FEDERAL STATE AUTONOMUS EDUCATIONAL INSTITUTION

FOR HIGHER PROFESSIONAL EDUCATION

NATIONAL RESEARCH UNIVERSITY HIGHER

SCHOOL OF ECONOMICS

THE GRADUATED SCHOOL OF BUSINESS

GROUP PROJECT

DECISION ANALYSIS

“Identifying toxic comments using BigData algorithms. Comprehension SVM, Decision trees and Logistic Regression Algorithms”

Made by:

Belov Boris, BASB student

Okhapkina Arina, BASB student

Moscow
2021

Table of contents

Introduction of the problem

The threat of insults and harassment on the Internet means that many people stop expressing their opinions and refuse to seek a different opinion. Platforms are struggling to effectively facilitate communication, because of which many communities restrict or completely close user comments, which undoubtedly hinders both users and businesses. In addition, aggressive comments not only reduce people's desire to speak out, but also carry a potential danger from the point of view of the law and can also affect the image of the service in which they are left. Therefore, solutions are so necessary for quickly classifying comments into negative and neutral to facilitate their search and deletion.

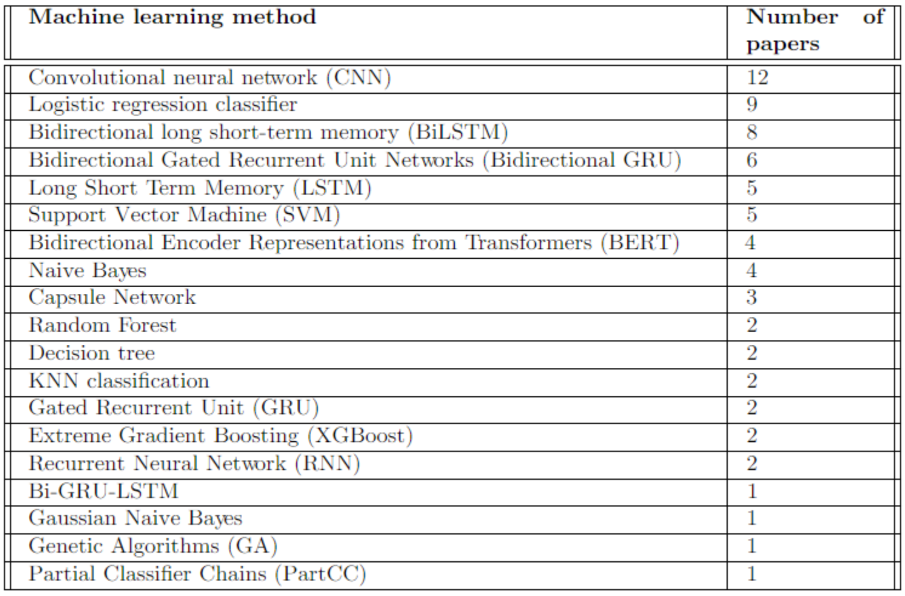
Artificial intelligence algorithms are well suited for solving problems of classifying comments into neutral and toxic. Many social networks and forums are already using artificial intelligence algorithms to solve this problem.

In this paper, three different algorithms will be tested to solve this problem: SVM, Decision tree, Logical regression.

We will see with what accuracy each of them copes with the task of binary classification to determine the most effective of them.

Literature review

Researchers have an interest to make papers which provide us methods to analyze which algorithm is the best to solve the problem of classification comments into negative or positive. Darko Androcec from University of Zagreb made research about how other scientist try to solve this problem and provide us information about which algorithm is most useful in this research. Below we can see result of his work.



Machine learning method	Number of papers
Convolutional neural network (CNN)	12
Logistic regression classifier	9
Bidirectional long short-term memory (BiLSTM)	8
Bidirectional Gated Recurrent Unit Networks (Bidirectional GRU)	6
Long Short Term Memory (LSTM)	5
Support Vector Machine (SVM)	5
Bidirectional Encoder Representations from Transformers (BERT)	4
Naive Bayes	4
Capsule Network	3
Random Forest	2
Decision tree	2
KNN classification	2
Gated Recurrent Unit (GRU)	2
Extreme Gradient Boosting (XGBoost)	2
Recurrent Neural Network (RNN)	2
Bi-GRU-LSTM	1
Gaussian Naive Bayes	1
Genetic Algorithms (GA)	1
Partial Classifier Chains (PartCC)	1

The picture below showed that the most useful and efficient methods are CNN, Logistic regression and BiLSTM [1]. But SVM and Decision trees are useful too.

So, let’s briefly look how researchers used these algorithms for our problem and see was it efficient or not.

For example, Prabowo and colleagues evaluated Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest Decision Tree (RFDT) algorithms for detecting hate speech and abusive language on Indonesian Twitter [2]. The experimental results demonstrated an accuracy of 68.43% for the hierarchical approach with word uni-gram features and the SVM model. It’s an example on non-Russian language dataset and the greatest number of research don’t analyze using of these methods on comments in Russian language.

The problem is that Russian Language Toxic Comments Dataset [3] is the only publicly available dataset of Russian-language toxic comments. However, this dataset was published at Kaggle without any description of the creation and annotation process, so it can be unreliable to use this dataset in academic papers[4].

But, Mujahed A. Saif, Alexander N. Medvedev, Maxim A. Medvedev, Todorka Atanasova from Ural Federal University used logistic regression model and three neural networks models - convolutional neural network (Conv), long shortterm memory (LSTM), and Conv + LSTM on the dataset in Russian language which we mentioned earlier. As a result, their mentioned that the most effective is the combined model Conv + LSTM, which the best accuracy: 0.9820 and 0.9645 when testing on 0.1 and 0.33 of the training data set respectively [5].

So, in our work we will try SVM, Decision trees and Logistic regression method on Russian Language Toxic Comments Dataset.

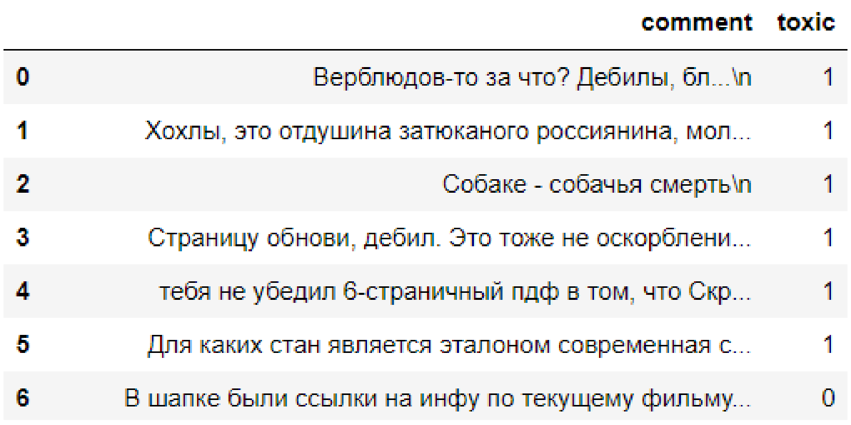
Data description

The selected data set was compiled from comments left on the Pikabu and 2ch platforms. In total, it contains 14412 unique values: 9586 (67%) neutral and 4826 (33%) poisonous comments. The data set consists of two columns:

- Comment – the text of the comment in Russian
- Toxic – if the comment is toxic - 1, otherwise - 0.

The dataset contains text that may be considered obscene, vulgar, or offensive.

Sample data:



	comment	toxic
0	Верблюдов-то за что? Дебилы, бл...\n	1
1	Хохлы, это отдушина затюканого россиянина, мол...	1
2	Собаке - собачья смерть\n	1
3	Страницу обнови, дебил. Это тоже не оскорблени...	1
4	тебя не убедил 6-страничный пдф в том, что Скр...	1
5	Для каких стан является эталоном современная с...	1
6	В шапке были ссылки на инфу по текущему фильму...	0

Methodology description

First, we defined the metrics that we will focus on when solving the problem. When developing a binary classifier, two metrics are of great importance:

1) Precision - the accuracy of the algorithm on neutral comments – the proportion of neutral comments that have been identified as neutral.

2) Recall - the accuracy of the algorithms on toxic comments – the proportion of toxic comments that have been identified as toxic.

The probability of recognizing a toxic comment is the target metric of the algorithm being developed. But for the application of this algorithm in real life, it is important that it correctly identifies neutral comments. When developing the algorithm, the lower bound for Precision was set to 0.95.

Before launching the algorithm, the data contained in the dataset was processed, namely:

1) Tokenization, where every word in the document is split into tokens

2) Removing stop words, punctuation, or unwanted tokens

3) Lemmatization/Stemming, Shorten words to their root stems, e.g. to take walk, walked, walking, walks as Walk etc.

4) Feature vector representation

SVM, Decision Trees and Logistic Regression algorithms were selected for testing:

For each algorithm, the following approach will be applied

- Using GridSearchCV, iterate through the different parameters of the model and select the parameters with which the model shows the greatest accuracy.
- Train the model with optimal parameters.
- Calculate Precision and Recall for the model (with default threshold value).
- Build a Precision and Recall dependency curve.
- Calculate Precision and Recall for different thresholds. (This and subsequent steps are not performed for Decision Trees)
- Select the smallest threshold value at which Precision will be more than 0.95.

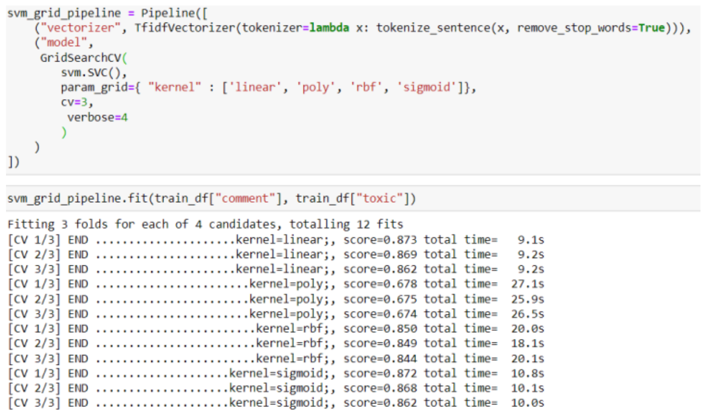
- Calculate Precision and Recall with the selected threshold value. The resulting Recall will be considered the best result of this algorithm.

To verify the results of the algorithms, the source data is divided into a dataset for training and a dataset for testing. The dataset for testing contains 500 records. The ratio of neutral and toxic comments in different datasets differs slightly.

Application of the model and results analysis

SVM:

1) To select the best configuration, different types of kernels were sorted out: 'linear’, 'poly’, ‘rbf’, 'sigmoid'.



```
svm_grid_pipeline = Pipeline([
    ("vectorizer", TfidfVectorizer(tokenizer=lambda x: tokenize_sentence(x, remove_stop_words=True))),
    ("model",
     GridSearchCV(
        svm.SVC(),
        param_grid={ "kernel" : ['linear', 'poly', 'rbf', 'sigmoid']},
        cv=3,
         verbose=4
        )
    )
])
```

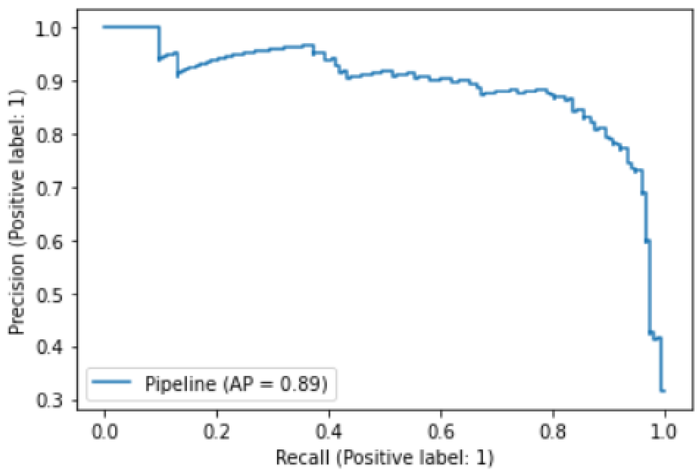
```
svm_grid_pipeline.fit(train_df["comment"], train_df["toxic"])
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV 1/3] END ....................kernel=linear;, score=0.873 total time=   9.1s
[CV 2/3] END ....................kernel=linear;, score=0.869 total time=   9.2s
[CV 3/3] END ....................kernel=linear;, score=0.862 total time=   9.2s
[CV 1/3] END ......................kernel=poly;, score=0.678 total time=  27.1s
[CV 2/3] END ......................kernel=poly;, score=0.675 total time=  25.9s
[CV 3/3] END ......................kernel=poly;, score=0.674 total time=  26.5s
[CV 1/3] END .......................kernel=rbf;, score=0.850 total time=  20.0s
[CV 2/3] END .......................kernel=rbf;, score=0.849 total time=  18.1s
[CV 3/3] END .......................kernel=rbf;, score=0.844 total time=  20.1s
[CV 1/3] END ...................kernel=sigmoid;, score=0.872 total time=  10.8s
[CV 2/3] END ...................kernel=sigmoid;, score=0.868 total time=  10.1s
[CV 3/3] END ...................kernel=sigmoid;, score=0.862 total time=  10.0s
```

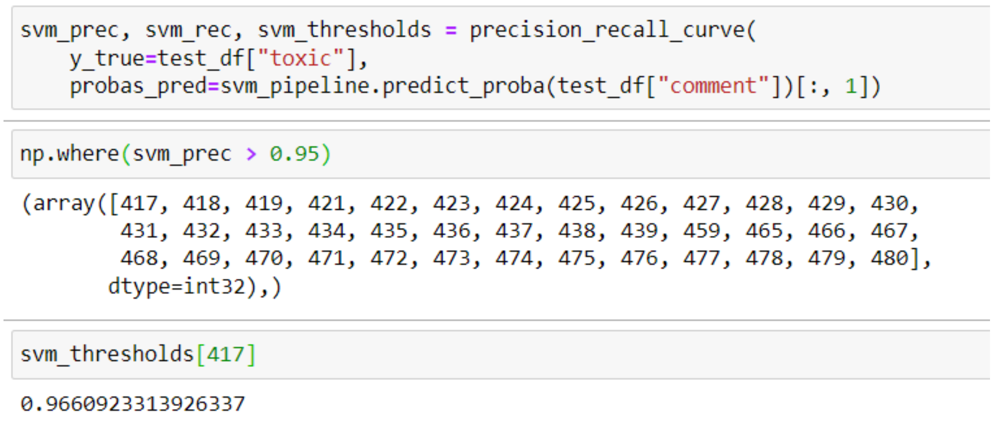
The best result was obtained with a "linear" type kernel.

2) After training the algorithm, the following values were obtained: Precision = 0.88, Recall = 0.77.

The curve of the dependence of Recall and Precision.



3) Next, the dependence of Recall and Precision on the threshold value was calculated and the minimum threshold value was selected, at which Recall>0.95. The resulting threshold value is 0.9661.



```
svm_prec, svm_rec, svm_thresholds = precision_recall_curve(
    y_true=test_df["toxic"],
    probas_pred=svm_pipeline.predict_proba(test_df["comment"])[:, 1])
```

```
np.where(svm_prec > 0.95)
```

```
(array([417, 418, 419, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430,
        431, 432, 433, 434, 435, 436, 437, 438, 439, 459, 465, 466, 467,
        468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480],
       dtype=int32),)
```

```
svm_thresholds[417]
```

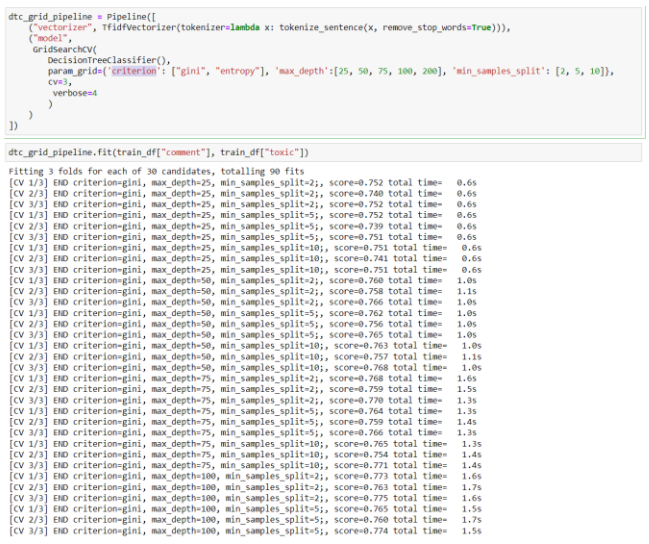
```
0.9660923313926337
```

4) Results at this threshold value: Precision = 0.95, Recall = 0.39.

The best accuracy of the SVM algorithm is 0.39.

Decision Tree:

1) To select the best configuration, the parameters criterion (the function to measure the quality of a split), min_samples_split (the minimum number of samples required to split an internal node), max_depth (the maximum depth of the tree.) was iterated over. (The screenshot does not show all the results of the search)



```
dtc_grid_pipeline = Pipeline([
    ("vectorizer", TfidfVectorizer(tokenizer=lambda x: tokenize_sentence(x, remove_stop_words=True))),
    ("model",
     GridSearchCV(
        DecisionTreeClassifier(),
        param_grid={'criterion': ["gini", "entropy"], 'max_depth':[25, 50, 75, 100, 200], 'min_samples_split': [2, 5, 10]},
        cv=3,
         verbose=4
        )
    )
])
```

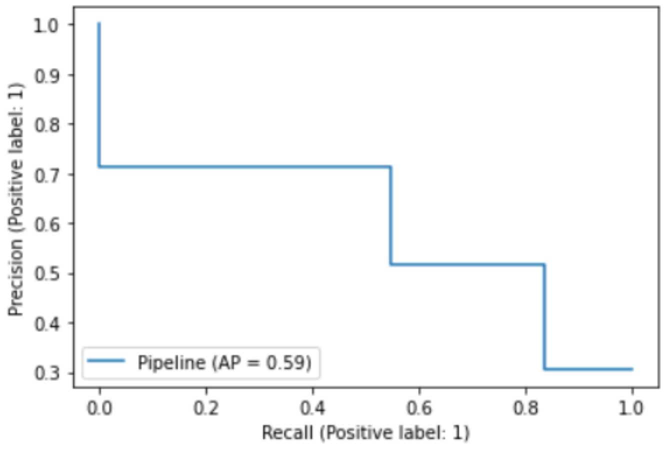
```
dtc_grid_pipeline.fit(train_df["comment"], train_df["toxic"])
```

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits
[CV 1/3] END criterion=gini, max_depth=25, min_samples_split=2;, score=0.752 total time=   0.6s
[CV 2/3] END criterion=gini, max_depth=25, min_samples_split=2;, score=0.740 total time=   0.6s
[CV 3/3] END criterion=gini, max_depth=25, min_samples_split=2;, score=0.752 total time=   0.6s
[CV 1/3] END criterion=gini, max_depth=25, min_samples_split=5;, score=0.752 total time=   0.6s
[CV 2/3] END criterion=gini, max_depth=25, min_samples_split=5;, score=0.739 total time=   0.6s
[CV 3/3] END criterion=gini, max_depth=25, min_samples_split=5;, score=0.751 total time=   0.6s
[CV 1/3] END criterion=gini, max_depth=25, min_samples_split=10;, score=0.751 total time=   0.6s
[CV 2/3] END criterion=gini, max_depth=25, min_samples_split=10;, score=0.741 total time=   0.6s
[CV 3/3] END criterion=gini, max_depth=25, min_samples_split=10;, score=0.751 total time=   0.6s
[CV 1/3] END criterion=gini, max_depth=50, min_samples_split=2;, score=0.760 total time=   1.0s
[CV 2/3] END criterion=gini, max_depth=50, min_samples_split=2;, score=0.758 total time=   1.1s
[CV 3/3] END criterion=gini, max_depth=50, min_samples_split=2;, score=0.766 total time=   1.0s
[CV 1/3] END criterion=gini, max_depth=50, min_samples_split=5;, score=0.762 total time=   1.0s
[CV 2/3] END criterion=gini, max_depth=50, min_samples_split=5;, score=0.756 total time=   1.0s
[CV 3/3] END criterion=gini, max_depth=50, min_samples_split=5;, score=0.765 total time=   1.0s
[CV 1/3] END criterion=gini, max_depth=50, min_samples_split=10;, score=0.763 total time=   1.0s
[CV 2/3] END criterion=gini, max_depth=50, min_samples_split=10;, score=0.757 total time=   1.1s
[CV 3/3] END criterion=gini, max_depth=50, min_samples_split=10;, score=0.768 total time=   1.0s
[CV 1/3] END criterion=gini, max_depth=75, min_samples_split=2;, score=0.768 total time=   1.6s
[CV 2/3] END criterion=gini, max_depth=75, min_samples_split=2;, score=0.759 total time=   1.5s
[CV 3/3] END criterion=gini, max_depth=75, min_samples_split=2;, score=0.770 total time=   1.3s
[CV 1/3] END criterion=gini, max_depth=75, min_samples_split=5;, score=0.764 total time=   1.3s
[CV 2/3] END criterion=gini, max_depth=75, min_samples_split=5;, score=0.759 total time=   1.4s
[CV 3/3] END criterion=gini, max_depth=75, min_samples_split=5;, score=0.766 total time=   1.3s
[CV 1/3] END criterion=gini, max_depth=75, min_samples_split=10;, score=0.765 total time=   1.3s
[CV 2/3] END criterion=gini, max_depth=75, min_samples_split=10;, score=0.754 total time=   1.4s
[CV 3/3] END criterion=gini, max_depth=75, min_samples_split=10;, score=0.771 total time=   1.4s
[CV 1/3] END criterion=gini, max_depth=100, min_samples_split=2;, score=0.773 total time=   1.6s
[CV 2/3] END criterion=gini, max_depth=100, min_samples_split=2;, score=0.763 total time=   1.7s
[CV 3/3] END criterion=gini, max_depth=100, min_samples_split=2;, score=0.775 total time=   1.6s
[CV 1/3] END criterion=gini, max_depth=100, min_samples_split=5;, score=0.765 total time=   1.5s
[CV 2/3] END criterion=gini, max_depth=100, min_samples_split=5;, score=0.760 total time=   1.7s
[CV 3/3] END criterion=gini, max_depth=100, min_samples_split=5;, score=0.774 total time=   1.5s
```

The best result turned out to be the following values criterion="gini", max_depth=200, min_samples_split=2.

2) After training the algorithm, the following values were obtained Precision = 0.71, Recall = 0.54.

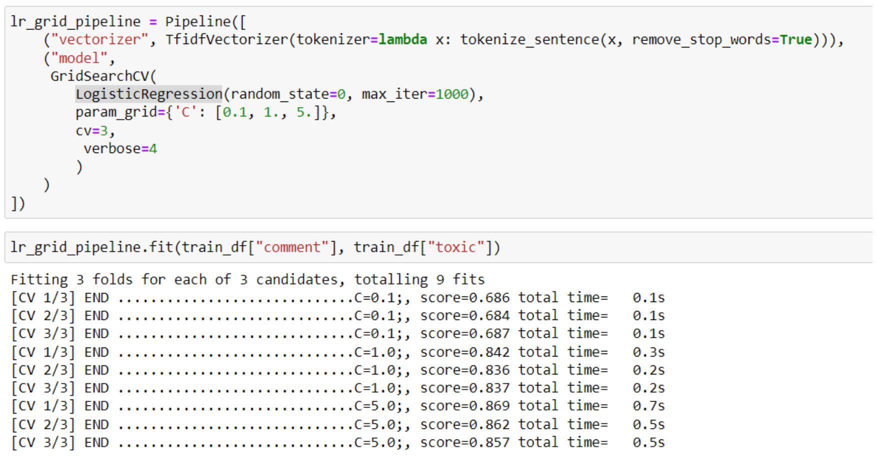
The curve of the dependence of Recall and Precision.



In this algorithm, it was not possible to obtain a Precision of at least 0.95, so no further steps were performed.

Logistic regression:

1) To select the best configuration, the Inverse of regularization strength parameter was iterated.



```
lr_grid_pipeline = Pipeline([
    ("vectorizer", TfidfVectorizer(tokenizer=lambda x: tokenize_sentence(x, remove_stop_words=True))),
    ("model",
     GridSearchCV(
        LogisticRegression(random_state=0, max_iter=1000),
        param_grid={'C': [0.1, 1., 5.]},
        cv=3,
         verbose=4
        )
    )
])
```

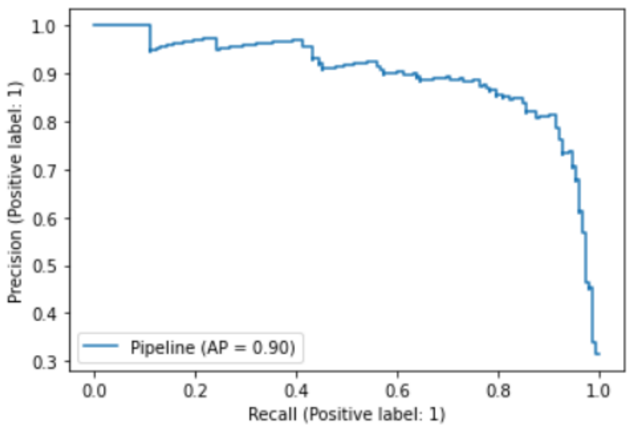
```
lr_grid_pipeline.fit(train_df["comment"], train_df["toxic"])
```

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
[CV 1/3] END ............................C=0.1;, score=0.686 total time=   0.1s
[CV 2/3] END ............................C=0.1;, score=0.684 total time=   0.1s
[CV 3/3] END ............................C=0.1;, score=0.687 total time=   0.1s
[CV 1/3] END ............................C=1.0;, score=0.842 total time=   0.3s
[CV 2/3] END ............................C=1.0;, score=0.836 total time=   0.2s
[CV 3/3] END ............................C=1.0;, score=0.837 total time=   0.2s
[CV 1/3] END ............................C=5.0;, score=0.869 total time=   0.7s
[CV 2/3] END ............................C=5.0;, score=0.862 total time=   0.5s
[CV 3/3] END ............................C=5.0;, score=0.857 total time=   0.5s
```

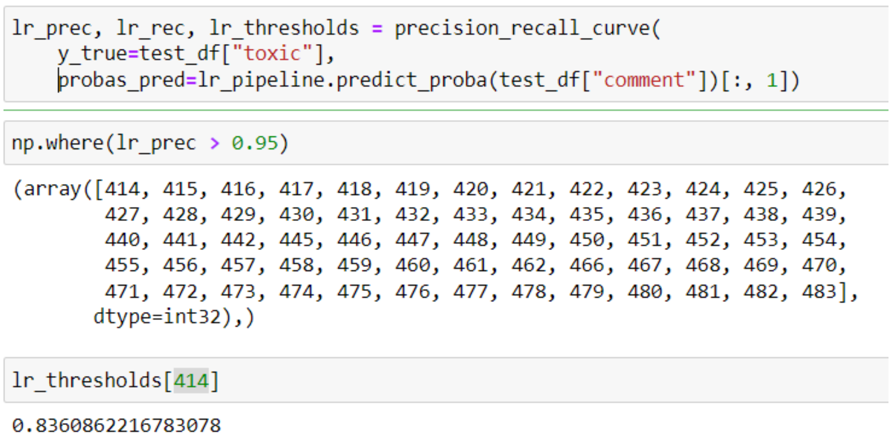
The best result is Inverse of regularization strength = 5.

2) After training the algorithm, Precision = 0.87, Recall = 0.77 were obtained.

The curve of the dependence of Recall and Precision.



3) Next, the dependence of Recall and Precision on the threshold value was calculated and the minimum threshold value was selected, at which Recall>0.95. The threshold value is 0.836.



```
lr_prec, lr_rec, lr_thresholds = precision_recall_curve(
    y_true=test_df["toxic"],
    probas_pred=lr_pipeline.predict_proba(test_df["comment"])[:, 1])
```

```
np.where(lr_prec > 0.95)
```

```
(array([414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426,
        427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439,
        440, 441, 442, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
        455, 456, 457, 458, 459, 460, 461, 462, 466, 467, 468, 469, 470,
        471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483],
       dtype=int32),)
```

```
lr_thresholds[414]
```

```
0.8360862216783078
```

4) Results at this threshold value: Precision = 0.96, Recall = 0.42.

The best accuracy of the Logistic Regression algorithm is 0.42.

Source Python3 code:

https://github.com/ArinaOkhapkina/toxic_comment_classifiacation

Conclusion

The task of classifying comments into toxic and neutral is currently relevant for many companies. Its solution allows you to quickly respond to violations and reduce costs for the company.

To solve this problem, three algorithms were used – SVM, Decision tree, Logical regression. Logical regression proved to be the best algorithm, with Recall 0.42. SVM showed a slightly lower accuracy 0.39. The Decision tree turned out to be not optimal for this task.

From the above, it should be concluded that the Logical Regression algorithm turned out to be the most optimal for selected dataset.

References:

1. Darko Androcec, Machine learning methods for toxic comment classification: a systematical review. Acta Univ. Sapientiae Informatica 12, 2. 2020. Pp. 205–216.

2. Prabowo, F. A. et al., Hierarchical multi-label classification to identify hate speech and abusive language on indonesian twitter. In: 2019 6th international conference on information technology, computer, and electrical engineering (icitacee). 2019. Pp. 1–5.

3. Russian Language Toxic Comments Dataset. Available on https://www.kaggle.com/blackmoon/russian-language-toxic-comments. 2018.

4. Sergey Smetanin, Toxic Comments Detection in Russian. Available on https://towardsdatascience.com/toxic-comment-detection-in-russian-2950be3b9787. 2020.

5. Mujahed A. Saif, Alexander N. Medvedev, Maxim A. Medvedev, and Todorka Atanasova, Classification of online toxic comments using the logistic regression and neural networks models. Available on https://aip.scitation.org/doi/10.1063/1.5082126. 2018.