| | zalando | zalando_color | zalando_title | zalando_description | aboutyou | aboutyou_title | aboutyou_description | aboutyou_color |
|---|---|---|---|---|---|---|---|---|
| 0 | 5547d05e-620a-4a4a-8cd9-2fd6bda5b737 | schwarz | vegan signature brush set pinsel-set | main_supplier_code K88101 $name_suffix-$ co... | f8980475-3d94-48ae-978c-16f77f50d7be | kabukipinsel 'the essential kabuki' | {"Detailfarbe": ["schwarz"], "Ordersaison": ["... | schwarz |
| 1 | ec93b1f5-2147-44da-83f3-0c1dcc0ebfb7 | schwarz | foundation brush make-up-pinsel | main_supplier_code K88101 $name_suffix-$ co... | f8980475-3d94-48ae-978c-16f77f50d7be | kabukipinsel 'the essential kabuki' | {"Detailfarbe": ["schwarz"], "Ordersaison": ["... | schwarz |
| 2 | 7749438b-97a0-47ac-ba83-926083e0069f | schwarz | classic contour make-up-pinsel | main_supplier_code K88101 $ name_suffix black ... | f8980475-3d94-48ae-978c-16f77f50d7be | kabukipinsel 'the essential kabuki' | {"Detailfarbe": ["schwarz"], "Ordersaison": ["... | schwarz |
| 3 | 2d5a2ec8-73f6-45dd-a32a-779ef48d82fe | schwarz | eye shader brush lidschattenpinsel | main_supplier_code K88101 $name_suffix-$ co... | f8980475-3d94-48ae-978c-16f77f50d7be | kabukipinsel 'the essential kabuki' | {"Detailfarbe": ["schwarz"], "Ordersaison": ["... | schwarz |
| 4 | 07a1a805-8a2d-4e5b-8981-be5118d841ed | schwarz | foundation brush make-up-pinsel | main_supplier_code K88101 $ name_suffix black ... | f8980475-3d94-48ae-978c-16f77f50d7be | kabukipinsel 'the essential kabuki' | {"Detailfarbe": ["schwarz"], "Ordersaison": ["... | schwarz |

Fig. 13. The Test data

The final predictions is done on the above test set and the matches have been predicted and even verified as shown in the images 14 and 15 below:

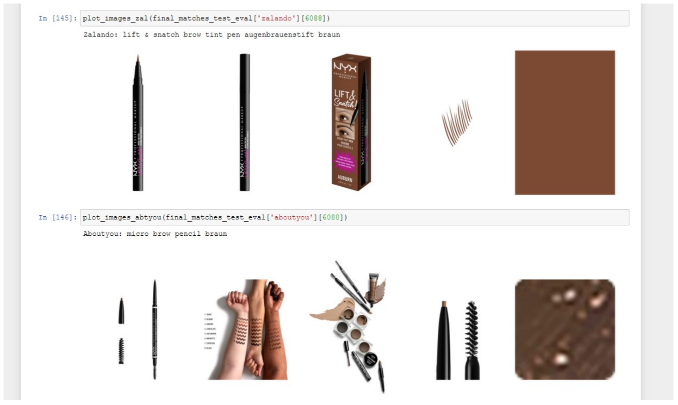


Test Predictions

```
In [125]: test_predictions = final_test_similarity.drop(['color_matching_numbers','color_matching_numbers_log','zalando','aboutyou

In [126]: test_predictions = list(model.predict(test_predictions))

In [127]: final_test_matches = pd.DataFrame()

In [128]: final_test_matches['zalando'] = final_test_data['zalando']
          final_test_matches['aboutyou'] = final_test_data['aboutyou']

In [129]: final_test_matches['match']=test_predictions

In [130]: final_test_matches['match'].value_counts()

Out[130]: 1    12740
          0       34
          Name: match, dtype: int64

In [131]: final_test_matches = final_test_matches[final_test_matches['match']==1]

In [132]: final_test_matches.tail(50)

Out[132]:
```

| | zalando | aboutyou | match |
|---|---|---|---|
| 12724 | 1ea5ad7c-c1d0-4649-8d7b-00c09e1429ac | cfe6aea5-8923-4c11-83a9-3a0a49a44782 | 1 |
| 12725 | b1d4030d-9253-4023-8aa1-afdcc799cd2c | cfe6aea5-8923-4c11-83a9-3a0a49a44782 | 1 |

Fig. 14. Predictions done on Test data



```
In [145]: plot_images_zal(final_matches_test_eval['zalando'][6088])

          Zalando: lift & snatch brow tint pen augenbrauenstift braun
```

```
In [146]: plot_images_abtyou(final_matches_test_eval['aboutyou'][6088])

          Aboutyou: micro brow pencil braun
```

Fig. 15. An example of a match in test data

## V. Conclusion

To summarize, we performed a supervised machine learning job for both zalando and aboutyou by categorizing matched and unmatched products.We started with training the model with the similarity scores of the text strings from the title, color and description of each of the products of the two shops, and classified them using different classification algorithms. Finally, we then predicted the unseen matches present in the given test data.

Apart from this, there were several difficulties encountered during the procedure, such as memory concerns and a time-consuming difficulty due to the big datasets. But still there were predictions done successfully. There is still more that can be done when matching pridućts, we can consider the images also, while matching but this assignement was mainly text focussed,

## References

[1] S. Zhang, Y. Hu and G. Bian, "Research on string similarity algorithm based on Levenshtein Distance," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017, pp. 2247-2251, doi: 10.1109/IAEAC.2017.8054419.

[2] Bookstein, Abraham Kulyukin, Vladimir Raita, Timo. (2002). Generalized Hamming Distance. Information Retrieval. 5. 10.1023/A:1020499411651.

[3] Fuzzy String Matching – A Hands-on Guide by Mimi Dutta — July 15, 2021. https://www.analyticsvidhya.com/blog/2021/07/fuzzy-string-matching-a-hands-on-guide/

[4] Metrics to Evaluate your Classification Model to take the right decisions by Sumeet Kumar Agrawal — July 20, 2021 https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

[5] Product matching algorithms find identical products on ecommerce sites so users can compare products and retailers can compare prices, by Matt Clarke, Saturday, March 13, 2021. https://practicaldatascience.co.uk/machine-learning/how-to-create-a-product-matching-model-using-xgboost