Business and Project Management

Project Design
Clickbait Detector

Bensi Simone, Nocchi Tommaso

CONTENTS

1. Introduction

Clickbait is a text or a thumbnail link that is designed to attract attention and to entice users to follow that link and read, view, or listen to the linked piece of online content, being typically deceptive, sensationalized, or otherwise misleading. An advertiser aims to exploit the "curiosity gap", providing just enough information to make readers of news websites curious, but not enough to satisfy their curiosity without clicking through to the linked content.

More in depth, sentiment analysis concepts have been used to highlight some frequent patterns that are ordinarily followed in sensationalistic titles to increase curiosity in readers.

Usually a clickbait is a form of false advertisement, however it can also be used properly to increment the amount of visitors in a web site.

1.1 Possible usages

This tool can be exploited by web browsers and in general by companies in their digital marketing campaign.

1.1.1 Integration with web browsers

A team of researchers from the University of Mississippi and the University of Oklahoma has highlighted that about 25% of websites on the internet are clickbait. Web browsers can use a clickbait detector to automatically filter out from search results titles classified as clickbait, optimizing users research time. This browser functionality would be particularly recommended for people with poor browsing experience who, otherwise, would likely visit unreliable websites.

Clickbait detection has become very useful for web browsers as the number of web pages (and so of clickbait) on the internet is increasing year after year, as shown in the figure below.

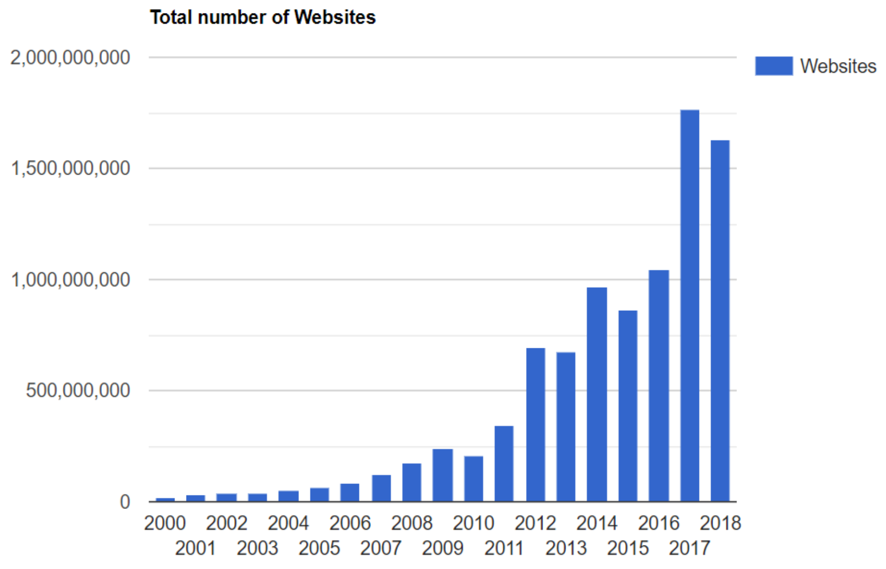


Figure 1: total number of websites over years

Nowadays there are about 2 billion websites on the internet, hence there are about 500 million clickbait sites. Therefore, without any automatic filtering mechanism there is an extremely high likelihood that web browsers would show a list of search results containing some clickbait web pages.

1.1.2 Integration with companies marketing strategy

This application can be exploited by the marketing manager of companies to improve their marketing strategy campaign. In particular, it can be used in the digital marketing scope when performing the Search Engine Optimization (SEO), namely the process which aims to achieve the highest ranking in the organic listing on the search engines results page. Indeed, the position of any page is determined by the algorithm of the search engine, which evaluates websites keywords, titles and content, so it becomes particularly important to elaborate some attractive sentences.

The use of this tool is not restricted in digital marketing, in fact it can be very useful also in traditional marketing. For instance, it can allow to produce some very effective slogans able to describe in an iconic way the company. Other examples could be phrases through which companies perform the promotion or advertising of their products and services, not only on the internet but also on TV and on billboards.

To use this tool, a company marketing team just has to elaborate a text and check if it is engaging. If not, they only have to produce another one and check again, until one of them is valid.

1.2 The Application

After the login or the registration, the application allows users to insert a new title to detect if the given title is a clickbait title or not. To perform this evaluation, the system uses the best model trained during the sentiment analysis phase, carried out through the use of natural language processing (NLP) techniques.

The only thing users need to do is to type the title to check in the command line interface (CLI) and look at the prediction.

Moreover, the application can provide to the user the list of titles already checked during the application usage.

1.3 Requirements

This section describes the requirements that the application must provide.

1.3.1 Functional Requirements

- The application must provide a registration form in order to allow new user to sign in as Standard User
- The application must handle the login process, so that Anonymous Users can identify themselves inserting username and password
- Anonymous User must be denied the opportunity to access the functionalities offered by the application
- Standard Users must be given the possibility to check the type of title typed

- Standard Users must be given the ability to browse the full list of titles checked by him

1.3.2 Non-functional Requirements

- Usability: The application must be user friendly, with a simple e intuitive command line interface

1.4 Use Cases Diagram

The figure below shows the UML use case diagram representing the user's interaction with the application.

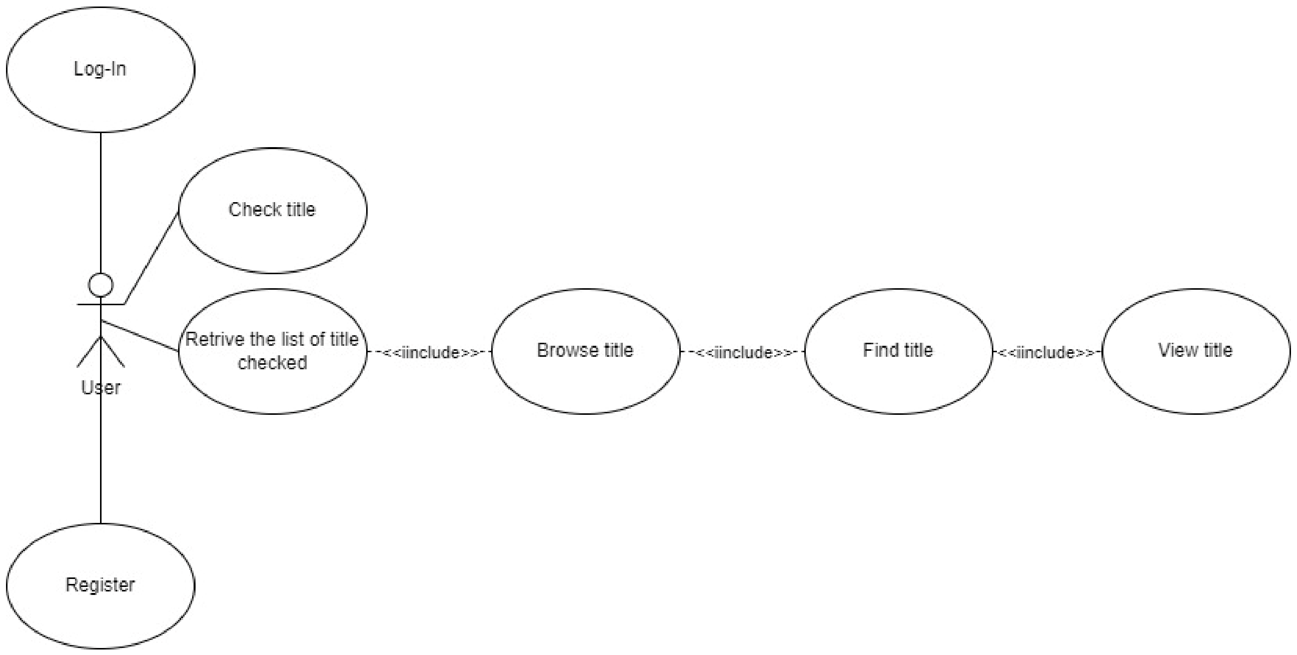


Fig.1: Application use case diagram

1.5 Class Analysis Diagram

The figure below shows the main entities of the application and the relationship between them.

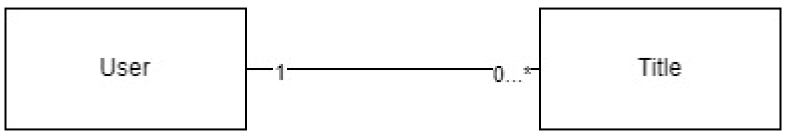


Fig.2: Application class analysis diagram

From this class analysis diagram we can see that users can insert zero or more titles, while a specific title can be inserted from one only user.

1.6 Data Model

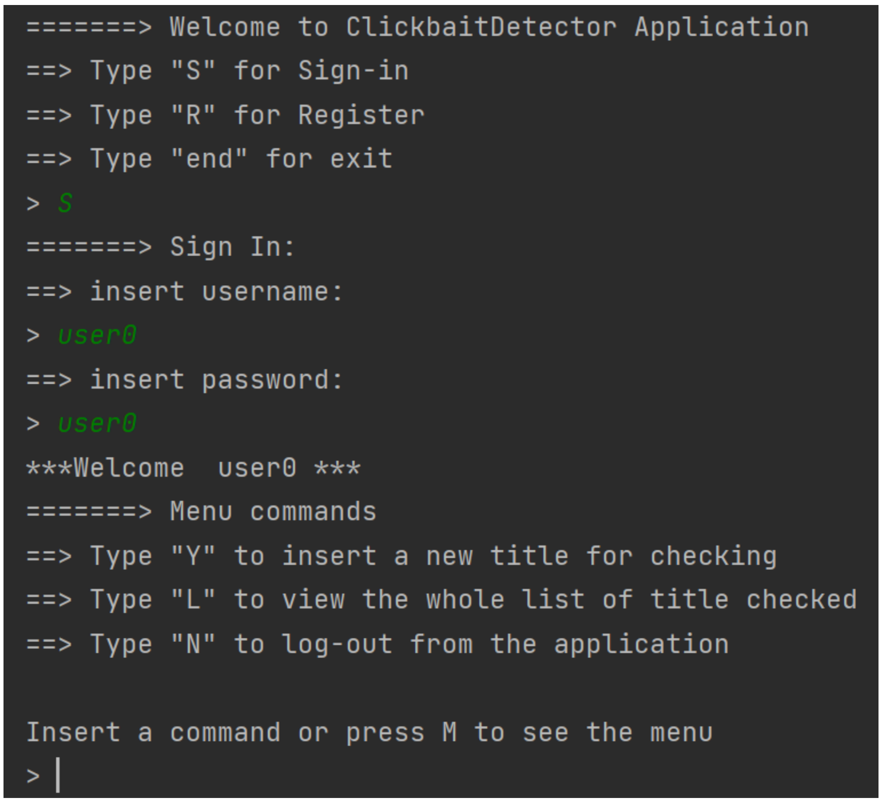
We decided to use MongoDB to store application data. The DB contains one collection, namely the user collection, which has the same structure of the following example:

```
[
    {
        "_id": "user0",
        "password": "user0",
        "personalTitles":
        [
            {
              "title": "These award-winning photos show planet Earth in ways you've never seen it",
              "prediction": "clickbait",
              "timestamp": {"$date": "2022-01-24T22:47:59.731Z"}
            },
            {
              "title": "US attorney conducting criminal investigation into Fox News ",
              "prediction": "non_clickbait",
              "timestamp": {"$date": "2022-01-24T22:48:17.728Z"}
            }
        ]
    },
    {
        "_id": "user1",
        "password": "user1",
        "personalTitles":
        [
            {
              "title": "Here’s what you should know about the newfound TRAPPIST-1 solar system",
              "prediction": "clickbait",
              "timestamp": {"$date": "2022-01-24T20:12:17.728Z"}
            },
            {
              "title": "After Fukushima: the tenants rebuilding a destroyed community",
              "prediction": "non_clickbait",
              "timestamp": {"$date": "2022-01-22T21:08:11.728Z"}
            },
            {
              "title": "Chealsea wins the Premier League",
              "prediction": "non_clickbait",
              "timestamp": {"$date": "2022-01-24T22:48:17.223Z"}
            }
        ]
    }
]
```

Fig.3: MongoDB collection example

1.7 Implementation

The implementation of the application has been written in Python, using scikit-learn functionalities which allow to exploit NLP techniques. When the application starts it requires the user to register or to login himself.

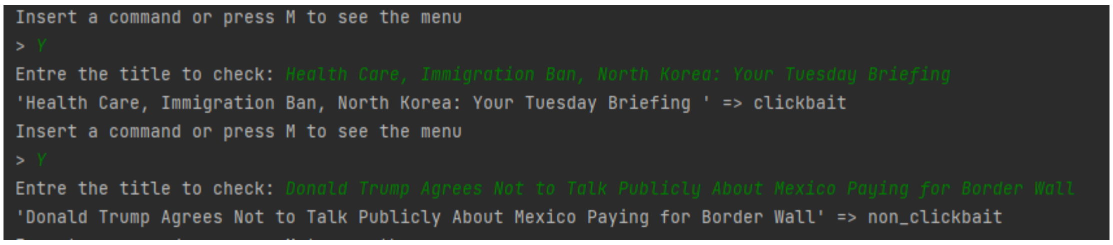


```
=======> Welcome to ClickbaitDetector Application
==> Type "S" for Sign-in
==> Type "R" for Register
==> Type "end" for exit
> S
=======> Sign In:
==> insert username:
> user0
==> insert password:
> user0
***Welcome  user0 ***
=======> Menu commands
==> Type "Y" to insert a new title for checking
==> Type "L" to view the whole list of title checked
==> Type "N" to log-out from the application

Insert a command or press M to see the menu
> |
```

Fig.4: Registration/login phase

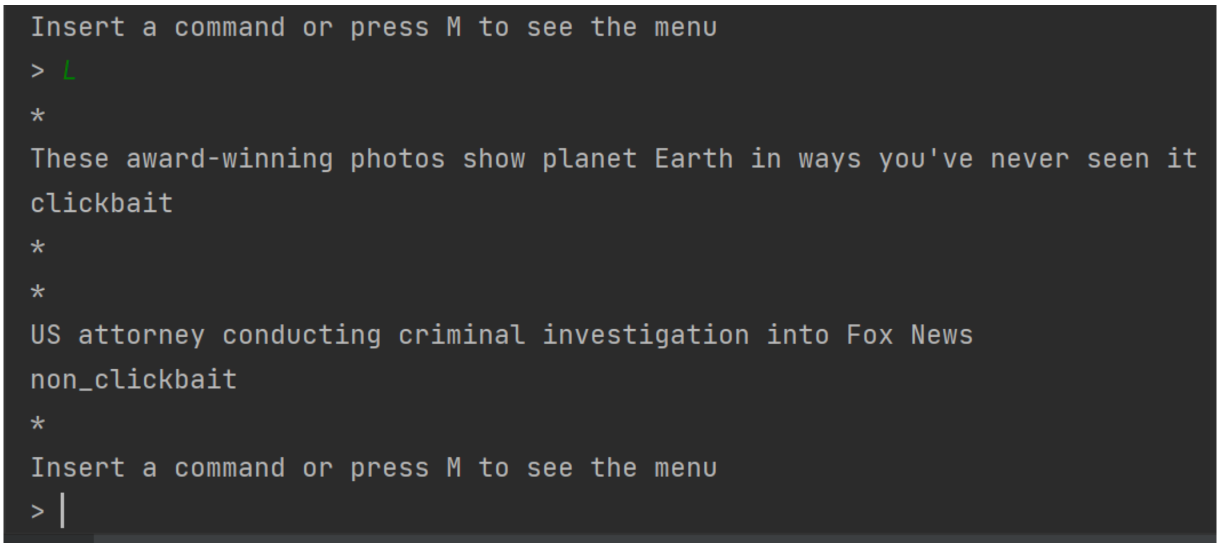
Then the user can insert a new title to check whether the title is a clickbait title or not by inserting “Y” as command in the terminal.



```
Insert a command or press M to see the menu
> Y
Entre the title to check: Health Care, Immigration Ban, North Korea: Your Tuesday Briefing
'Health Care, Immigration Ban, North Korea: Your Tuesday Briefing ' => clickbait
Insert a command or press M to see the menu
> Y
Entre the title to check: Donald Trump Agrees Not to Talk Publicly About Mexico Paying for Border Wall
'Donald Trump Agrees Not to Talk Publicly About Mexico Paying for Border Wall' => non_clickbait
```

Fig.5: Text checking

Furthermore, the user can see the list of all previously checked titles inserting “L” as command in the terminal.



```
Insert a command or press M to see the menu
> L
*
These award-winning photos show planet Earth in ways you've never seen it
clickbait
*
*
US attorney conducting criminal investigation into Fox News
non_clickbait
*
Insert a command or press M to see the menu
> |
```

Fig.6: List of user previously checked texts

Users can leave the application at any time by inserting “N” as command in the terminal.

2. Data Management

In this section it is explained how we managed row data and how we built a classifier used by the application to produce the output for each title.

2.1 Dataset

We found a dataset containing about 32000 titles, which occupies around 1.84 MB. This dataset is balanced, indeed 16000 titles are labeled as clickbait and 16000 as non-clickbait.

Some typical clickbait titles examples found in the dataset:

- Which Candy Is The Best
- Millennials At Work: Expectations Vs. Reality
- Does Coffee Make You Poop
- Who Is Your Celebrity Ex Based On Your Zodiac
- 17 Hairdresser Struggles Every Black Girl Knows To Be True
- Are You More Walter White Or Heisenberg
- The Most Canadian Groom Ever Left His Wedding To Plow Out His Guests In A Snow Storm
- Here's One Really Weird Thing About Butterfree
- 15 Resolutions To Make Good On In 2016
- What New Thing Should You Try In 2016
- Zoo Animals Around The World Are Opening Their Christmas Presents Early
- Tell Us About Yourself: Erica Ash
- 9 Times I Cried
- This Goat Has Been Bullying His Tiger Friend

Some typical clickbait titles examples found in the dataset:

- U.S. Helps Palestinians Build Force for Security
- Airbus wins Qatar Airways order worth $15bn
- Russian novelist Aleksandr Solzhenitsyn dies aged 89
- Hillary Clinton wins Pennsylvania, cuts Obama's lead
- Afghani heroin shipment shot down in Tajikistan
- Court in France convicts Scientology of organized fraud
- Resolution on Obama Stirs Debate
- At least eight dead after bombings in Iraq
- Soldiers question U.S. Defense Secretary on issues during Kuwait visit
- Medical School Says Ex-Army Surgeon, Hid Medtronic Ties
- Rapper Kanye West denounces Bush response, American media at hurricane relief telethon
- Coordinated terrorist attack hits London
- Botswana holds parliamentary elections
- US government stops Haiti evacuations

This dataset is available at:
https://www.kaggle.com/amananandrai/clickbait-dataset

2.2 Text Classification Process

Text classification is an NLP technique that assigns a set of predefined categories to open-ended text. In particular, the text classification process has to assign to a title a label which can be “clickbait” or “non-clickbait”. To do this, we followed the steps shown in the figure below.

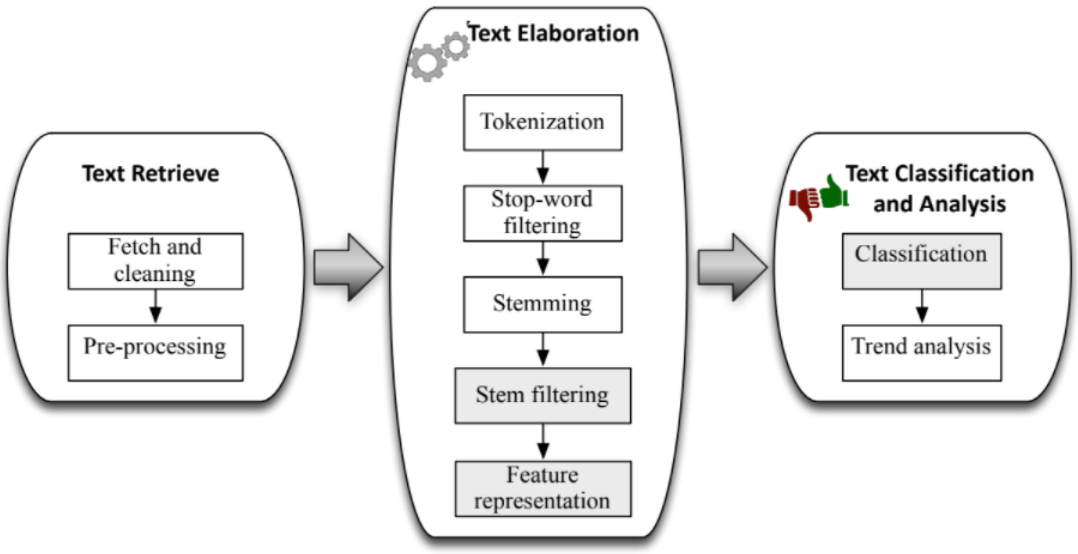


Fig.7: Text classification process

2.2.1 Data Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct.

In the dataset we found, we had to check if there were missing, inconsistent or duplicated values. Moreover, we converted all text in lower-case, and we removed punctuation marks and useless meta-information, such as links, hashtags, timestamp and emoticons. An example of text cleaning that we made is shown below.

Original title: *"Iraq on verge of civil war, head of Arab league fears"*

Cleaned title: *iraq on verge of civil war head of arab league fears*

2.2.2 Tokenization

Tokenization consists in transforming a stream of characters into a stream of processing units called tokens, namely words. In this way, each title is represented as a set of words, according to the BoW (Bag of Words) representation. An example of tokenization that we made is shown below.

Cleaned title: *iraq on verge of civil war head of arab league fears*

Title tokens: *<iraq>, <on>, <verge>, <of>, <civil>, <war>, <head>, <of>, <arab>, <league>, <fears>*

2.2.3 Stop-word Filtering

Stop words are any words in a list called stop list which are filtered out before processing text data. By removing these words, we remove the low-level information from text in order to give more focus to the important information.

There is no single universal list of stop words used by all NLP tools, indeed any group of words can be chosen as the stop words for a given purpose. However, there are some established stop word lists. Common stop-words include articles, conjunctions, prepositions and pronouns. Some stop word lists come out from NLP research work and some are just manually curated by different people. We found an English stop-word list containing 319 words. However, some of these stop words can be useful for our prediction, such as in classifying writing style. For this reason we decided to create a customized stop-words list in order to not exclude some relevant words for our purpose. This stop-words list contains the following 239 words:

```
['a', 'about', 'above', 'across', 'afterwards', 'alone', 'along', 'already', 'also', 'although', 'always', 'am',
'among', 'amongst', 'amoungst', 'amount', 'an', 'and', 'another', 'any', 'anyhow', 'anyway', 'are',
'around', 'as', 'at', 'back', 'be', 'became', 'because', 'become', 'becomes', 'becoming', 'been',
'beforehand', 'behind', 'being', 'below', 'beside', 'besides', 'between', 'beyond', 'bill', 'both', 'bottom',
'by', 'call', 'co', 'con', 'could', 'couldnt', 'de', 'describe', 'do', 'done', 'down', 'due', 'during', 'each', 'eg',
'eight', 'either', 'eleven', 'else', 'elsewhere', 'etc', 'even', 'except', 'few', 'fifteen', 'fify', 'fill', 'find', 'five',
'for', 'former', 'formerly', 'forty', 'found', 'four', 'from', 'front', 'further', 'get', 'give', 'go', 'had', 'has',
'hasnt', 'have', 'he', 'hence', 'her', 'here', 'hereafter', 'hereby', 'herein', 'hereupon', 'hers', 'herself',
'him', 'himself', 'his', 'hundred', 'i', 'ie', 'in', 'inc', 'interest', 'into', 'is', 'it', 'its', 'itself', 'keep', 'latter',
'latterly', 'less', 'ltd', 'made', 'many', 'me', 'meanwhile', 'mill', 'mine', 'more', 'moreover', 'mostly',
'move', 'name', 'namely', 'neither', 'nevertheless', 'next', 'nine', 'no', 'nor', 'not', 'of', 'off', 'often', 'on',
'once', 'one', 'only', 'onto', 'or', 'other', 'others', 'otherwise', 'our', 'ours', 'ourselves', 'out', 'over', 'own',
'part', 'per', 'perhaps', 'put', 're', 'see', 'serious', 'she', 'show', 'side', 'since', 'sincere', 'six', 'sixty', 'so',
'some', 'such', 'take', 'ten', 'than', 'that', 'the', 'their', 'them', 'themselves', 'then', 'thence', 'there',
'thereafter', 'thereby', 'therefore', 'therein', 'thereupon', 'these', 'they', 'thick', 'thin', 'third', 'this',
'those', 'though', 'three', 'through', 'throughout', 'thru', 'thus', 'to', 'together', 'too', 'top', 'toward',
'towards', 'twelve', 'twenty', 'two', 'un', 'under', 'up', 'upon', 'us', 'very', 'via', 'was', 'we', 'well', 'were',
'whence', 'which', 'while', 'whither', 'who', 'whoever', 'whole', 'whom', 'whose', 'why', 'will', 'with',
'within', 'without', 'yet', 'yourself', 'yourselves']
```

At the end of this step, each title is cleaned from stop-words, and thus reduced to a sequence of relevant tokens. An example of stop-word filtering that we made is shown below.

Title tokens: *<iraq>, <on>, <verge>, <of>, <civil>, <war>, <head>, <of>, <arab>, <league>, <fears>*

Filtered title tokens: *<iraq>, <verge>, <civil>, <war>, <head>, <arab>, <league>, <fears>*

2.2.4 Stemming

Stemming is the process of reducing each token to its stem or root form, by removing its suffix, in order to group words having closely related semantics. Hence, at the end of this step each text is represented as a sequence of stems. An example of stemming that we made is shown below.

Filtered title tokens: *<drug>, <resistant>, <infections>, <rise>, <hospitals>*

Stemmed title tokens: *<drug>, <resist>, <infect>, <ris>, <hospital>*

2.2.5 Stem Filtering

During this stage, the number of stems of each text are reduced, by removing noisy stems and maintaining only the most relevant ones. Thus, each text is cleaned from stems not belonging to the set of relevant stems. The set of relevant stems can be provided as a vocabulary or identified during a supervised learning stage, using the corpus of training documents.

2.2.6 N-gram Vectorization

The N-grams vectorization method counts combinations of adjacent words of length N in the title. We decided to test 1-gram vectorization and 2-gram vectorization because we observed in several titles similar sentence construction. Therefore, we needed to compare these two N-gram vectorization methods to highlight some relevant patterns to recognize clickbait titles.

From this analysis we found the 150 most frequent 1-gram and 2-gram tokens, having the following frequency distribution.

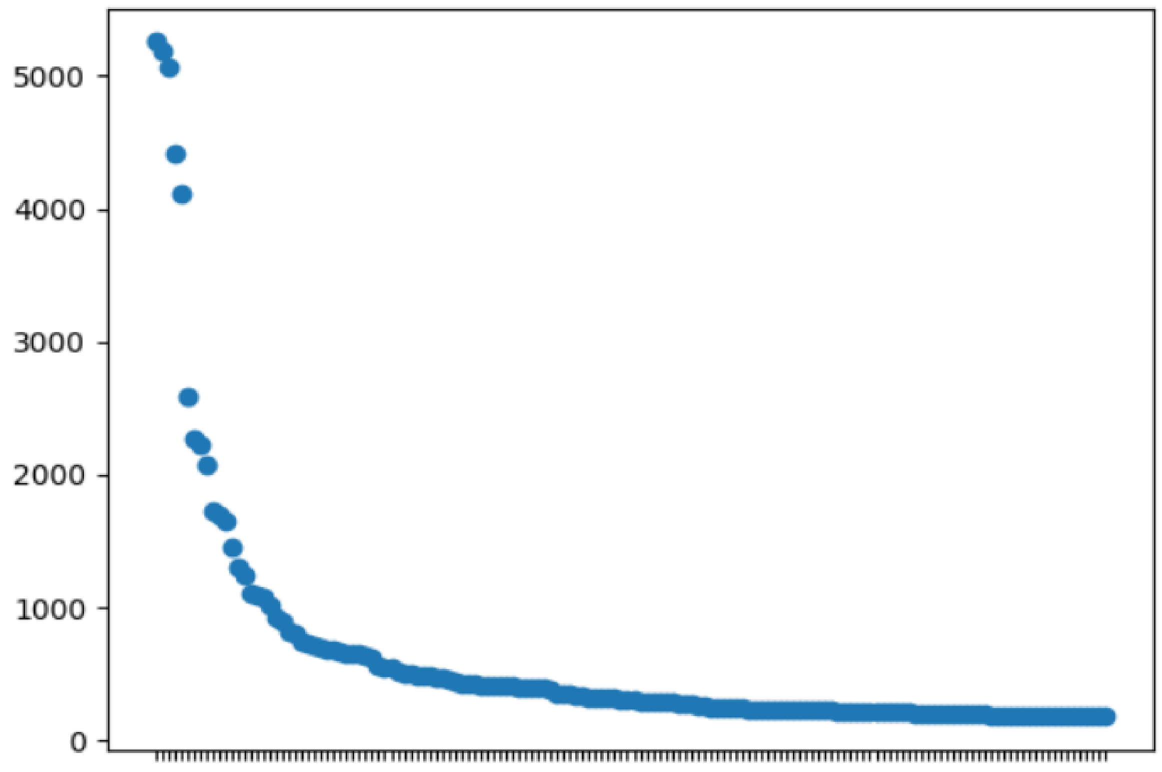


Fig.8: 1-gram vectorization

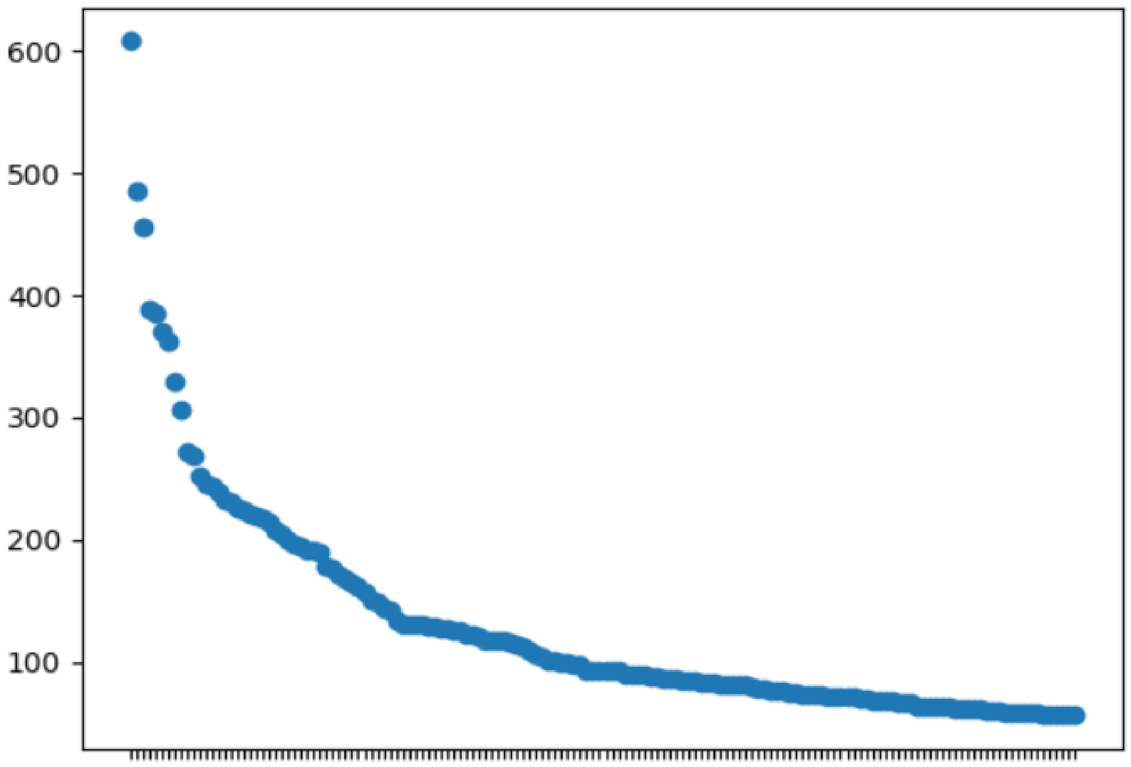


Fig.9: 2-gram vectorization

In these graphics, on the x-axis are reported the tokens and on the y-axis are reported the number of tokens occurrences in the dataset.

2.2.7 Feature Selection

When the input data to an algorithm is too large to be processed and it is suspected to be redundant, then it can be transformed into a reduced set of features. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. It yields better results than applying machine learning directly to the raw data.

In particular, feature selection is a process which selects those features in the dataset that contribute most to the prediction variable or output in which we are interested. The benefits of performing feature selection before modeling your data are:

- Overfitting avoidance: less redundant data gives performance boost to the model and results in less opportunity to make decisions based on noise
- Training time reduction: less data means that algorithms train faster

For these reasons we decided first to select at most 150 features, and then we filtered them by using chi-square value.

2.2.7.1 Chi-square Test

We decided to use the Chi-Square feature selection in order to test the independence of features. More specifically we used it to test whether the occurrence of a specific term and the occurrence of a specific class are independent. We computed a rank for each term in the dataset accordingly with their chi-value as follows:

$$\chi^2(D,t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

where:

- N is the observed frequency in and E the expected frequency
- e_t takes the value 1 if the dataset contains term t and 0 otherwise
- e_c takes the value 1 if the dataset is in class c and 0 otherwise
- D is the dataset

For each feature (term), a corresponding high χ^2 score indicates that the null hypothesis of independence should be rejected and the occurrence of the term and class are dependent. In this case, we should select the feature for the text classification.

Then we decided to order features by their rank and to select the 50th percentile of this list, because these terms are considered to be more relevant to the task than the others. We thought to compare configurations with and without this chi-square test selection in order to understand which of these configurations performs better.

For the Chi-Square feature selection we should expect that out of the total selected features, a small part of them are still independent from the class. In text classification, however, it rarely matters when a few additional terms are included in the final feature set. All is good as long as the feature selection is ranking features with respect to their usefulness and is not used to make statements about statistical dependence or independence of variables.

2.2.8 Feature Representation

We needed to perform some feature extraction in order to transform raw data into numerical features that can be processed while preserving the information in the original data set. In particular, we decided to adopt a feature representation named BoW, which consists in building for each text the corresponding vector of numeric features in order to represent all the texts in the same feature space.

A summary of our dataset BoW feature representation is the following:

```
{'hood': 8754, 'texas': 18319, 'shoot': 16510, 'question': 14515, 'ask': 1365, 'before': 1979, 'jersey':
9735, 'legal': 10494, 'medic': 11423, 'microsoft': 11608, 'acquire': 558, 'firm': 6979, '32': 232, 'movie':
12026, 'excite': 6446, 'toronto': 18624, 'film': 6922, 'festival': 6856, 'region': 14953, 'elect': 6025, 'win':
20025, 'support': 17836, 'opponent': 12806, 'ban': 1781, 'keep': 9984, 'animal': 1070, 'forc': 7175, 'cat':
3194, 'inhale': 9289, 'americ': 981, 'hiker': 8608, 'arrest': 1307, 'iran': 9541, 'girl': 7743, 'cold': 3816,
'cute': 4667, 'dad': 4703, 'gold': 7839, 'glob': 7790, 'message': 11547, 'emo': 6118, 'kid': 10051,
'israeli': 9591,...}
```

In information retrieval the term frequency–inverse document frequency (TF-IDF) is a numerical statistic that is intended to reflect how important a word is to a document in a collection. We used it as a weighting factor during text mining. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

In particular, given the set of Q stems extracted from the training texts:

$$CS_{tr} = \{s, \dots, s_q, \dots, s_Q\}$$

We computed the importance of each stem s_q in the set of training texts by means of a weight w_q, computed as:

$$w_q = \ln \frac{N_{tr}}{N_q}$$

where N_{tr} is the number of labeled texts and N_q is the number of texts containing stem s_q.

We removed all the stems which have a weight equal to zero and represented each training text by using the features associated to the relevant stems. We computed the TF-IDF value for each feature of each text in the following way:

$$w_{q,j} = Tf_{q,l} \cdot w_q$$

where $Tf_{q,l}$ is the frequency of s_q in text j.

A partial list of computed TF-IDF values for the words of our dataset is shown below:

(Title number, word index, TF-IDF value)

```
(0, 14189) 0.4837246770272478
(0, 14169) 0.5153009697574068
(0, 12748) 0.40623935221085267
(0, 12324) 0.254824561949247
(0, 5528) 0.5201051612363043
(1, 13067) 0.4189189959768637
(1, 11710) 0.4015229052566267
(1, 11102) 0.3949092185121046
(1, 10632) 0.4312617187682462
(1, 8241) 0.3258578319108594
(1, 5551) 0.3386111227996955
(1, 1312) 0.3170259473953434
(2, 20108) 0.5022569906568705
(2, 15125) 0.5343005392737009
(2, 13919) 0.4291973669910092
(2, 10711) 0.3218860801201089
(2, 10682) 0.4176599342300297
(3, 19630) 0.3319762714156978
(3, 12702) 0.42874250033416667
(3, 9519) 0.30403174684025974
(3, 6737) 0.5144099805244287
(3, 3886) 0.478937454646117
(3, 3643) 0.34574211055547416
(4, 20114) 0.6503873572333579
(4, 4824) 0.5237818190991552
: :
(25597, 17063) 0.3304874071423482
(25597, 15015) 0.2758643979318368
(25597, 12752) 0.3593165305248812
(25597, 7980) 0.3145854677806458
(25597, 5252) 0.44814672227160773
(25597, 3388) 0.4103709282996824
(25597, 3365) 0.46622391460595264
(25598, 15138) 0.40189051974385365
(25598, 7586) 0.4525700459104757
(25598, 4948) 0.4354158317495961
(25598, 1741) 0.45678939205141345
(25598, 1188) 0.4852018842252293
(25599, 20312) 0.3396795421202499
(25599, 18770) 0.3886163809353975
(25599, 16531) 0.34368990463114646
(25599, 4799) 0.41127319131630036
(25599, 3370) 0.34888914213527883
(25599, 2287) 0.48796263485146896
(25599, 1810) 0.29410127878285713
```

We trained the classification model using the numeric dataset we built in this way.

3. Classification Phase

During the classification phase, we compared both several classifiers and attribute selection techniques in order to understand which combination is the best.

Selected classifiers are:

- Multinomial Naïve Bayes Classifier
- Linear Support Vector Classifier
- Decision Tree Classifier
- K Nearest Neighbors Classifier (with K=1, K=3, K=5)
- Random Forest Classifier
- AdaBoost Classifier

For what concern attribute selection, we choose to compare configurations with and without the following attribute selection methods:

- Chi-square test, selecting best 50th percentile
- Stop-word filtering, using a customized stop-word list
- N-gram vectorization, with N=1 or N=2

In all tested configurations we decided to select at most 150 features, otherwise we would obtain too slow response times.

3.1 Result Evaluation

To compare results, we used precision, recall, f1-score, accuracy and execution time as metrics.

We decided to perform a cross-validation because we needed to use a validation technique for assessing how the results of our analysis would generalize to an independent data set. Indeed, cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset.

In particular, we performed 5-fold cross-validation: we randomly shuffled the dataset into five sets, so that all sets are equal size. Then, we trained the model on four of these sets and validated on the other one, doing this for all sets.

3.2 Comparison Between Classifiers

The list of all the tested combinations of stop-word filtering, N-gram vectorization and Chi-square filtering is shown in the following table.

Configuration name	Stop-word	N-gram	Chi-square
1	None	1	None
2	None	1	50%
3	None	2	None
4	None	2	50%
5	Custom	1	None
6	Custom	1	50%
7	Custom	2	None
8	Custom	2	50%

For each classifier, we tested all these configurations and the relative results are listed in the following table.

Classifier + Configuration name	Precision	Recall	F1-score	Accuracy	Execution time
MultinomialNB + configuration 1	90%	90%	90%	90%	0.001 s
MultinomialNB + configuration 2	86%	86%	86%	86%	0.001 s
MultinomialNB + configuration 3	74%	74%	73%	74%	0.001 s

MultinomialNB + configuration 4	71%	70%	70%	70%	0.002 s
MultinomialNB + configuration 5	86%	86%	86%	86%	0.021 s
MultinomialNB + configuration 6	84%	83%	83%	83%	0.002 s
MultinomialNB + configuration 7	68%	67%	66%	67%	0.001 s
MultinomialNB + configuration 8	66%	64%	63%	64%	0.004 s
LinearSVC + configuration 1	92%	92%	92%	92%	0.001 s
LinearSVC + configuration 2	91%	91%	91%	91%	0.003 s
LinearSVC + configuration 3	84%	80%	79%	80%	0.002 s
LinearSVC + configuration 4	82%	75%	74%	75%	0.001 s
LinearSVC + configuration 5	90%	90%	90%	90%	0.002 s
LinearSVC + configuration 6	89%	89%	89%	89%	0.002 s
LinearSVC + configuration 7	81%	70%	68%	70%	0.001 s
LinearSVC + configuration 8	80%	67%	63%	67%	0.001 s
DecisionTree + configuration 1	90%	90%	90%	90%	0.001 s
DecisionTree + configuration 2	91%	91%	91%	91%	0.001 s
DecisionTree + configuration 3	84%	80%	79%	80%	0.001 s
DecisionTree + configuration 4	82%	75%	74%	75%	0.001 s
DecisionTree + configuration 5	89%	89%	89%	89%	0.001 s
DecisionTree + configuration 6	89%	89%	89%	89%	0.001 s
DecisionTree + configuration 7	81%	70%	68%	70%	0.002 s
DecisionTree + configuration 8	80%	67%	63%	67%	0.002 s
KNN (K = 1) + configuration 1	87%	87%	87%	87%	0.004 s
KNN (K = 1) + configuration 2	85%	84%	84%	84%	0.004 s
KNN (K = 1) + configuration 3	74%	74%	74%	74%	0.002 s
KNN (K = 1) + configuration 4	71%	70%	70%	70%	0.002 s
KNN (K = 1) + configuration 5	84%	84%	84%	84%	0.005 s
KNN (K = 1) + configuration 6	82%	82%	82%	82%	0.022 s
KNN (K = 1) + configuration 7	68%	67%	66%	67%	0.005 s

KNN (K = 1) + configuration 8	66%	64%	63%	64%	0.006 s
KNN (K = 3) + configuration 1	89%	89%	89%	89%	0.004 s
KNN (K = 3) + configuration 2	86%	85%	85%	85%	0.003 s
KNN (K = 3) + configuration 3	75%	74%	74%	74%	0.002 s
KNN (K = 3) + configuration 4	71%	71%	70%	71%	0.002 s
KNN (K = 3) + configuration 5	85%	85%	85%	85%	0.020 s
KNN (K = 3) + configuration 6	83%	83%	83%	83%	0.003 s
KNN (K = 3) + configuration 7	68%	67%	66%	67%	0.002 s
KNN (K = 3) + configuration 8	66%	64%	63%	64%	0.002 s
KNN (K = 5) + configuration 1	90%	90%	90%	90%	0.005 s
KNN (K = 5) + configuration 2	90%	90%	90%	90%	0.003 s
KNN (K = 5) + configuration 3	75%	75%	75%	75%	0.002 s
KNN (K = 5) + configuration 4	71%	71%	70%	71%	0.003 s
KNN (K = 5) + configuration 5	89%	88%	88%	88%	0.005 s
KNN (K = 5) + configuration 6	88%	88%	88%	88%	0.002 s
KNN (K = 5) + configuration 7	61%	60%	59%	60%	0.003 s
KNN (K = 5) + configuration 8	66%	64%	63%	64%	0.003 s
RandomForest + configuration 1	92%	92%	92%	92%	0.010 s
RandomForest + configuration 2	91%	91%	91%	91%	0.016 s
RandomForest + configuration 3	84%	80%	79%	80%	0.013 s
RandomForest + configuration 4	82%	75%	74%	75%	0.010 s
RandomForest + configuration 5	90%	90%	90%	90%	0.010 s
RandomForest + configuration 6	89%	89%	89%	89%	0.009 s
RandomForest + configuration 7	81%	70%	68%	70%	0.022 s
RandomForest + configuration 8	80%	67%	63%	67%	0.024 s
AdaBoost + configuration 1	91%	91%	91%	91%	0.013 s
AdaBoost + configuration 2	91%	91%	91%	91%	0.012 s
AdaBoost + configuration 3	83%	75%	73%	75%	0.009 s

AdaBoost + configuration 4	82%	74%	72%	74%	0.009 s
AdaBoost + configuration 5	89%	88%	88%	88%	0.009 s
AdaBoost + configuration 6	89%	88%	88%	88%	0.011 s
AdaBoost + configuration 7	79%	66%	62%	66%	0.009 s
AdaBoost + configuration 8	79%	66%	61%	66%	0.013 s

These tests highlight that the best classifiers in terms of precision, recall, f1-score and accuracy are:

- Linear SVC (with configuration 1) which achieve about 92% for each of metric, with an execution time of 0.001 seconds
- Random forest (with configuration 1) which achieve about 92% for each of metric, with an execution time of 0.010 seconds
- Linear SVC (with configuration 2) which achieve about 91% for each of metric, with an execution time of 0.003 seconds
- Decision tree (with configuration 2) which achieve about 91% for each of metric, with an execution time of 0.001 seconds
- Random forest (with configuration 2) which achieve about 91% for each of metric, with an execution time of 0.016 seconds
- AdaBoost (with configuration 1 and 2) which achieve about 91% for each of metric, with an execution time respectively of 0.013 seconds and 0.012 seconds

We decided to discard configuration 2 of Linear SVC and configuration 2 of Random Forest because these classifiers have a better configuration both in terms of previously considered metrics and execution time. Furthermore, we excluded AdaBoost with both its configurations because they have execution times significantly higher than other classifiers. For the same reason we decided to discard also configuration 1 of Random Forest classifier because it is about 10 times slower than other classifiers, and this would become a big issue if the application would be used for processing automatically a large amount of titles, such as when used by web browsers.

For these reasons we compared the Linear SVC with configuration 1 and the Decision Tree with configuration 2. We computed for these two classifiers the respective confusion matrix, which are shown below.

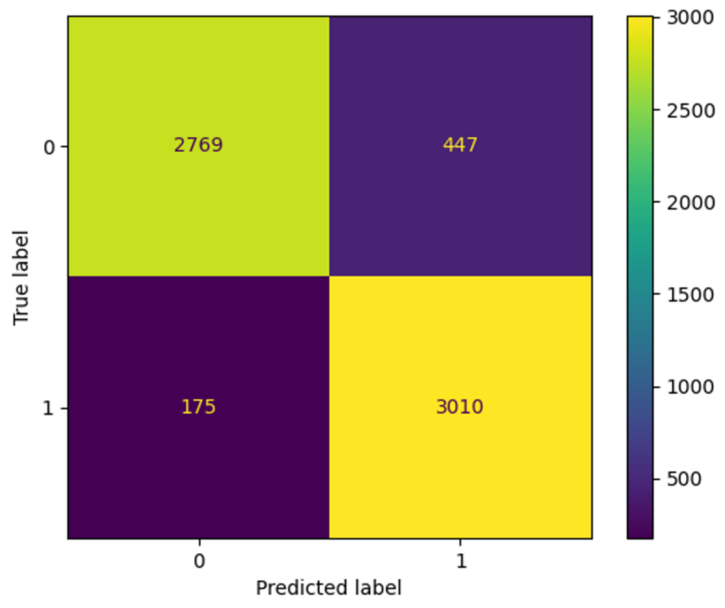


Fig.10: Decision Tree’s Confusion Matrix

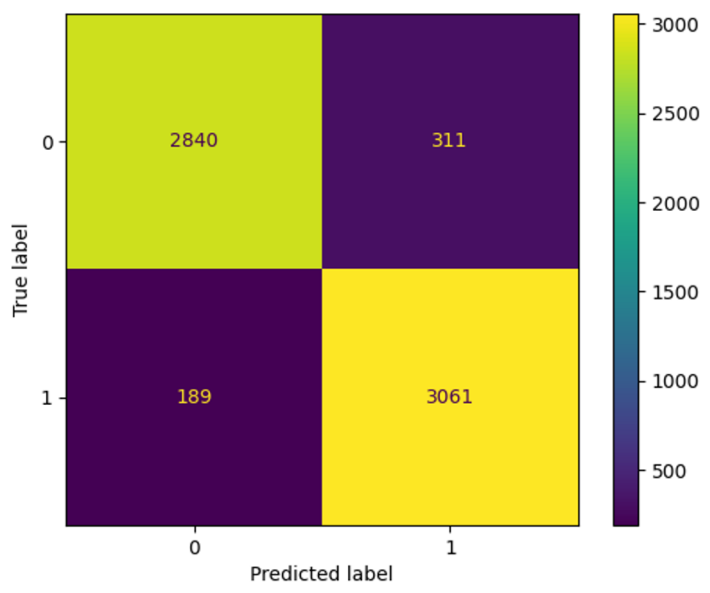


Fig.11: LinearSVC’s Confusion Matrix

Also from confusion matrices we can see that these two models perform very well in terms of accuracy, indeed the number of misclassified samples is negligible than the total number of samples.

To determine which of the two classifiers is the best, we assumed that the classification accuracy was normally distributed within each group and that the variances of the two populations are not reliably different, in order to use the t-test to evaluate if the two trained models are statistically different. In particular, we had to prove if the null hypothesis is valid, namely we verified if the two distributions of accuracy for the two models are the same. Therefore, we computed the t-statistic value as:

$$t = \frac{\widehat{p1} - \widehat{p2}}{\sqrt{\hat{p}(1-\hat{p})/n}}$$

where:

- $\hat{p} = \frac{x_1 - x_2}{2n}$
- $\widehat{p_1} = \frac{x_1}{n}$
- $\widehat{p_2} = \frac{x_2}{n}$
- n: number of samples
- x_1: number of samples correctly classified in Linear SVC
- x_2: number of samples correctly classified in Decision tree

We choose a significance level equal to 5%, so we extract from the T-Distribution table the correspondent value which is equal to 1.645. To verify the null hypothesis it must be true that: $-z < t < z$. In this case this isn't true, hence we can reject the null hypothesis, namely that the difference between the two models is statistically significant. For this reason, we selected as classifier the configuration 1 of Linear SVC because it reaches the best performances in terms of accuracy, precision, recall and f1-score.

3.3 Conclusions

Since the number of websites increased quickly during last years, the number of clickbait web pages increased proportionally. For this reason automatic clickbait detection has become particularly significant for web browsers because, otherwise, users would spend a lot of time to understand whether a website is reliable or not, making their internet browsing experience not comfortable.
Clickbait detector can also be used by companies which want to improve their marketing strategy campaign, by selecting an effective slogan or promotion message for a certain product or service. It can also be exploited by the SEO process to obtain highest positions in the list returned by search engines.
The high performances obtained by the Linear SVC guarantees to have quite reliable results during the classification phase, so this tool can be used safely by companies.