Towards Classifying HTML-embedded Product Data Based On Machine Learning Approach

Oleksandr Matveiev, Anastasiia Zubenko, Dmitry Yevtushenko and Olga Cherednichenko

National Technical University "Kharkiv Polytechnic Institute", Kirpicheva st. 2, Kharkiv, 61002, Ukraine

Abstract

In this paper we explored machine learning approaches using descriptions and titles to classify footwear by brand. The provided data were taken from many different online stores. In particular, we have created a pipeline that automatically classifies product brands based on the provided data. The dataset is provided in JSON format and contains more than 40,000 rows. The categorization component was implemented using K-Nearest Neighbour (K-NN) and Support Vector Machine (SVM) algorithms.

The results of the pipeline construction were evaluated basing on the classification report, especially the Precision weighted average value was considered during the calculation, which reached 79.0% for SVM and 72.0% for K-NN.

Keywords 1

Product classification, SVM, K-Nearest Neighbour, TF-IDF, machine learning, vectorization, item matching

1. Introduction

Today, there is an enormous number of e-shops that allow consumers to buy goods online. As a result, the number of products sold through e-shops grew rapidly. A recent study estimated that total e-commerce retail sales were $791.70 billion in 2020, up 32.4% from the previous year's $598.02 billion. This is the highest annual growth of digital technologies for any year for which data are available this information reported by the Ministry of Trade in 2019 [1]. One of the reasons for this growth was the result of COVID-19, which further increased e-commerce revenue in 2020 by 105.47 billion dollars [1]. For example, web giants such as Amazon reached $100.83 billion in the fourth quarter of 2020, up a whopping 47.5% from $ 68.34 billion a year earlier. This is 2.5 times higher than the level of income on the Internet by 19.5% during the fourth quarter of 2019.

This global trend of e-commerce is forcing all businesses to go online, resulting in an increasing number of e-commerce stores. Each e-commerce store has different streams to publish an added item on the platform. Some markets, such as Amazon, eBay, etc., allow users to become sellers and add products themselves. This functionality permits retailers to increase the number of products they sell. However, the process of adding new products and assigning a category can lead to consistency issues. An error in the classification of the product in the first place can lead to some problems with finding the exact product. Therefore, the correct categorization of products is critical for all e-commerce platforms, as it speeds up the search for the definite product and provides better interaction with users, highlighting the correct categories.

To solve these problems with the assignment of goods to the wrong category, an automatic tool that can classify any product by name in the product taxonomy is needed. At the same time, this process

MoMLeT+DS 2021: 3rd International Workshop on Modern Machine Learning Technologies and Data Science, June 5, 2021, Lviv-Shatsk, Ukraine
EMAIL: matwei1970@gmail.com(O. Matveiev); zubenkoanastasia94@gail.com (A. Zubenko); yevtushenkods@gmail.com (D. Yevtushenko); olha.cherednichenko@gmail.com (O. Cherednichenko)
ORCID: 0000-0001-5907-3771 (O. Matveiev); 0000-00001-9178-0847 (A. Zubenko); 0000-0001-6250-4616 (D. Yevtushenko); 0000-0002-9391-5220 (O. Cherednichenko)

will facilitate human work and further improve the consistency of product categorization on e-commerce websites.

In this paper, we apply some approaches to product categorization for the provided data collection. The data provided were taken from many different online stores. The total amount of data provided in the JSON file is over 40,000 lines. This number of records will allow us to teach the model to predict the category of goods for future products.

2. Related work

This section provides an overview of existing research on product classification based on product specifications that have been studied with different approaches and methods in recent years.

Due to not all websites use a hierarchy of product classification and some of them use but it can be completely different, a unified product classification from different websites is needed in order to provide the user with useful features like browsing and searching.

Although there are several approaches to product data classification [2] introduced a modified Naive Bayesian model for classifying goods, using the usual Bayesian naive instead of a text classifier. Although the accuracy is somewhat high, the main disadvantage of this approach is how to choose the right weight, as it is based on data observation and manual assignment of scales based on selected functions. Failure to select the appropriate weight will significantly change the results. Lin and Shankar [3] investigated using effective pre-treatment methods and multi-class features to improve classification accuracy. The paper [4] discussed the classification process in terms of what a classification was, and they represented a model of SCM semantic classification. In [5] used fuzzy modelling of sets to identify categories, but this model lacked a comparison of classification accuracy for evaluation.

Recently, the categorization of goods using product descriptions by Chen and Warren has aroused great interest [6]. Despite these efforts, there are not many studies aimed at classifying goods by name and description.

3. The product classification pipeline

At an elevated level, the goal for our system is to build a multi-class classifier, which can accurately predict the product category of a new unlabeled product title. The high-level steps are presented in Figure 1.

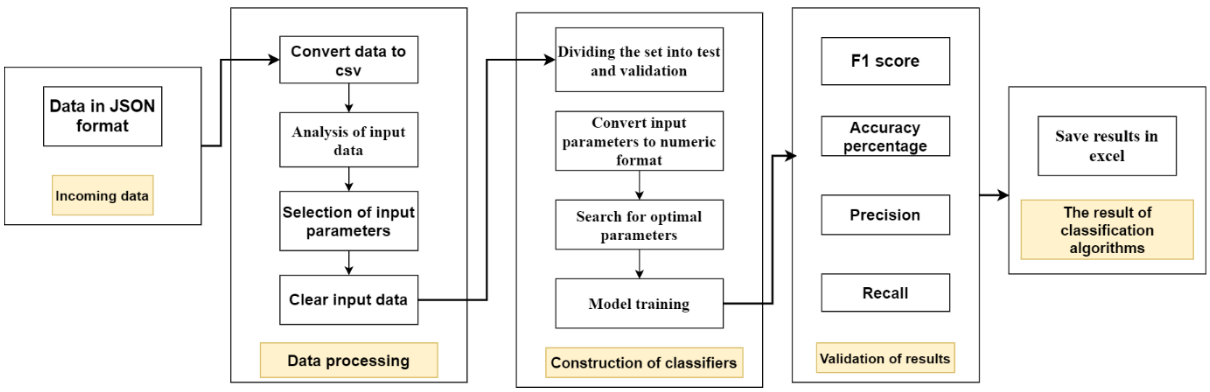

Figure 1: Stages for this classification process

As shown in Figure 1, we performed the following steps to build a classification model:

1. Exploratory data analysis
2. Feature Selection based on the Exploratory data analysis (EDA).
3. Pre-processing
4. Data transformation

a. Removes topic-neutral words such as articles (a, an, the), prepositions (in, of, at), conjunctions (and, or, nor), etc. from the documents.
b. Word stemming
5. Classification models: Multi-Class SVM, K nearest neighbours (K-NN) for the selected features. These two models were selected to compare the discriminative (SVM) and nonparametric models (K-NN).
6. Analysis of the results
The full process is described below.

3.1. Classifiers Overview

The classifier is built basing on the learning from the provided dataset and can be used to classify unknown products by brand in future. We choose two algorithms (K-NN and SVM) to implement. We provide a brief description of each algorithm in this Section.

3.1.1. SVM Based Categorization

SVM is introduced as an algorithm for text classification by Joachims [14]. Let $D_n = \{(\overrightarrow{d_1},\ c_1),\ ...(\overrightarrow{d_n},\ c_n)\}$ be a set of n instances for training, where $\overrightarrow{d_1} \in R^N$, and category $c_i \in \{-1,+1\}$. SVM learns linear decision rules $f(\vec{d}) = sign\{\vec{w}\,\vec{d} + \delta\}$, described by a weight vector w and a threshold δ. If D_n is linearly separable, SVM finds the hyperplane with maximum Euclidean distance to the closest training instances. If D_n is non-separable, the amount of training errors is measured using slack variables ξ_i. Computing the hyperplane is equivalent to solving the following optimization problem [16].

$$minimize: V(\vec{w},\delta,\vec{\xi}) = \frac{1}{2}\vec{w} * \vec{w} + C\sum_{i=1}^{n}\xi_i \tag{1}$$

$$subject\ to: \forall_{n=1}^{n}: c_i[\vec{w} * \vec{w} + \delta] \geq 1 - \xi_i \tag{2}$$

$$\forall_{i=1}^{n}: \xi_i > 0 \tag{3}$$

The factor C in (1) is a parameter used for trading off training error vs. model complexity. The constraints (2) require that all training instances be classified correctly up to some slack ξ_i.

3.1.2. K-NN Algorithm

The K-Nearest Neighbour (K-NN) is one of the popular algorithms [15, 16]. The algorithm is based on finding the most similar objects from sample groups about the mutual Euclidean distance [7, 8].
The algorithm assumes that it is possible to classify documents in the Euclidean space as points [17]. The distance between two points can be calculated as following:

$$d(p,q) = d(q,p) = \sqrt{(x-a)^2} + (y-b)^2 \tag{4}$$

3.2. Exploratory Data Analysis

3.2.1. Convert a file from JSON format to CSV

First of all, it is necessary to convert the input format to CSV. This format is more common in Python and gives us more opportunities to work with data.

To do this, we installed an additional library of pandas. We did this with the following command: pip install pandas.

This library contains the read_json () method, which allows you to upload a file to the program and continue working with it. The read_json () method can take several parameters but we used only one: path_or_buf. This parameter is responsible for the path to our JSON file. This library contains the read_json() method, which allows you to upload a file to the program and continue working with it.

Once we download the file data to the program's memory, we can start working on it. The data downloaded to the program's memory can be written to a CSV file, using the following method - to_csv(). In this method, we passed the path where we wanted to place our CSV file as a parameter.

The code needed to convert a file from JSON to CSV can be found in the convert.py script. Run the file with the following command: python convert.py.

3.2.2. Input analysis

After we have converted the input file, we can start its analysis. The input file contains 41,664 records and 17 columns:

```
brand               object
category            object
cluster_id           int64
description         object
id                   int64
keyValuePairs       object
price               object
specTableContent    object
title               object
/gtin8              object
/gtin13             object
/identifier         object
/gtin14             object
/mpn                object
/gtin12             object
/sku                object
/productID          object
dtype: object
```

Figure 2: All columns in the input file

Consider the source data contained in the tables. The data is presented in Figures 3 and 4.

A	B	C	D	E	F	G	H	I
	brand	category	cluster_id	description	id	keyValuePairs	price	specTableConte
4000034	baseball monkey	Shoes	6973042	perfect for off fie	4271963		usd 99 99	
4000041	blondo	Shoes	13952277		4271970			
4000113	jessica simpson	Shoes	6004853	strut your stuff a	4272044			
4000140	billabong	Shoes	15168440	features	4272071		44 9 eur	
4000179		Shoes	12849105		4272113			
4000209		Shoes	13921865		4272145		44 usd	
4000216		Shoes	8387741		4272152		69 99 usd	men s apparel x
4000217		Shoes	5195630		4272153		66 58 usd	
4000262		Shoes	13918557	mi kka i lekka ch	4272198			
4000271		Shoes	7589636		4272207		usd 499 99	
4000280		Shoes	7885318	men s nike revol	4272216			

Figure 3: All source data in columns A-I

J	K	L	M	N	O	P	Q	R
name	/mpn	/productID	/sku	/gtin8	/gtin12	/identifier	/gtin13	/gtin14
new balance fresh foam zante team men s training			[34328296]					
farima fudge blondo b1983 92 women s dress boot			[34328296]					
jessica simpson silea high heel belk			[34328296]					
billabonglaneway canvas backpack for men black			[34328296]					
shoes men tomford com			[34328296]					
			[34328296]					
s s m l xl xxl neck 14 14 5in 35 6 36 8cm 15 15 5in			[34328296]					
			[34328296]					
buty mizuno morelia fg czarny bia y czerwony pi ka			[34328296]					
			[34328296]					
men s nike revolution 2 running shoes mens peltz			[34328296]					
khaki men increasing wool lining shoes grow tall 7c			[34328296]					
francesco furini 1603 1646 italy			[34328296]					
dc shoes beryle comprar y ofertas en dressinn			[34328296]					
dallas cowboys fanband womens headband shop			[34328296]					
hennessy hammock expedition zip			[34328296]					
			[34328296]					
charpe en coton bross			[34328296]					

Figure 4: All source data in columns J-H

We focused on each of the provided columns separately. This is important because a more detailed analysis allowed us to understand exactly how to configure the script for automatic data processing.

The amount of zero data in the tables was analysed, the result is presented in Figure 5.

```
df_products.isnull().sum()

brand               25617
category                0
cluster_id              0
description         19904
id                      0
keyValuePairs       41106
price               27438
specTableContent    38302
title                7159
/gtin8                  0
/gtin13                 0
/identifier             0
/gtin14                 0
/mpn                    0
/gtin12                 0
/sku                    0
/productID              0
dtype: int64
```

Figure 5: Sums of zero data in the initial columns

Analysing Figure 5, we concluded that the data contains many zero values, but this function calculates the sum of zero values. Therefore, if there are no records in the column, the sum of the zero values will not be found correctly. The proof of this issue is presented in Figures 3 and 4 where we can see the empty columns. Thus, before deleting the null rows the additional manual examination for the columns is required. The result of our additional analysis is presented in Figure 6

3.3. Feature Selection based on the Exploratory Data Analysis

Based on the data analysis stage, we identified columns that were used for further modelling. Thus, for the machine learning model, we used: title, description, and brand. The example of the columns and the data they contain is presented in Figure 7.

Name	Description	Necessity: Yes / No
without name	Create only unique information	No
brand	Contain the name of the shoe brand. Not all lines contain these values.	Yes
category	Contains only duplicates	No
cluster_id	Contains unique values (except for duplicates)	No
category	Contains only duplicates.	No
description	Contains product description.	Yes
id	Contains unique values.	No
keyValuePairs	Data is mostly omitted.	No
price	Contains the price of the product. Not all lines contain these values.	No
specTableContent	Data is mostly skipped.	No
name	Product title.	Yes
/ mpn	Does not contain data.	No
/ productID	Does not contain data.	No
/ sku	Contains only duplicates.	No
/ gtin8	Does not contain data.	No
/ gtin12	Does not contain data.	No
/ gtin14	Does not contain data.	No
/ gtin13	Does not contain data.	No
/ ID	Does not contain data.	No

Figure 6: Additional analysis of columns in the input file and analysis of the need for columns for further use

brand	description	title
baseball monkey	perfect for off field conditioning runs the slee	new balance fresh foam zante team men s training shoes black royal running and
blondo		farima fudge blondo b1983 92 women s dress boot shoes walking on a cloud
jessica simpson	strut your stuff as soon as step onto the stree	jessica simpson silea high heel belk
billabong	features	billabonglaneway canvas backpack for men black
		shoes men tomford com
	mi kka i lekka cholewka wykonana z najwy sz	buty mizuno morelia fg czarny bia y czerwony pi karskie zwarta murawa lanki
	men s nike revolution 2 running shoes a light	men s nike revolution 2 running shoes mens peltz
		khaki men increasing wool lining shoes grow tall 7cm 2 75inches cheap on sale at topoutshoes com
wahooart com francesco furini		francesco furini 1603 1646 italy
dc shoes	descripci n dc shoes beryle traducci n genera	dc shoes beryle comprar y ofertas en dressinn
		dallas cowboys fanband womens headband shop
		hennessy hammock expedition zip
		charpe en coton bross

Figure 7: Columns to be used for further analysis

Based on the provided example in Figure 6, we concluded that the existing data cannot be used for appropriate product categorization, because of the:

- a large number of empty values;
- data duplication.

Therefore, before approaching the categorization of data, we decided to proceed with further cleaning. So, we developed the component which cleans input data automatically.

3.4. Pre-Processing. Automatic Cleaning for the Input Data

Since our solution in future will work with the new data itself, we developed a script that cleans the data automatically.

First of all, we removed cells that contain empty values. Otherwise, the algorithm cannot process the data correctly. To do this, we used the dropna() method that comes with the pandas() package. This method automatically deleted empty cells.

Next, duplicates are removed with the drop_duplicates() method. For this method to process the current file (and not return a new one), set the inplace = TRUE parameter. Since the input data will be obtained from several resources, we need to process them further.

The HTML tags were removed, as there was a risk that they might be in our sample. It was done using the methods BeautifulSoup() and get_test() from the bs4 library.

Then the special characters which could be in these data were removed. The library re and the sub() method were imported. As the first parameter, we passed the following pattern: [^ a-zA-Z \ d].

The next step was to transform all the text data into lowercase and broke it down into words. To do this we used two lower() and split() methods.

After that, the "stop words" can be applied, for that the stopwords() function was used. This function takes one argument: the language we work with. As this argument, we transferred the value "English feature". This set parameter analyzes the language in each cell and removes all non-English rows.

To start automatic cleaning of input data, you should run python clear.py script that contains all the steps described above. After executing the submitted script, our document contains 3 columns and 10,200 unique cleaned lines. An example of the processed data is shown in Figure 8.

brand	description	title
quiksilver	caracteristicas	quiksilverclassic mug shot camiseta para hombres blanco
o neill	caracteristicas	o neill365 energize hipster bikini parte de abajo para mujeres negro
the dudes	caracteristicas	the dudesdirty doods sudadera para hombres negro
element	caratteristiche	elementmineral font t shirt per uomo bianco
g star	caract eacute ris	g stararc 3d low boyfriend jean pour femme bleu
the north face	features	the north faceendeavor thermoball functional jacket for women black
the north face	features	the north facenfz functional jacket for men red
rvca	features	rvcacompound t shirt for men grey
nike	turbo green jlp w	wmns air mogan 2 kixpress com nike 386615 300
naketano	caracteristicas	naketanoblack italienischer hengst langen sudadera para hombres negro
dickies	caracteristicas	dickiesknoxville botines para hombres marr n
roxy	caratteristiche	roxysassy giacca snowboard per donna multicolore
burton	specificaties	burtonak gore clutch snowboard handschoen voor heren bruin
dc	features	dcseger snowboard gloves for men black
max q com	caract eacute ris	max q comleg warmer accessoire noir
volcom	caracteristicas	volcomrun around ringer camiseta de tirantes para mujeres rosa
o neill	features	o neillbeach break shirt for men pink
anon	descripci n parcł	anonwallace gorro para hombres negro
element	features	elementmason knitted pullover for men blue
volcom	caratteristiche	volcomvorta tapered jeans per uomo nero

Figure 8: Example of processed data

Thus, after processing 41,664 lines, 10,200 lines were left, which is 24,48% of the initial dataset.

3.5. Data Transformation. Text Vectorization

Machine learning algorithms usual operate on a numeric feature space. To perform the algorithm on the text, we transformed our text data into vector representations. It is called feature extraction or vectorization [9].

In this paper, we evaluated performance of two methods HashingVectorizer, CountVectorize which are used for converting the collection of text data to a matrix of token counts and TfidfVectorizer method for converting a collection of raw data to a matrix of TF-IDF features.

HashingVectorizer and CountVectorizer are meant to do the same thing, which is to convert a collection of text documents to a matrix of token occurrences. [10]. Term frequency-inverse document frequency (TF-IDF) is a feature vectorization method used to reflect the importance of a term to a document in the corpus [11]. TFIDF can be calculated as:

$$a_{ij} = tf_{ij} idf_i = tf_{ij} \times log_2 \left(\frac{N}{df_i}\right) \tag{5}$$

where a_{ij} is the weight of term i in document j, N is the number of documents in the collection, tf_{ij} is the term frequency of term i in document j and df_i is the document frequency of term i in the collection.

To obtain better results with documents of different length, we used a modified equation [14]:

$$a_{ij} = tf_{ij}idf_i = \frac{f_{ij}}{\sqrt{\sum_{s=1}^{N}(fidf(a_{sj}))^2}} \times log_2\left(\frac{N}{td_i}\right) \quad (6)$$

Each row is converted into appropriate representation and applied to training, validation, and classification phases.

The vectorization also allows us to calculate the number of unique categories which we are going to classify, as a result, we performed with 323 different classes.

3.6. Modelling Classification Algorithms

Both algorithms will process with the features selected in Section 3.3.

For selected, cleaned features we applied the vectorization CountVectorizer function which was done during applying the StratifiedKFold method which splits the dataset into the 3.6 test groups. The selected vectorization function allows us to evaluate the performance of the built model and compare the results we got with applying K-NN. After we applied the vectorization function, the next step was determining the optimal value of the C parameter. The evaluation for selected C parameters is presented in Section 3.7.1.

As we selected features, we vectorized the data. For the K-NN model, we evaluated the performance by applying each vectorizing functions described in Section 3.5. The evaluation is done in Section 3.7.2.

In the next step, we determined the K value. To determine the vectors distance between the data for K-NN, we used both Cosine similarity and Euclidean space.

3.7. Models Evaluation

The method of stratified cross-validation kfold (Stratified kfold cross validation) was used to assess the quality of the model at the initial stage. Choosing between regular cross-checking kfold and stratified cross-checking. The kfold check selected a stratified kfold cross check. Because we have unbalanced data, stratified kfold cross-checking is useful for our experiment. It was decided not to use regular kfold cross-checking because we do not have enough data and this method often preserves the ratio of classes, and this can lead to partitioning in such a way that some networks will contain examples of training from only one class.

Stratified cross-checking is suitable for assessing the quality of a classifier without the use of test data. Testing occurs from parts of the training sample that are not known to the classifier. This assessment approach helps determine if the system is capable of relearning. In our experiment, we used a stratified cross-check with k bends (k = 6) for 10,200 products and 323 categories. Therefore, the evaluation was done for the 1703 products.

The evaluation was performed in several test phases:

- quality classification;
- speed text classification;
- classifications recall according to categories of product.

Classification results, reported in this section, were based on the evaluation which was done according to F1–measure, precision, recall, and accuracy metrics [19]. To evaluate the overall performance of the algorithms on given datasets we focused on the F1 macro average. F1 macro average calculates the score separated by class but not using weights for the aggregation. The F1 weighted average calculates the score for each class independently but when it adds them together uses a weight that depends on the number of true labels of each class. Therefore, F1 weighted average favoring the majority class which we do not want.

3.7.1. SVM Model Evaluation

We applied different values for the C parameter to ensure that the experimental results faithfully reflect the performance of the algorithms.

Parameter C	Total goods	Goods with the correct classification	Goods with incorrect classification	Percentage of correct classification	Percentage of misclassification	Metrics	Precision	Recall	F1-score	Support	Accuracy percentage	Execution time (sec)
0,03125	10218	7864	2353	76,96%	23,04%	macro	0,71	0,70	0,69	1 703	0,78	138,5
						weighted	0,77	0,78	0,76	1 703	0,78	138,5
0,0625	10218	78696	2321	77,28%	22,72%	macro	0,66	0,67	0,65	1 703	0,77	197,2
						weighted	0,76	0,77	0,75	1 703	0,77	197,2
0,125	10218	8012	2205	78,41%	21,59%	macro	0,73	0,73	0,72	1 703	0,80	310,8
						weighted	0,79	0,80	0,78	1 703	0,80	310,8
0,25	10218	7946	2271	77,77%	22,23%	macro	0,72	0,71	0,70	1 703	0,78	535,4
						weighted	0,79	0,79	0,78	1 703	0,78	535,4
1	10218	7839	2378	76,72%	23,28%	macro	0,69	0,69	0,68	1 703	0,76	1622,7
						weighted	0,77	0,76	0,76	1 703	0,76	1622,7
2	10218	7812	2405	76,46%	23,54%	macro	0,69	0,67	0,66	1 703	0,76	2585,7
						weighted	0,77	0,76	0,75	1 703	0,76	2585,7

Figure 9: SVM result

From the experimental result of the SVM, the C parameter equals 0,125 is optimal based on the execution time of 310.8 sec which is 5.5 min and the macro average for F1-score is 72%.

Also, for measuring the performance we calculated the number of goods with correct and incorrect classification based on that the percentage of correct and misclassified categories was found. So, the algorithm creates a separate file for initial and classified values, and automatically compares values. Then this function calculates the sum of correct and incorrect predicted values and percentage accordingly.

The output of this function is presented in Figure 8. The comparison is presented in Figure 10. As we selected features, we vectorized the data. For the K-NN model, we evaluated the performance by applying each vectorizing functions described in Section 3.5. The evaluation is done in Section 3.7.2.

In the next step, we determined the K value. K value of the K-NN algorithm is a factor that indicates a required amount of data from the collection which is closest to the selected row. To determine the vectors distance between the data for K-NN, we used both Cosine similarity and Euclidean space.

394	272	1264
395	1264	1264
396	1264	1264
397	1264	1264
398	1264	1264
399	541	1264
400	1264	1264
401	1264	1264
402	1264	1264
403	1264	1264
404	1264	1264
405	794	1264
406	1264	1264
407	541	1264

Figure 10: SVM incorrect classification calculation

3.7.2. K-NN Model Evaluation

Various scaling methods were used to evaluate the efficiency of the model, such as the similarity of cosines and Euclidean space. The final analysis of the model efficiency is analyzed based on the chosen method. Figures 11-13 represent some results of our experimets.

classifier	scale_method	max_f	k_clusters	nbrs	Metric	Correctly Classified Instances	Incorrectly Classified Instances	Inaccurate percentage	Precision	Recall	F1-score	Accuracy percentage	Support	Execution time (sec)
KNeighborsClassifier	tf-idf	1500	3	3	euclidean	6330	3888	38,05	0,58	0,60	0,57	0,58	1 703	7,44
KNeighborsClassifier	tf-idf	1500	3	3	cosine	6396	3822	37,40	0,64	0,63	0,62	0,63	1 703	9,05
KNeighborsClassifier	tf-idf	2500	3	3	euclidean	6357	3861	37,79	0,61	0,64	0,58	0,61	1 703	8,54
KNeighborsClassifier	tf-idf	2500	3	3	cosine	6477	3741	36,61	0,66	0,66	0,63	0,65	1 703	9,43
KNeighborsClassifier	tf-idf	5000	3	3	euclidean	6305	3913	37,30	0,65	0,66	0,59	0,63	1 703	7,76
KNeighborsClassifier	tf-idf	5000	3	3	cosine	6514	3704	34,05	0,70	0,69	0,66	0,69	1 703	8,87
KNeighborsClassifier	tf-idf	10000	3	3	euclidean	6388	3830	36,48	0,68	0,66	0,66	0,67	1 703	7,68
KNeighborsClassifier	tf-idf	10000	3	3	cosine	6644	3573	31,98	0,72	0,68	0,70	0,70	1 703	9,04

Figure 11: K-NN model with tf-idf scale method

classifier	scale_method	max_f	k_clusters	nbrs	Metric	Correctly Classified Instances	Incorrectly Classified Instances	Inaccurate percentage	Precision	Recall	F1-score	Accuracy percentage	Support	Execution time (sec)
KNeighborsClassifier	HashingVectorizer	1500	3	3	euclidean	6237	3981	48,20	0,53	0,55	0,53	0,55	1 703	6,18
KNeighborsClassifier	HashingVectorizer	1500	3	3	cosine	6341	3877	37,20	0,64	0,64	0,62	0,63	1 703	7,41
KNeighborsClassifier	HashingVectorizer	2500	3	3	euclidean	6253	3965	38,79	0,54	0,56	0,54	0,55	1 703	6,39
KNeighborsClassifier	HashingVectorizer	2500	3	3	cosine	6396	3822	36,81	0,68	0,63	0,63	0,65	1 703	7,38
KNeighborsClassifier	HashingVectorizer	5000	3	3	euclidean	6272	3946	38,30	0,61	0,60	0,59	0,60	1 703	6,06
KNeighborsClassifier	HashingVectorizer	5000	3	3	cosine	6456	3762	34,05	0,68	0,66	0,64	0,66	1 703	7,51
KNeighborsClassifier	HashingVectorizer	10000	3	3	euclidean	6259	3959	36,48	0,62	0,64	0,62	0,63	1 703	6,06
KNeighborsClassifier	HashingVectorizer	10000	3	3	cosine	6422	3796	32,87	0,70	0,68	0,66	0,68	1 703	7,34

Figure 12: K-NN model with HashingVectorizer scale method

classifier	scale_method	max_f	k_clusters	nbrs	Metric	Correctly Classified Instances	Incorrectly Classified Instances	Inaccurate percentage	Precision	Recall	F1-score	Accuracy percentage	Support	Execution time (sec)
KNeighborsClassifier	CountVectorizer	1500	3	3	euclidean	4764	5454	51,06	0,47	0,51	0,49	0,49	1 703	7,40
KNeighborsClassifier	CountVectorizer	1500	3	3	cosine	5366	4852	43,48	0,59	0,55	0,55	0,56	1 703	8,63
KNeighborsClassifier	CountVectorizer	2500	3	3	euclidean	4851	5367	47,52	0,51	0,55	0,54	0,53	1 703	7,48
KNeighborsClassifier	CountVectorizer	2500	3	3	cosine	5415	4803	40,98	0,60	0,63	0,57	0,60	1 703	8,73
KNeighborsClassifier	CountVectorizer	5000	3	3	euclidean	4796	5422	43,06	0,55	0,59	0,57	0,57	1 703	8,18
KNeighborsClassifier	CountVectorizer	5000	3	3	cosine	5495	4723	39,22	0,66	0,60	0,56	0,61	1 703	9,33
KNeighborsClassifier	CountVectorizer	10000	3	3	euclidean	4845	5373	42,58	0,57	0,57	0,59	0,58	1 703	7,67
KNeighborsClassifier	CountVectorizer	10000	3	3	cosine	5632	4586	38,88	0,66	0,60	0,58	0,62	1 703	9,05

Figure 13: K-NN model with CountVectorizer scale method

Based on the K-NN models evaluation results, the best result for classification by a brand we got while using the vectorization method TfidfVectorizer and cosine similarity metric, where the macro average for F1 is 70%. The number of goods with correct and incorrect classification and the percentage of correct and misclassified categories were calculated as the same for SVM presented in Figure 10.

Also, we can see that the execution time which is 9,04 sec for the best result depends on the selected scale method, metrics and the number of features used for the elevation.

Therefore, we can conclude that if the number of input features is increased, the execution time could become critical, and another faster model can be used.

4. Conclusion

In this paper, we present an investigation of two widely used approaches for text categorization K-NN and the SVM algorithms.

The main goal of the research was to evaluate the performance of two popular K-NN and SVM algorithms, compare execution time for both of them and to develop an MVP pipeline that can automatically classify the shoes category based on the brand.

The combination of the K-NN algorithm and different vectorization methods showed good results as well as SVM and CountVectorizer. However, despite the good performance results of the SVM algorithm, it has the highest execution time, which can be significant for big marketplaces.

Therefore, the gained results which are reported in this paper are satisfactory, however, they are not the best that can be achieved. Moreover, additional investigation is needed to improve the performance of applied algorithms.

To further study and improve the model, the following steps are suggested:

- Get more data to test models;
- Implement of the algorithm for automatic search of optimal arameters;

- Prepare the developed module for integration with e-commerce stores.

5. References

[1] Quarterly retail e-commerce sales in the last quarter of 2020. US Digital Commerce Bureau News (2020) https://www.digitalcommerce360.com/article/quarterly-online-sales/

[2] Kim, Young-Gon Modified naïve bayes classifier for e-catalog classification, Seoul 151-742.

[3] I. Lin, S. Shankar Applying Machine Learning to Product Categorization Stanford University. CS229.

[4] Kim Dongkyu, Sang-Goo Lee, Jonghoon Chun, Juhnyoung LeeA semantic classification model for e-catalogs.: Proceedings - IEEE International Conference on E-Commerce Technology, CEC 2004, p 85-92, (2004).

[5] Wan, Hongxin; Peng, Yun A technique of e-commerce goods classification and evaluation based on fuzzy set. Proceedings, International Conference on Internet Technology and Applications, ITAP (2010).

[6] Jianfu Chen, David Warren. Cost-sensitive learning for large-scale hierarchical classification. In Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management, CIKM, pages 1351–1360 (2013).

[7] S.Tan, Neighbor-weighted K-nearest neighbour for unbalanced text corpus, Expert Systems with Applications 28 (2005) 667–671.

[8] M.Lan, C.L.Tan, J.Su, Y.Lu, Supervised and Traditional Term Weighting Methods for Automatic Text Categorization, IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 31, NO. 4, (2009).

[9] Text Vectorization and Transformation Pipelines. Chapter 4. https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html

[10] HashingVectorizer, CountVectorizer https://kavita-ganesan.com/hashingvectorizer-vs-countvectorizer/

[11] H. Uguz, A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm, Knowledge-Based Systems 24 (2011) 1024–1032.

[12] J. T.-Y. Kwok, Automatic Text Categorization Using Support Vector Machine, Proceedings of International Conference on Neural Information Processing, (1998) 347-351.

[13] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features, In Proceedings of 10th European Conference on Machine Learning, Chemnitz, Germany, pages 137-142 (1998).

[14] Joachims, T. A Statistical Learning Model of Text Classification for Support Vector Machines. In Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval, pages 128-136 (2001).

[15] Wang, X. Li, An improved KNN algorithm for text classification, (2010).

[16] G. Guo, H.Wang, D.Bell, Y. Bi, K. Greer, KNN Model-Based Approach in Classification, (2003) 986 – 996.

[17] Ming-Yang Su, Using clustering to improve the KNN-based classifiers for online anomaly network traffic identification, Journal of Network and Computer Applications 34 (2011) 722–730.

[18] K. Mikawa, T. Ishidat, M.Goto, A Proposal of Extended Cosine Measure for Distance Metric Learning in Text Classification, 2011.

[19] Sebastiani, F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol.34, No.1, March 2002, pages 1-47 (2002).