ECOICOP

classification

Organisation: Statistics Poland

Author(s): Marta Kruczek-Szepel, Krystyna Piątkowska

Date: 20.07.2020

Version: 1.1

Table of Contents

1. Background and why and how this study was initiated

The Statistics Poland (also known as the Central Statistical Office – Główny Urząd Statystyczny – GUS) is Poland's chief government executive agency charged with collecting and publishing statistics related to the country's economy, population and society, at the national and local levels. The regional units are located across voivodeships and are obligated to report statistical information to the President of GUS consistently.

The subject of HICP (*Harmonised Index of Consumer Prices*) and product classification is mainly dealt with by the Trade and Services Department in GUS and some work is conducted in the Statistical Office in Opole (Opole voivodeship).

On March 23rd 2019 we were asked in the Statistical Office in Poznań (Greater Poland voivodeship) to take part in the UNECE High Level Group for Modernisation of Official Statistics – Machine Learning Project. The subject of the study was suggested by the Director of our Statistical Office. Our goal was to test if it is possible to automate the ECOICOP (*European Classification of Individual Consumption by Purpose*) product classification process and replace the manual classification by the machine learning methods (the classification is necessary to calculate the HICP). All the process has been conducted by the Data Engineering Department team of the Statistical Office in Poznań - Marta Kruczek-Szepel and Krystyna Piątkowska. Moreover, Anna Jaborska helped with web scraping and labeling the data at the beginning. The person who has designed the application to classify products was Maja Kramarz.

It is worth highlighting that we had not have any prior experience in machine learning but we were open to learn and eager to collect all the essential information by ourselves. We read a lot of internet articles and participated in online machine learning courses (Udemy, Coursera). We looked also into the books and went to the data science conference in Warsaw and applied machine learning conference in Poznań. We took any opportunity to gain new information and experience from the ML world. Being occupied with experimenting with the data and looking for the best solutions, let us learn whole new theory about machine learning process. However, we were more focused on the practical use of the models and wanted to make it available not only to advanced programmers but also the beginners.

We prepared the machine learning tutorial for the beginners on colab: https://colab.research.google.com/drive/1Epn2NeFRuFC_XyXtQ4qezGVBA5aAzqIh
We are working on the application that can help users with products classification when they have no experience with programming. (More details in point 11.)

The participation in the UNECE High Level Group for Modernisation of Official Statistics – Machine Learning Project was a value added for our team. We could share our experience with people from other countries and watch their successes in the machine learning field. We could compare our solutions and ask questions if something was unclear. The leaders of the project are very supportive and they encourage us to work hard.

2. Data

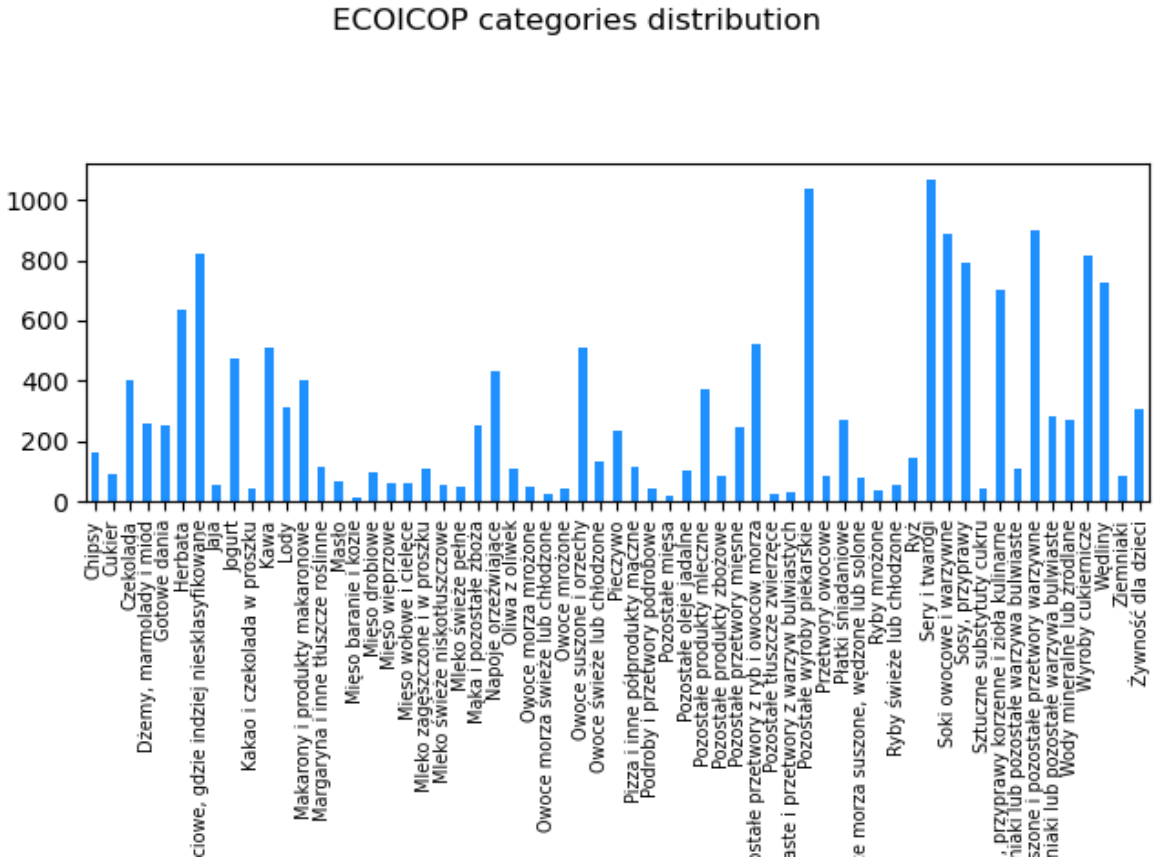
2.1 Input Data

We do not have access to the actual data used to count the HICP and we do not know the process of product classification. We had to collect our own dataset and find the best solution to classify it with machine learning methods.

Our main dataset is the list of the products' names web scraped from online shops. It counts about 17 000 unique names. All the products are classified in compliance with European Classification of Individual Consumption according to Purpose (ECOICOP) manually by no-experts. This fact can pose a serious impact on the results. We are still finding many errors and misclassified products. We hope that with the correctly classified set of data, the results could be only better.

The web scraped data will not be used in production. The dataset was created only to test the possibility of classification with the usage of machine learning methods. The data

is very imbalanced, as you can see at the histogram:



There are categories that contain numerous products (for example: cheese, other bakery products or dried vegetables and other vegetable preparations) and there are categories with only few examples (for example: sheep and goat meat, other animal fat and fresh or chilled seafood). We can deduct that people prefer buying bread or cheese to some unpopular meat.

The first few lines of data file (in polish language):

PRODUKT	KATEGORIA
"słynne roślinne"	Margaryna i inne tłuszcze roślinne
#Hejki - Emotki lizaki ręcznie robione o smakach owocowych	Wyroby cukiernicze
100% Pur jus d'orange - sok pomarańczowy z miąższe...	Soki owocowe i warzywne
100% sukraloza bez cukru (substancje słodzące)	Sztuczne substytuty cukru
100% z brzoskwiń produkt owocowy słodzony zag.sokiem winogronowym	Dżemy, marmolady i miód
100% z czarnych porzeczek produkt owocowy słodzony zag.sokiem winogronowym	Dżemy, marmolady i miód

2be BIO - jagoda mrożona bio	Owoce mrożone
2be BIO - jeżyna mrożona bio	Owoce mrożone
2be BIO - malina mrożona bio	Owoce mrożone
3Bit Baton w czekoladzie mlecznej z nadzieniem mlecznym i herbatnikiem 46 g	Czekolada
3in1 Napój kawowy w proszku 180 g (10 saszetek)	Kawa
3in1 Napój kawowy w proszku 360 g (20 saszetek)	Herbata
4 health - Baton musli z orzech i pistacją na czekoladzie des...	Wyroby cukiernicze
4Health Musli - Crunchy Red Fruit chrupiące musli z nasionami chia... mieszanką owoców	Płatki śniadaniowe
4Health Musli - Crunchy chrupiące musli czekoladowe z orzechami i ... chia	Płatki śniadaniowe
4Move - Napój Blueberry	Napoje orzeźwiające
4Move Active Water Magnez + Witaminy Napój niegazowany o smaku cytrusowym 556 ml	Napoje orzeźwiające
4Move Active Water Magnez + Witaminy Napój niegazowany o smaku wiśniowym 556 ml	Napoje orzeźwiające
4Move Lemon napój izotoniczny	Napoje orzeźwiające

In each machine learning process there is a need to divide the dataset into three groups:

- **training dataset** - the sample of data used to fit the model,
- **validation dataset** - the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters,
- **test dataset** - the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

When it comes to the dataset for the ECOICOP classification we start with the random reindexing of all the data. Then we divide it into the three groups. The test dataset constitutes the first 1500 rows of the source file, next 1500 rows is a validation dataset, and the last part constitutes the training dataset. The code below represents the division mentioned above:

```
df = df.reindex(np.random.permutation(df.index))
# Dividing the data into the training, validation and testing group
df_test = df[0:1500].copy()
```

```
df_validation = df[1500:3000].copy()
df_training = df[3000:].copy()
```

It is worth mentioning that our dataset has been translated into english and french by the leader of the project - Claude Julien. It is available online and can be used to test different classification solutions.

2.2 Data Preparation

2.2.1 Vectorization

Vectorization is carried out by CountVectorizer or TfidVectorizer depending on the algorithm. We wanted to test whether it helps to improve model evaluation measures or not. It does, however very slightly. Both vectorizers come from the Sci-kit learn package for Python.

Generally, vectorizers are methods for converting text data into vectors as the model can process only numerical data. Below you can see an example of how it works:

id	Nazwa produktu
1	RYŻ BIAŁY DŁUGOZIARNISTY SUPREME 1KG KAR
2	SER BIAŁY CAPREGGIO SOTTILE GUSTO 1KG

id	RYŻ	BIAŁY	DŁUGOZIARNISTY	SUPREME	1KG	KAR	SER	CAPREGGIO	SOTTILE	GUSTO
1	1	1	1	1	1	1	0	0	0	0
2	0	1	0	0	1	0	1	1	1	1

In CountVectorizer we only count the number of times a word appears in the document (one product name) and in the corpus (all term occurrences in our dataset) which results in biasing in favour of most frequent words. This ends up in ignoring rare words which could have helped in processing our data more efficiently. So we re-weight the count feature vectors using the tf-idf transform method and then feed the data into classifier for better classification. That's how the TfidfVectorizer works - it combines all options of CountVectorizer and TfidfTransformer in a single model.

TfidfVectorizer considers overall document weightage of a word. It helps us in dealing with most frequent words. Using it we can penalize them. TfidfVectorizer weights the word

counts by a measure of how often they appear in the documents (all product names from the data source).

$$\text{Term Frequency (TF)} = \frac{\text{Frequency of a term in the document}}{\text{Total number of terms in documents}}$$

$$\text{Inverse Document Frequency (IDF)} = \log\left(\frac{\text{total number of documents}}{\text{number of documents with term t}}\right)$$

$$\text{TFIDF} = \text{TF} \times \text{IDF}$$

As you can see at the equation above, the Tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

That's why we decided to compare how the words in the main dataset behave, what words are the most frequent for the Count - or Tfidf-Vectorizer and how the n-grams influence the vectorizer that was used.

An N-gram is a sequence of N words: a 1-gram is just a single word (uni-gram) and 2-gram (or bi-gram) is a two-word sequence of words like "peanut butter" or "mineral water". N-grams of texts are extensively used in text mining and natural language processing tasks. We assumed that n-grams and the probabilities of the occurrences of certain words in certain sequences could improve the predictions. That's why in the next process - parameter optimization - unigrams and bigrams were considered as the hyperparameters to be tuned (Naive Bayes, Linear SVC models).

To examine the uni- and bi-grams of our dataset we reached for the Wordcloud package support. With a script written in Python we are able to visualize what words (uni-grams or bi-grams) happen to be the most relevant for the vectorizer in the aspect of the occurrence frequency, example below:

```
wordcloud = WordCloud(stopwords=stop_words,mode="RGBA",
background_color="white", colormap="Blues", width=1000, height=600,
max_words=100).generate_from_frequencies(dict_of_frequencies)
plt.imshow(wordcloud, interpolation='bilinear')
```

Wordclouds generated for each vectorizer are presented below. Each wordcloud includes 100 most frequent words.

Uni-grams for the Countvectorizer:

As we can see above, in the aspect of simple word frequency the words (uni-grams) “smaku”, “napój”, “ser” or “100” appear to be in the Top 3 - the higher frequency the bigger size of the font in the wordcloud. As we can see not only words but numbers can give us crucial information about the dataset. It is an effect of the next process which is more described in the point 2.2.2.

Bigrams for the Countvectorizer:

In bi-grams case, the situation looks quite different. There are words like “woda mineralna”, “500 ml” or “herbata czarna”. The more bi-grams in the vectorizer the higher probability of accurate category prediction is. However, we must remember that this is not the set of most informative terms for the classifier.

Combination of uni- and bigrams for the Countvectorizer:

Countvectorizer as well as TfidfVectorizer offer a possibility to use a combination of uni-grams and bi-grams. As we can see the most frequent n-grams are uni-grams.

Unigrams for the TfidfVectorizer:

In the aspect of Tf-idf frequency the words (uni-grams) like: “żelatyna”, “śliwka”, “włoszczyzna” appear to be the most informative. It’s nearly impossible to find the CountVectorizer uni-grams in this wordcloud. That's why we assumed that the occurrence of such words is more relevant for the classifier in making accurate predictions.

Bi-grams for the TfidfVectorizer:

We also made an analysis of the bi-grams in the TfidfVectorizer. Ones with the word "żurawina" happen to have the highest Tf-idf weights. Next we undertook a step into an analysis of how uni-grams and bi-grams of TfidfVectorizer relate to each other and which one could be more informative for our models.

<u>Combination of uni- and bi-grams for the TfidfVectorizer:</u>

TfidfVectorizer seems to treat uni-grams as more relevant in the aspect of Tf-idf frequency. Nevertheless, there are several bi-grams visible in the wordcloud above: "czosnek 200", "rodzynki 300", "chipsy solone", "knorr curry". This fact confirms the results of the parameter optimization process, where n_gram hyperparameter was tuned to '(1,2)' value which means that the best parameters for the model can be achieved with the combination of uni-grams and bi-grams.

2.2.2 Normalization

Some of the normalization is conducted by the vectorizer (CountVectorizer or TfidVectorizer). All the words are lowercase. The punctuation and numbers should be removed (however in our situation some numbers and punctuations are essential and we added manually regular expressions that do not allow the vectorizer to remove all the numbers and points, commas or percent symbols. The details are described in the point 2.2.4). We do not provide any extra normalization (like normalization of abbreviations and acronyms or conversion of numeric expressions and verbal-numeric expressions to verbal form).

2.2.3 Stop Words

Stop words are the words that are filtered out before natural language processing. The words that can be eliminated are usually prepositions, pronouns and conjunctions. Every language has its own stopword list. In our project we used polish stopword list and also added the names of the on-line shops that we used to web scrape data. They do not bring any valuable information for the algorithms.

It is possible to add the list of stopwords to the vectorizer in sci-kit learn Python library. In our algorithms we experimented both with and without the stopword list and the difference was not statistically significant. Finally in some models we used a vectorizer with a stopword list, in others without it.

2.2.4 Regular expressions written manually (regex)

We had to add the manually written regex to add percent values to the vector of words (it is important for us, because there are two different categories: fat milk and skim milk and we need the information about the percentage of fat to distinguish both of them). 3.2% of fat means that this is fat milk, 2.0% or 0.5% of fat means that it is skim milk. In order to add percentages to the list of analysed words we can add to our vectorizer the following rule:

```
token_pattern='\w\w+|[1-9]\.[1-9]\%|[1-9]\,[1-9]\%|[1-9]\.[1-9]|[1-9]\,[1-9]|[1-9]\%'
```

2.3 Feature Selection

We do not have any feature selection. We have only two columns in our data set: the name of the product and its category. In some cases we have information about the price and unit but we decided this will not be useful for us.

If it is possible, we would like to have the information about the category given by the shop. It could help to distinguish some products that have the same names and different categories (for example the frozen strawberries and the fresh ones).

2.4 Output data

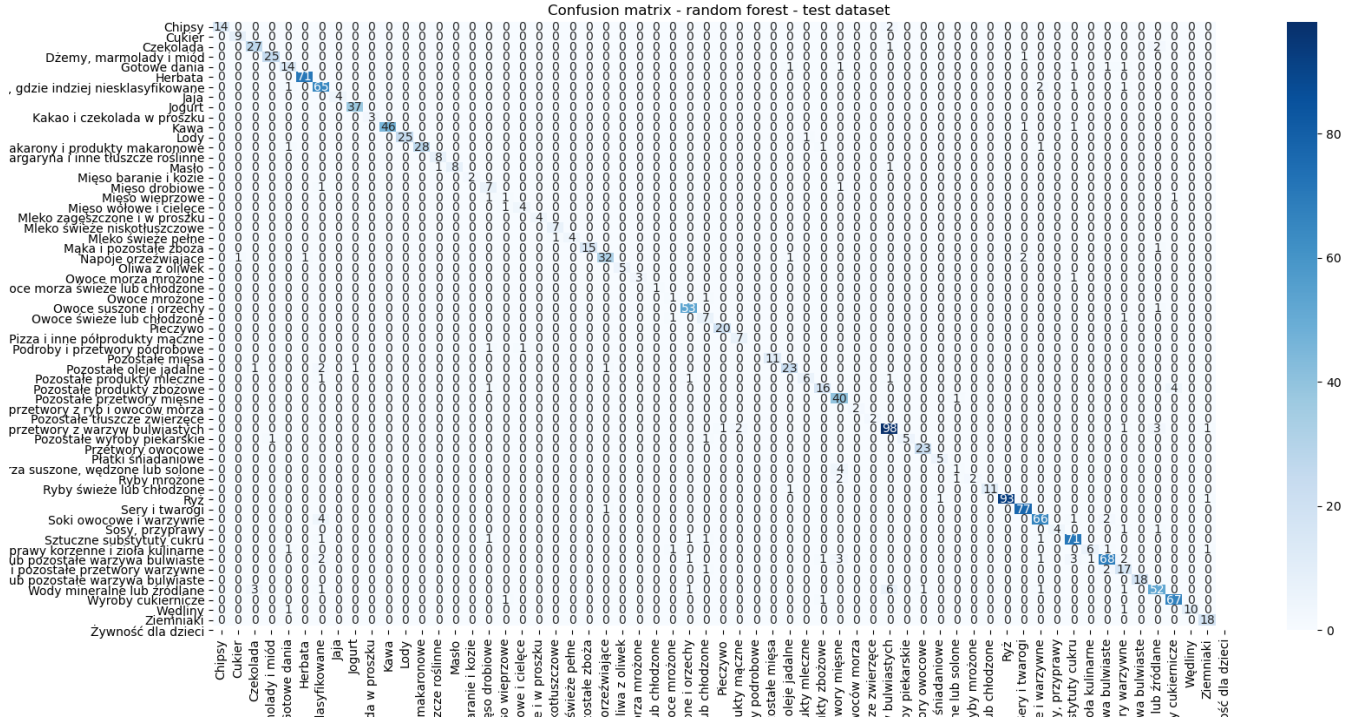
We want our classifier based on a machine learning algorithm to predict the category for every input data given. In fact we are searching for the category with the highest probability for each product name. We are saving the best results for the testing set of data to the file where we can compare the correct category with the predicted one. We are also saving the probability of this category.

In the table below there is one product with the wrong category. It is “ZPC-Nord - Mieszanka cukierków trufli”. It should be in the category “Confectionery products” but the algorithm put it in the category of “Dried fruits and nuts” with the probability of 28%. It is a rather often situation that the cases with the wrong category have low probability. We believe that it is possible to use it to choose the products for manual classification. Anytime we want, we can check the probabilities for all possible product categories (from the code level).

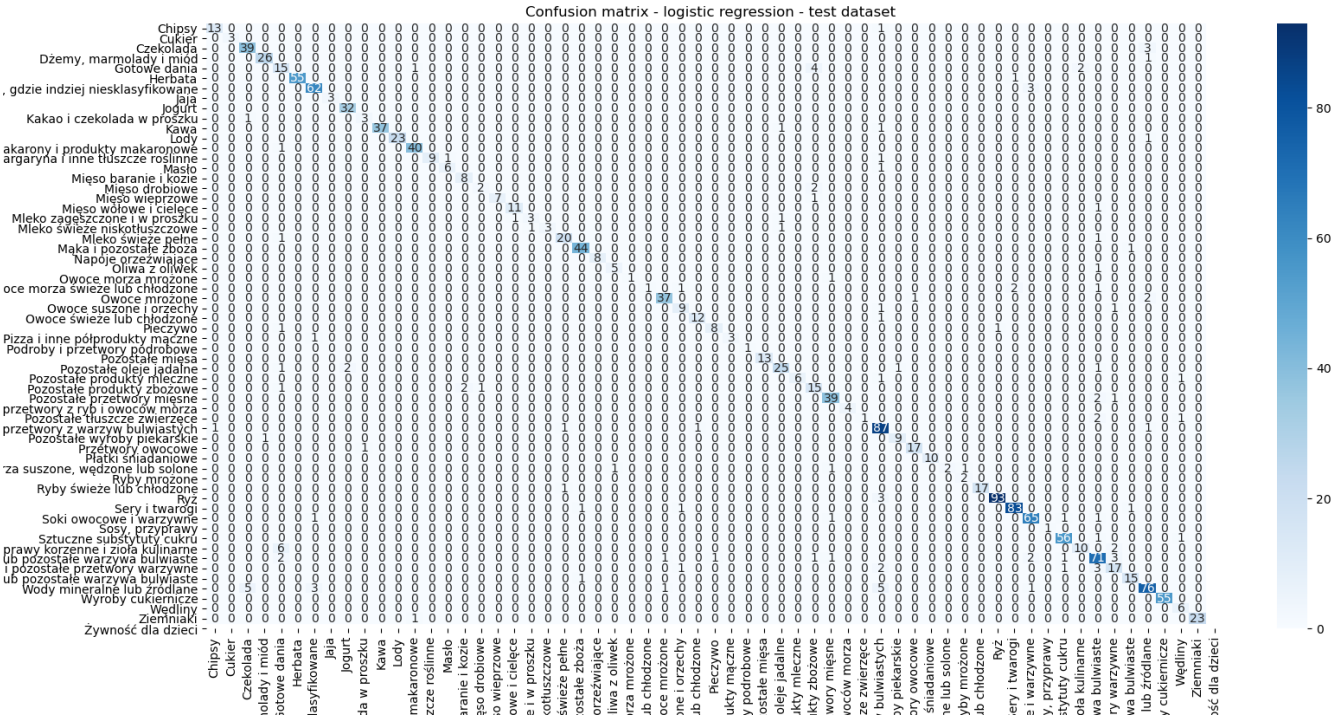
produkt	kategoria	Autocode	Probability
Filety z mintaja mrożony	Ryby mrożone	Ryby mrożone	0,884135741
Mlekpol Ser Salami w plastrach 150 g	Sery i twarogi	Sery i twarogi	0,873373161
Lipton - Yuzu Lime niegazowany napój herbaciany o smaku lim...	Napoje orzeźwiające	Napoje orzeźwiające	0,864711151
Przyprawa Kotanyi kminek cały	Sól, przyprawy korzenne i zioła kulinarne	Sól, przyprawy korzenne i zioła kulinarne	0,940530488
Candia Candy'Up Napój mleczny o smaku czekoladowym 1 l	Pozostałe produkty mleczne	Pozostałe produkty mleczne	0,786012128
Pieprz cytrynowy kamis	Sól, przyprawy korzenne i zioła kulinarne	Sól, przyprawy korzenne i zioła kulinarne	0,877982337
Tassimo Morning Café Kawa mielona 124,8 g (16 kapsułek)	Kawa	Kawa	0,948326129
Pstrąg w galarecie	Pozostałe przetwory z ryb i owoców morza	Pozostałe przetwory z ryb i owoców morza	0,368226558
Danone - Fantasia jogurt kremowy z wiśniami	Jogurt	Jogurt	0,950650398
Gruszka zielona Konferencja	Owoce świeże lub chłodzone	Owoce świeże lub chłodzone	0,225478165
Pieprz zielony Kamis	Sól, przyprawy korzenne i zioła kulinarne	Sól, przyprawy korzenne i zioła kulinarne	0,893113446
Lipton - Herbata Earl Grey Tea 100 szt.	Herbata	Herbata	0,957400435
Owocowy Raj - Sok pomarańczowy 100% tłoczony	Soki owocowe i warzywne	Soki owocowe i warzywne	0,948813989
Twaróg Jana półtłusty	Sery i twarogi	Sery i twarogi	0,874891547
ZPC Nord - Mieszanka cukierków "trufli"	Wyroby cukiernicze	Owoce suszone i orzechy	0,284235909
Saga Herbatka owocowa o smaku owoce leśne 34 g (20 torebek)	Herbata	Herbata	0,963376685
Pasztet z suszonymi pomidorami vege	Pozostałe przetwory mięsne	Pozostałe przetwory mięsne	0,78010054

The confusion matrices below present how accurate our predictions are comparing to test data categories. On the diagonal there are correctly assigned values. All others are the wrong ones.

Confusion matrix – random forest:



Confusion matrix – logistic regression:



3. Machine Learning Solution

3.1 Models tried:

3.1.1 Naive Bayes (MultinomialNB)

This method implements the Naive Bayes algorithm for multinomially distributed data, and is one of the two classic Naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice).

3.1.2 Logistic Regression

Logistic Regression is a linear model for classification rather than regression. It is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In a logistic regression model the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

3.1.3 Random Forest

Here, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max_features. The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

3.1.4 SVM (Linear SVC)

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. LinearSVC is a faster implementation of Support Vector Classification for the case of a linear kernel.

3.1.5 Neural networks (Ludwig Library)

Ludwig is a toolbox built on top of TensorFlow that allows users to train and test deep learning models without the need to write code. It provides a set of model architectures that can be combined together to create an end-to-end model for a given use case.

In our case we had some problems with installing Ludwig library on the computer in our statistical office. We tested this algorithm on the private computer with a linux system and the result (with default parameters) was not better than all algorithms mentioned above that we could find in the sci-kit learn library. We decided not to experiment more with neural networks because of the hardware and software restrictions.

3.2 Model(s) finally selected and the criterion

Naive Bayes, Logistic Regression, Random Forest, SVM – in all those methods the accuracy was above 90%. The parameters were chosen by gridsearch or manually written selection rules (the process of selection can be find in our files on the Statistics Poland github account):

Naive Bayes:

- Suggested parameters: alpha = 0.5, fit_prior = False

SVM (LinearSVC):

- Suggested parameters: C = 1, loss = ‘hinge’, multi_class = ’crammer_singer’, penalty='l1', dual=False, max_iter=1200

Logistic Regression

- Suggested parameters: C=2, fit_intercept=True, class_weight='None', solver='lbfgs', multi_class='ovr', max_iter=200

Random Forest

- Suggested parameters: n_estimators=200, criterion='gini', min_samples_leaf=1, min_samples_split=3, max_features='log2', bootstrap=False, oob_score=False, warm_start=False, class_weight=None='ovr', max_iter=200

3.3 Hardware used

Our personal computers at the statistical office.

3.4 Runtime to train the model

We do not have such data.

4. Results

If we want to analyze our results we have to calculate the basic rates from a confusion matrix.

CONFUSUSION MATRIX	ACTUAL	
PREDICTED	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

$$\text{F1-Score} = \frac{2*Precision*Recall}{Precision+Recall} \qquad \text{Recall} = \frac{TP}{TP+FN}$$

source: cubalytictalks.blogspot.com

We also calculate the Matthews correlation coefficient. The coefficient takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient between the observed and predicted binary classifications. It returns a value between −1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and −1 indicates total disagreement between prediction and observation.

$$\mathbf{MCC} = \frac{\boldsymbol{TP} \times \boldsymbol{TN} - \boldsymbol{FP} \times \boldsymbol{FN}}{\sqrt{(\boldsymbol{TP}+\boldsymbol{FP})(\boldsymbol{TP}+\boldsymbol{FN})(\boldsymbol{TN}+\boldsymbol{FP})(\boldsymbol{TN}+\boldsymbol{FN})}}$$

In our project we decided to generate the classification report and the confusion matrix (some of them were shown already in the point 2.4) for each model.

Classification report (logistic regression, testing set):

Category	precision	recall	F1-score	support
Chipsy	0,93	0,93	0,93	14
Cukier	1	1	1	3
Czekolada	0,87	0,93	0,9	42
Dżemy, marmolady i miód	0,96	0,96	0,96	27
Gotowe dania	0,54	0,68	0,6	22
Herbata	1	0,98	0,99	56
Inne artykuły żywnościowe, gdzie indziej niesklasyfikowane	0,93	0,95	0,94	65
Jaja	1	1	1	3
Jogurt	0,94	1	0,97	32
Kakao i czekolada w proszku	0,75	0,75	0,75	4
Kawa	1	0,95	0,97	39
Lody	1	0,96	0,98	24
Makarony i produkty makaronowe	0,95	0,98	0,96	41
Margaryna i inne tłuszcze roślinne	1	0,82	0,9	11
Masło	0,86	0,86	0,86	7
Mięso drobiowe	0,8	1	0,89	8
Mięso wieprzowe	0,67	0,5	0,57	4
Mięso wołowe i cielęce	1	0,88	0,93	8

Mleko zagęszczone i w proszku	0,92	0,92	0,92	12
Mleko świeże niskotłuszczowe	0,75	0,6	0,67	5
Mleko świeże pełne	1	0,6	0,75	5
Mąka i pozostałe zboża	0,91	0,91	0,91	22
Napoje orzeźwiające	0,96	0,98	0,97	45
Oliwa z oliwek	1	1	1	8
Owoce morza mrożone	0,83	0,83	0,83	6
Owoce morza świeże lub chłodzone	1	0,5	0,67	2
Owoce mrożone	1	0,2	0,33	5
Owoce suszone i orzechy	0,95	0,93	0,94	40
Owoce świeże lub chłodzone	0,75	0,82	0,78	11
Pieczywo	0,92	0,92	0,92	13
Pizza i inne półprodukty mączne	0,89	0,8	0,84	10
Podroby i przetwory podrobowe	1	0,75	0,86	4
Pozostałe mięsa	1	1	1	1
Pozostałe oleje jadalne	1	1	1	13
Pozostałe produkty mleczne	0,89	0,83	0,86	30
Pozostałe produkty zbożowe	1	0,75	0,86	8
Pozostałe przetwory mięsne	0,65	0,79	0,71	19

Pozostałe przetwory z ryb i owoców morza	0,91	0,93	0,92	42
Pozostałe tłuszcze zwierzęce	1	1	1	4
Pozostałe warzywa bulwiaste i przetwory z warzyw bulwiastych	1	0,25	0,4	4
Pozostałe wyroby piekarskie	0,84	0,96	0,89	91
Przetwory owocowe	0,9	0,9	0,9	10
Płatki śniadaniowe	0,94	0,94	0,94	18
Ryby i owoce morza suszone, wędzone lub solone	1	1	1	10
Ryby mrożone	1	0,4	0,57	5
Ryby świeże lub chłodzone	0,67	1	0,8	2
Ryż	1	0,94	0,97	18
Sery i twarogi	0,99	0,97	0,98	96
Soki owocowe i warzywne	0,97	0,97	0,97	86
Sosy, przyprawy	0,92	0,94	0,93	69
Sztuczne substytuty cukru	1	1	1	1
Sól, przyprawy korzenne i zioła kulinarne	0,95	0,97	0,96	58
Warzywa mrożone inne niż ziemniaki lub pozostałe warzywa bulwiaste	0,83	0,56	0,67	18
Warzywa suszone i pozostałe przetwory warzywne	0,84	0,86	0,85	83
Warzywa świeże lub chłodzone inne niż ziemniaki lub pozostałe warzywa bulwiaste	0,71	0,71	0,71	24
Wody mineralne lub źródlane	0,88	0,94	0,91	16

Wyroby cukiernicze	0,9	0,84	0,87	91
Wędliny	1	1	1	55
Ziemniaki	0,67	1	0,8	6
Żywność dla dzieci	1	0,96	0,98	24
macro avg	**0,91**	**0,85**	**0,87**	**1500**
weighted avg	**0,92**	**0,91**	**0,91**	**1500**
accuracy	**0,91**			

In the table above you can see the precision, recall and F1-score for each category. The support is the number of products in this category in the testing set. As you can see, there are the categories that are perfectly classified (like sugar or eggs) and there are categories with poor classification (like fresh or chilled fish).

We also calculate the accuracy, F1-score and MCC for training, testing and validation set.

	Naive Bayes	**Random Forest**	**Logistic Regression**	**SVM**
training accuracy	0.9837857	0.9990462	0.9831988	0.9963316
training F1-score	0.9837857	0.9990462	0.9831988	0.9963316
training MCC	0.9832226	0.9990134	0.9826198	0.9962037
validation accuracy	0.89466666	0.9320000	0.9226666	0.9100000
validation F1-score	0.89466666	0.9320000	0.9226666	0.9100000
validation MCC	0.89096217	0.9295453	0.9198023	0.9068214
testing accuracy	0.91200000	0.9126666	0.9053333	0.9220000
testing F1-score	0.91200000	0.9126666	0.9053333	0.9220000
testing MCC	0.9089663	0.9094546	0.9018299	0.9193366

We decided also to check if the chosen random seed has an important impact on the accuracy and other results. If the data set is small, the changes in selection to the training, validation and testing set, can show quite big differences. With the set of 100 products the difference in accuracy could be between 60-100%. We wanted to know if with our 17000 products we have the same problem.

The tests were done for the logistic regression model. Below you can see the table with the results:

random seed	1	2	12	17	26	38	43	57	64	77	85	86	91	100	102	111
training data - accuracy	0,9820	0,9823	0,9829	0,9832	0,9817	0,9827	0,9829	0,9825	0,9815	0,9828	0,9820	0,9831	0,9823	0,9822	0,9832	0,9827
training data - F1-score	0,9820	0,9823	0,9829	0,9832	0,9817	0,9827	0,9829	0,9825	0,9815	0,9828	0,9820	0,9831	0,9823	0,9822	0,9832	0,9827
training data - MCC	0,9813	0,9817	0,9823	0,9826	0,9810	0,9821	0,9823	0,9819	0,9809	0,9822	0,9814	0,9825	0,9817	0,9816	0,9826	0,9821
validation data - accuracy	0,9073	0,9227	0,8920	0,9140	0,9133	0,9160	0,9107	0,9053	0,9060	0,9113	0,9220	0,9147	0,9007	0,9120	0,9227	0,9173
validation data - F1-score	0,9073	0,9227	0,8920	0,9140	0,9133	0,9160	0,9107	0,9053	0,9060	0,9113	0,9220	0,9147	0,9007	0,9120	0,9227	0,9173
validation data - MCC	0,9044	0,9199	0,8883	0,9109	0,9103	0,9131	0,9076	0,9020	0,9028	0,9082	0,9193	0,9117	0,8972	0,9090	0,9198	0,9145
test data - accuracy	0,9200	0,9233	0,9180	0,9100	0,9233	0,9040	0,9020	0,9153	0,9060	0,9207	0,9240	0,9013	0,9067	0,9100	0,9053	0,9113
test data - F1-score	0,9200	0,9233	0,9180	0,9100	0,9233	0,9040	0,9020	0,9153	0,9060	0,9207	0,9240	0,9013	0,9067	0,9100	0,9053	0,9113
test data - MCC	0,9171	0,9206	0,9150	0,9069	0,9205	0,9007	0,8986	0,9123	0,9028	0,9180	0,9212	0,8979	0,9033	0,9069	0,9018	0,9082

The results show that the difference is not great. The accuracy for the test data is always about 90-92%.

5. Code/programming language

We are using a Python library called scikit-learn (https://scikit-learn.org/stable/index.html). It is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation and many other utilities.

Our code and all the comments can be found on the github account of Statistics Poland: https://github.com/statisticspoland/ecoicop_classification

6. Evolution of this study inside the organisation

The results were presented not only to the management of our office (Statistical Office in Poznań – Greater Poland voivodeship) but also to the President of the whole Statistics Poland (GUS). It was well received.

During the working meeting for the Big Data Experts from all the statistical offices in Poland we discussed the possibility of using big data and machine learning methods in the

calculation of official statistics. Our example was the proof that the process of classification can be supported by machine learning methods.

We still need more cooperation between all the departments in the Statistics Poland. We should share our experience in order to introduce modern methods more efficiently.

7. Is it a proof of concept or is it already used in production?

This is a proof of concept. With this example we want to convince people that it is possible to use machine learning methods in official statistics. We would like to encourage people to experiment with modern methods.

7.1 What is now doable which was not doable before?

We proved that it is possible to classify products by machine learning methods with over 90% accuracy. We are in the process of preparing the application that implements all those methods and helps users with no knowledge of programming to classify products.

7.2 Is there already a roadmap/service journey available how to implement this?

No, there is not. We do not have knowledge about the current process of classification and it is not possible to prepare any roadmap.

7.3 Who are the stakeholders?

We are open to share our experience in machine learning methods with everybody in our organization that has to classify a lot of data. We shared our code on the github of Statistics Poland. We also had a presentation during the working meeting for the Big Data Experts from all the statistical offices in Poland. We are sure that every team that deals with classification could benefit from our work.

7.4 Fall Back

There is no fall back plan. The machine learning methods have not been used in the products classification so far. Before preparing any fall back plan, we would need to get the knowledge about the current process of classification.

7.5 Robustness

There is no gold standard in our office. We still wait for the set classified and verified by experts. There are also no fail checks so far.

8. Conclusions and lessons learned

ML can be used for product classification at the accuracy level over 90%. The products with very low probability could be classified manually and it could improve the accuracy. We learned a lot about the process of machine learning classification and natural language processing (NLP).

The international collaboration and support from the members of UNECE High Level Group for Modernisation of Official Statistics – Machine Learning Project was definitely the value added.

In the Statistics Poland we need more cooperation between the departments to share our knowledge and experience.

9. Potential organisation risk if ML solution not implemented

There is no potential organisation risk because machine learning solutions have not been used so far and nothing is going to change.

10. Has there been collaboration with other NSIs, universities, etc?

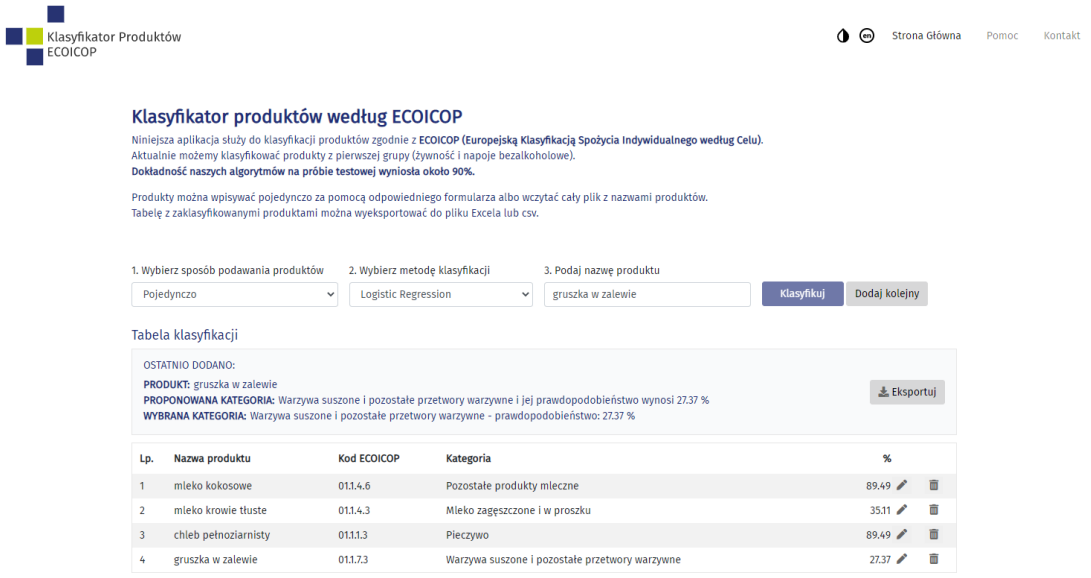
All the work was done by the team of specialists in the Data Engineering Department in the Statistical Office in Poznań. We shared our knowledge during the webex meetings and during the UNECE sprint meeting in Belgrade.

11. Next Steps

We would like to introduce the application which could help in classifying products' names according to ECOCIOP classification. It constitutes the next project value added. This multipurpose tool could be used by anybody who deals with product classification in everyday work. No programming experience required. We are of the opinion that such an application could encourage people to experiment with something new and show them that using machine learning methods can be simple.

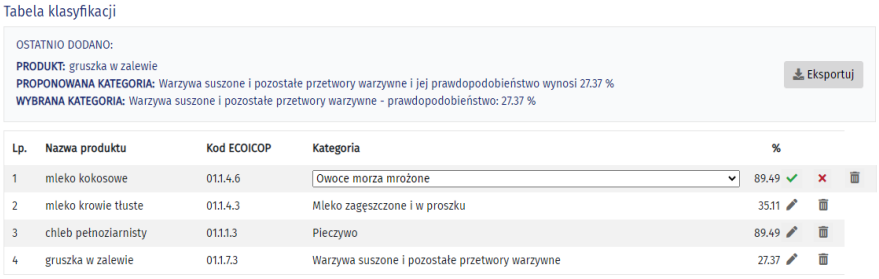
The current version of the application (screens below) has got functionalities like:

- choosing the way of the classification - individually (per one product name) or from the file (list of products' names),
- choosing the classification method - Naive Bayes, LinearSVC, Logistic Regression or Random Forest,
- exporting classification results into the xls or csv file.



- editing the classification results in a table:

- if autocoded category isn't the proper one, user has a possibility to change it and save the results



- deleting table row if it's necessary,
- preliminary file validation - files beyond xls or csv format will not be accepted, the list of products must be included in a particular column.

Considering external comments about application above we want to improve it in the near future in the context of:

- informing user the word entered doesn't occur in the feature matrix and the results might be inadequate,
- preventing entering the words without phonetic sense or curses,
- putting restrictions on the size of the uploaded file.

We are open to cooperate with other departments of Statistics Poland anytime they consider to use such an application. There is always the possibility to adjust this tool to the special needs of the people who work on ECOICOP classifications every day.