URL Categorization

Domantas Meidus

Abstract

In this document we report our proposal for the application of supervised classification methods to the URL Categorization problem. The classifiers that we have selected for the classification tasks were: Support vector machine(SVM), Logistic regression, Decision Trees, KNeighbors. I have implemented the classification process using the scikit-learn, nltk, numpy, urlib, BeautifulSoup and langdetect libraries. The classifiers is learnt by using the train data and computing the accuracy in the test data. From our results the best accuracy has been produced by the Logistic Regression Classifier.

1 Description of the problem

The task we have to solve is the classification of the "URL categorization", introduced in the project task.

The dataset includes category which each URL belongs to. The goal of the project is to predict the category of URL, among as many categories.

The database has the following characteristics:

- 18 attributes.
- 4 categories.
- 846 instances.

2 Description of our approach

The implementation of the project according to the tasks:

1. Design any preprocessing of the web pages in the dataset.
2. Define and learn the classifier using the training data.
3. Design the validation method to evaluate the accuracy of the proposed classification approach.

2.1 Preprocessing

For preprocessing data these steps have been made:

1. Using urlopen [1] html content of URL is downloaded and with BeautifulSoup tool it is converted to the plain text if URL "Main category" is not 'Not working' and main_category:confidence value is greater than 0.5.
2. Using langdetect tool by plain URL text program detect language which is used in URL content.

3. If detected language is english and URL domain belongs to 'com', 'org', 'net', '.us', '.uk', '.au' or '.ca' domains - text is tokenize by using nltk tool.

   These actions is done for all URL from URL categorization file.

4. After all URL is filtered, our program determinate which categories are compatible with cross_val_predict method: if each category is used more than the number of cross_val_predict parameter CV, that means that category can be predicted by cross_val_predict method.

   These applicable categories are marked as classifier labels.

5. For each category top 50 words which are the most frequently used in URL is stored in list excluding "stop words" and symbols in range 32-64 and 91 - 96 decimal values which represented in ASCII table (http://www.asciitable.com/).

6. Vector with zero values is created for each URL with top 50 words for each category. If each filtered URL top most frequently words contains categories used most frequently words, then value of word position in vector is changed to value 1.

   After this action vector representation of each filtered URL is marked as classifier features.

The data was split into two different sets: train and test, for validation. We use the same train set to learn all the classifiers, and the same test set for evaluating their accuracy.

2.2 Classifiers

We use three different classifiers:

1. Logistic regression
2. Decision trees
3. Support Vector Machine
4. KNeighbors with parameters:
   - n_neighbors=5
   - metric="euclidean"

For each classifier, these were the parameters by default of the scikit-learn library.

2.3 Validation

To validate our results we compute the classifier accuracy in the test data. Another possibility was to compute the cross-validation in the complete dataset but we used the split between train and test because it was simpler.

As an additional validation step we computed the confusion matrices for the three classifiers.

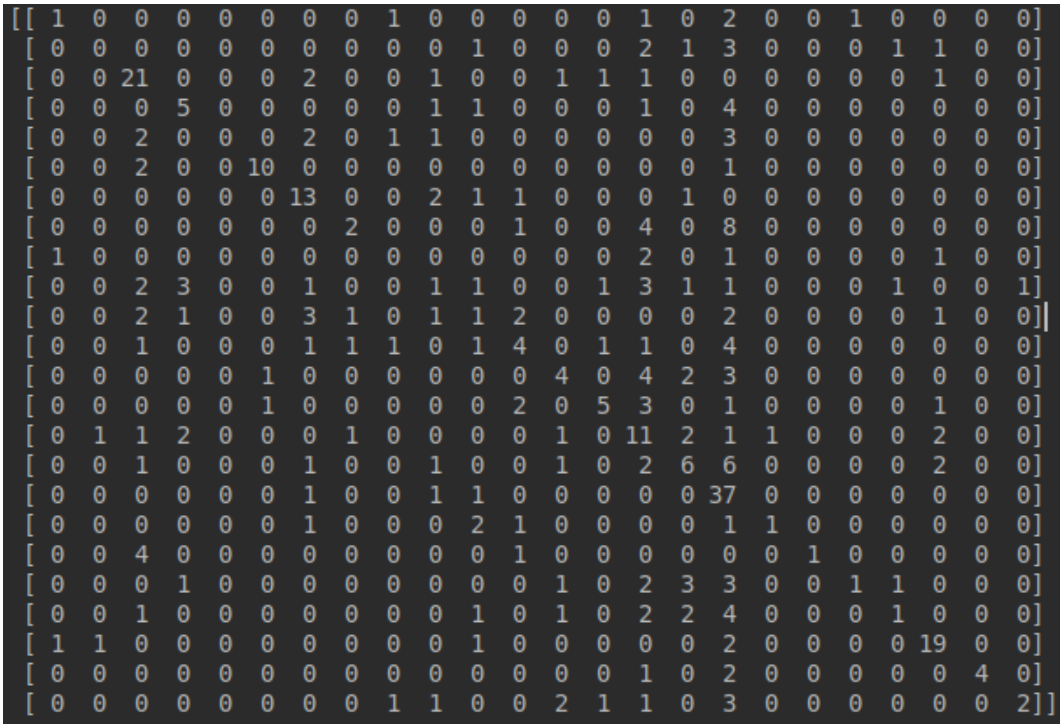
3 Implementation

All the project steps were implemented in Python. Libraries that was used in the project: urlib and BeautifulSoup for downloading URL and parsing HTML code into plain text, langdetect for language detection from plain text, numpy for using data structures to store labels and features information and scikit-learn for the classification tasks. Illustration how the implementation works in the Python notebook `URL-categorization-Jupyter-Notebook.ipynb`.

4 Results

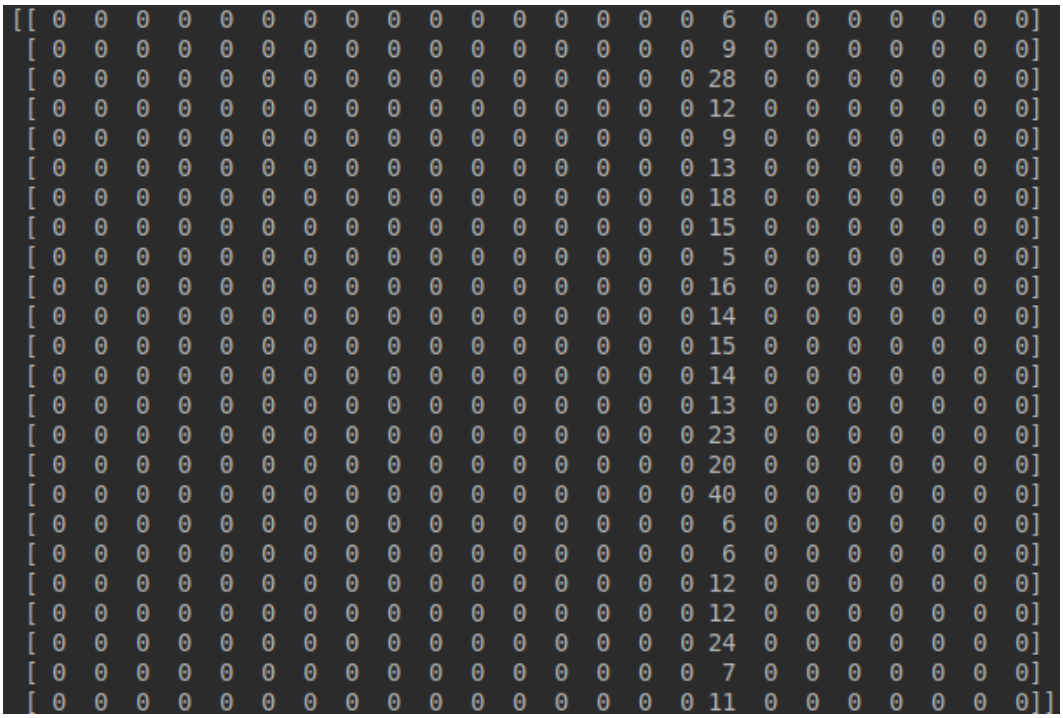
The accuracies produced by the Logistic regression, Decision tree, Support Vector Machine and KNeighbors classifiers were, respectively: 0.4310, 0.2528, 0.1149 and 0.0919. Therefore, the best classifier was logistic regression.

The results of the computation of the confusion matrices for Logistic regression, Support Vector Machine, Decision tree and KNeighbors classifiers are respectively shown in Tables 1, 2, 3 and 4.



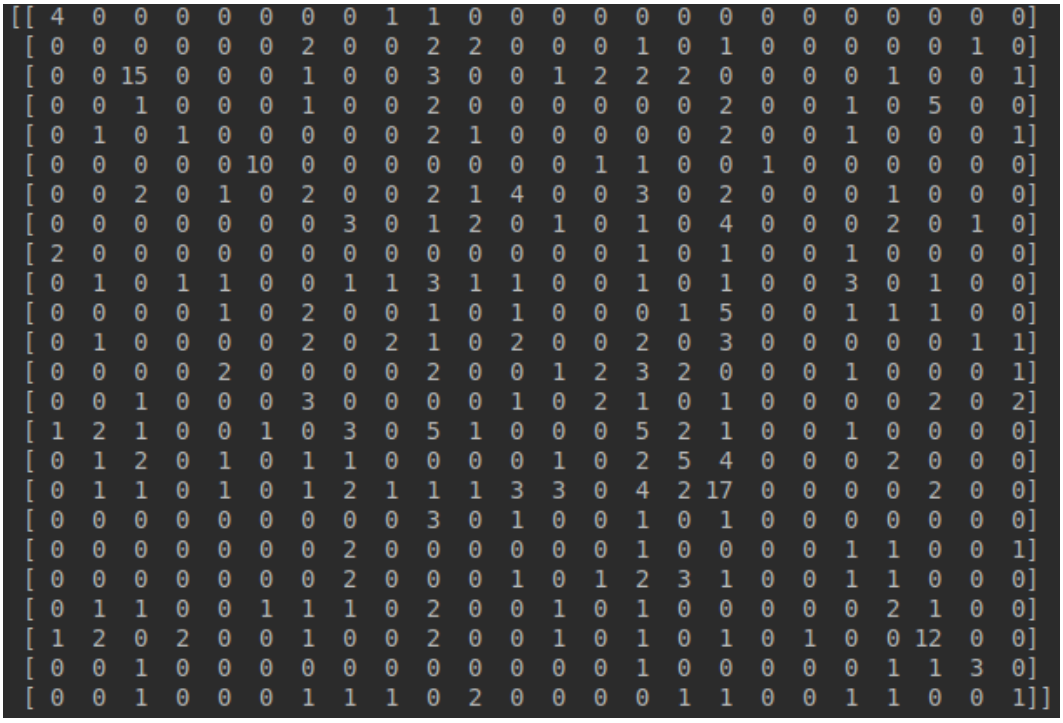
```
[[ 1  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  2  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0  0  2  1  3  0  0  0  1  1  0  0]
 [ 0  0 21  0  0  0  2  0  0  1  0  0  1  1  1  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  5  0  0  0  0  0  1  1  0  0  0  1  0  4  0  0  0  0  0  0  0]
 [ 0  0  2  0  0  0  2  0  1  1  0  0  0  0  0  0  3  0  0  0  0  0  0  0]
 [ 0  0  2  0  0 10  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 13  0  0  2  1  1  0  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  1  0  0  4  0  8  0  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  1  0  0  0  0  1  0  0]
 [ 0  0  2  3  0  0  1  0  0  1  1  0  0  1  3  1  1  0  0  0  1  0  0  1]
 [ 0  0  2  1  0  0  3  1  0  1  1  2  0  0  0  0  2  0  0  0  0  1  0  0]
 [ 0  0  1  0  0  0  1  1  1  0  1  4  0  1  1  0  4  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  4  0  4  2  3  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  2  0  5  3  0  1  0  0  0  0  1  0  0]
 [ 0  1  1  2  0  0  0  1  0  0  0  0  1  0 11  2  1  1  0  0  0  2  0  0]
 [ 0  0  1  0  0  0  1  0  0  1  0  0  1  0  2  6  6  0  0  0  0  2  0  0]
 [ 0  0  0  0  0  0  1  0  0  1  1  0  0  0  0  0 37  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  2  1  0  0  0  0  1  1  0  0  0  0  0  0]
 [ 0  0  4  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  1  0  2  3  3  0  0  1  1  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  1  0  1  0  2  2  4  0  0  0  1  0  0  0]
 [ 1  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  2  0  0  0  0 19  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  2  0  0  0  0  0  4  0]
 [ 0  0  0  0  0  0  0  0  1  1  0  0  2  1  1  0  3  0  0  0  0  0  0  2]]
```

Figure 1: Logistic Regression confusion matrix



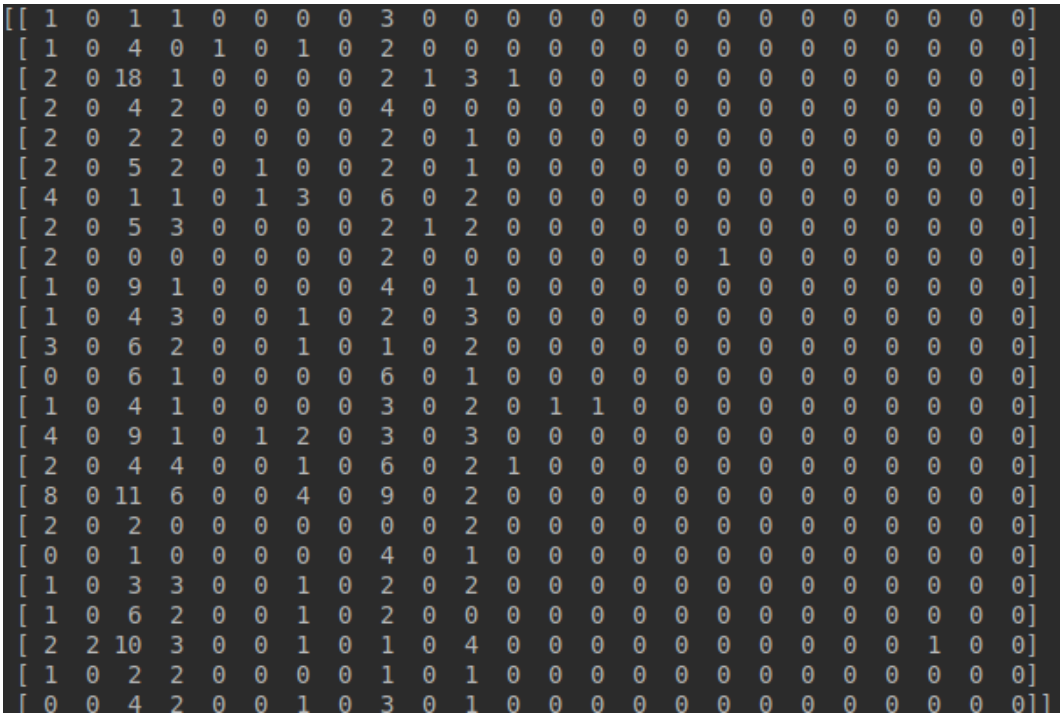
```
[[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 28  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 18  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 13  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 24  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0]]
```

Figure 2: Support vector machine confusion matrix



```
[[ 4  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  0  0  2  2  0  0  0  1  0  1  0  0  0  0  0  1  0]
 [ 0  0 15  0  0  0  1  0  0  3  0  0  1  2  2  2  0  0  0  0  1  0  0  1]
 [ 0  0  1  0  0  0  1  0  0  2  0  0  0  0  0  0  2  0  0  1  0  5  0  0]
 [ 0  1  0  1  0  0  0  0  0  2  1  0  0  0  0  0  2  0  0  1  0  0  0  1]
 [ 0  0  0  0  0 10  0  0  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0  0]
 [ 0  0  2  0  1  0  2  0  0  2  1  4  0  0  3  0  2  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  3  0  1  2  0  1  0  1  0  4  0  0  0  2  0  1  0]
 [ 2  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  1  0  0  0  0]
 [ 0  1  0  1  1  0  0  1  1  3  1  1  0  0  1  0  1  0  0  3  0  1  0  0]
 [ 0  0  0  0  1  0  2  0  0  1  0  1  0  0  0  1  5  0  0  1  1  1  0  0]
 [ 0  1  0  0  0  0  2  0  2  1  0  2  0  0  2  0  3  0  0  0  0  0  1  1]
 [ 0  0  0  0  2  0  0  0  0  2  0  0  1  2  3  2  0  0  0  1  0  0  0  1]
 [ 0  0  1  0  0  0  3  0  0  0  0  1  0  2  1  0  1  0  0  0  0  2  0  2]
 [ 1  2  1  0  0  1  0  3  0  5  1  0  0  0  5  2  1  0  0  1  0  0  0  0]
 [ 0  1  2  0  1  0  1  1  0  0  0  0  1  0  2  5  4  0  0  0  2  0  0  0]
 [ 0  1  1  0  1  0  1  2  1  1  1  3  3  0  4  2 17  0  0  0  0  2  0  0]
 [ 0  0  0  0  0  0  0  0  0  3  0  1  0  0  1  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  0  0  0  1  0  0  0  0  1  1  0  0  1]
 [ 0  0  0  0  0  0  0  2  0  0  0  1  0  1  2  3  1  0  0  1  1  0  0  0]
 [ 0  1  1  0  0  1  1  1  0  2  0  0  1  0  1  0  0  0  0  0  2  1  0  0]
 [ 1  2  0  2  0  0  1  0  0  2  0  0  1  0  1  0  1  0  1  0  0 12  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  1  3  0]
 [ 0  0  1  0  0  0  1  1  1  0  2  0  0  0  0  1  1  0  0  1  1  0  0  1]]
```

Figure 3: Decision tree confusion matrix



```
[[ 1  0  1  1  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  4  0  1  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0 18  1  0  0  0  0  2  1  3  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  4  2  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  2  2  0  0  0  0  2  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  5  2  0  1  0  0  2  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 4  0  1  1  0  1  3  0  6  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  5  3  0  0  0  0  2  1  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0]
 [ 1  0  9  1  0  0  0  0  4  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  4  3  0  0  1  0  2  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 3  0  6  2  0  0  1  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  6  1  0  0  0  0  6  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  4  1  0  0  0  0  3  0  2  0  1  1  0  0  0  0  0  0  0  0  0  0]
 [ 4  0  9  1  0  1  2  0  3  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  4  4  0  0  1  0  6  0  2  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 8  0 11  6  0  0  4  0  9  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  0  2  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  4  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  3  3  0  0  1  0  2  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  6  2  0  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 2  2 10  3  0  0  1  0  1  0  4  0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 1  0  2  2  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  4  2  0  0  1  0  3  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]]
```

Figure 4: KNeighbors confusion matrix

5 Conclusions

In our project we have applied Logistic regression, Decision tree, Support Vector Machine and KNeighbors classifiers to the "URL categorization", introduced in [2]. We have computed the accuracy of these classifiers and observed that Logistic regression produces the highest accuracy.

References

[1] urllib documentation. "https://docs.python.org/3/library/urllib.request.html".

[2] URL categorization. "https://www.crowdflower.com/data-for-everyone/", 2015.