Data Science Nanodegree

Capstone Project

Bader Alotaibi
August 23, 2019

I. Definition

Project Overview

Even though many technical communities such as Stackoverflow and Quora enable users to tag their questions, many times users miss-tag or forget to tag their questions with the appropriate programming language. This kind of issue reduce user interaction and many times prevent the question being directed or recommended to the right sub community within the forum. It also makes it difficult for users to search for relevant questions and create a great amount of duplicate questions. Eventually, miss-tagging create user dissatisfaction and the quality decay across the platform.

Problem Statement

In recent years, technical platforms such as Stackoverflow became the first stop for many users to seek community support and advice. However, with the rise of number of users and the wide range of programming languages in the market. It became important to ensure questions are tagged properly. In this project, the goal is to develop a method to predict what is the programming language for this question based on the code snippet. The result of the machine learning model to be developed is to identify the programming language a question should be tagged with.

Metrics

The evaluation metrics to be used in this project are the accuracy, precision, recall and F-1 Score All the evaluation metrics are defined below:

$$\text{Accuracy} = \frac{True\ Positive + True\ Negative}{Total}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F-1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

The above-mentioned evaluation metrics provide clear indications on how the model perform.

II. Analysis

Data Exploration

In this project the Stackoverflow dataset was used. In this dataset there are 200,000 instances and 2 features. The features are the content of the question and the tags set by the user. The tags could be the programming language and other technical terms such as user management, software requirements or some emerging technologies such as Data science. Additional features are constructed such as the code snippet length.

In the table below, we can see descriptive analysis of the dataset:

Table 1: Code snippet length

Code Length	
count	200000
mean	686.8971
std	1198.439
min	1
25%	145
50%	344
75%	755
max	25137

During this exploration analysis, questions without any code snippets or with codes less than 50 characters are removed as they near impossible to detect the programming language. From the table below, we can see the statistics of the dataset after processing and cleaning:

Table 2: Code Snippet Length after Processing

Code Length	
count	147241
mean	359.7911
std	245.2191
min	51
25%	156
50%	297
75%	518
max	999

Exploratory Visualization

In this section, we will look into some of the features with interesting insights. In figure 1, we can see the distribution of the programming languages in the data set is fairly balanced.

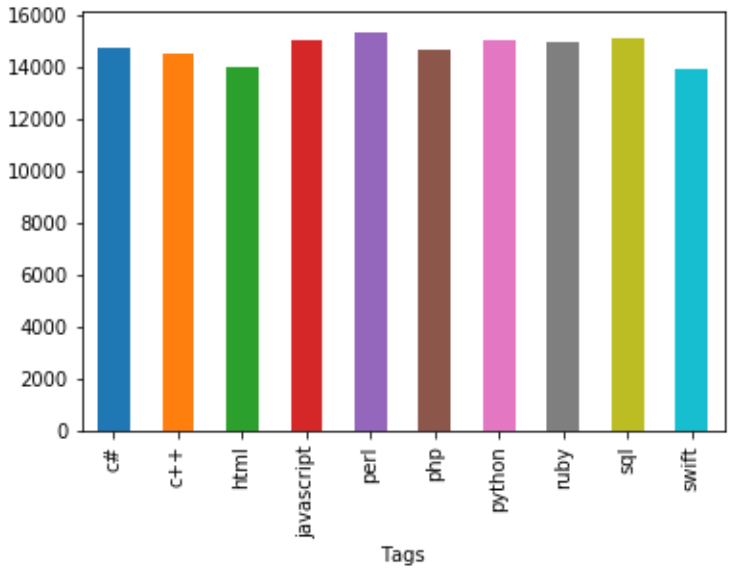

Figure 1: Programming Languages Distribution

In figure 2, we see that the length of the code snippets by programming languages. In figure 2 we can see the median length of code snippet per programming language which is between 200 and 400 characters. Also we can see that the 75% is less than 600 characters for all languages.

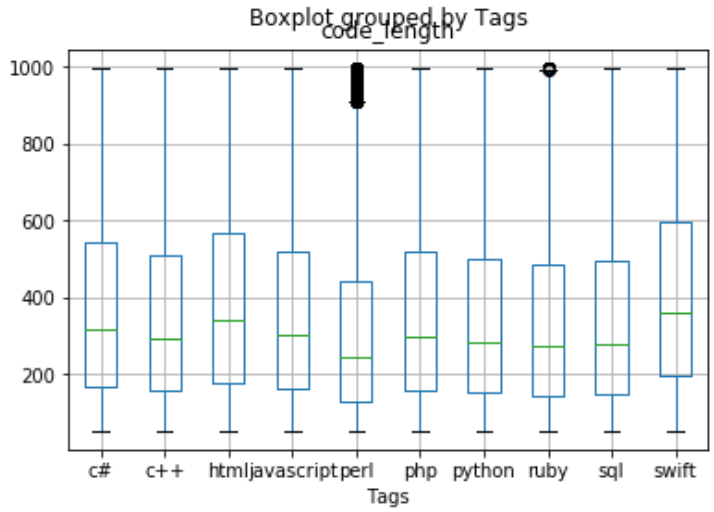

Figure 2: Code Snippet Length by Language

Algorithms and Techniques

In this work, I decided to use two algorithms. The first algorithm is Naïve Bayes (NB) estimate the conditional probability with the assumption that the features are conditionally independent given the class label. NB perform well with irrelevant features as their conditional probability will becomes almost uniformly distributed. However, it is sensitive to correlated features as the conditional probability assumption no longer hold which lead to slower performance. NB is good and quick way to establish a base line of the least performance we can expect.

The second algorithm, is Support Vector Machine (SVM). SVM try to find the optimal hyperplane that divide the classes. There are some key parameters need to optimized in order to find the optimal hyperplane. Namely, the regularization factor which control how much we want to avoid misclassification. Another key parameter is the kernel type which map the data into higher dimensional space to be linearly separable. Also, Gamma which control the influence of a single training example.

Benchmark

All algorithms results will be benchmarked based on the metrics defined above will be compared.

III. Methodology

Data Preprocessing

The data has 3 features which may not be all important to predict the target variable. In this project TFIDF, short for term frequency–inverse document frequency is used to generate the numerical features to be used within the model

Implementation

In this project solution the workflow can be seen in figure 3 below.

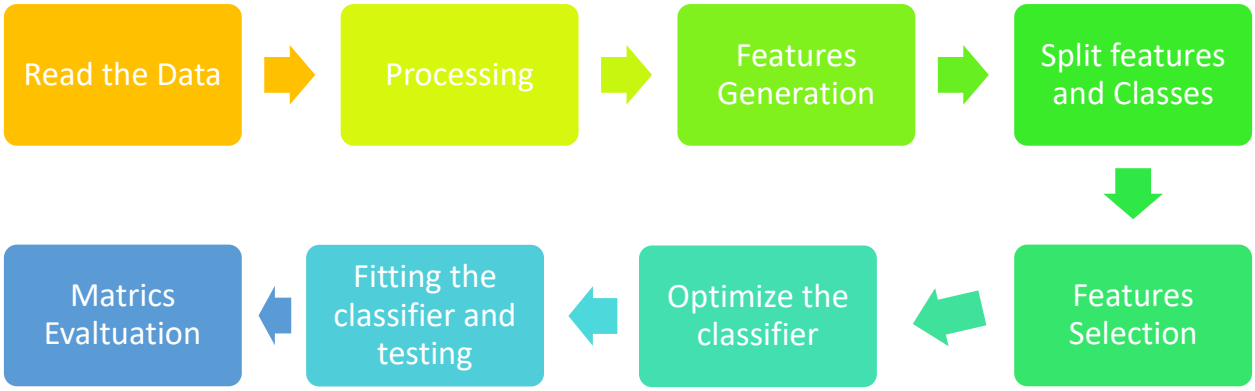

Figure 3: Project Workflow Overview

The dataset was loaded from Excel file and loaded into a data structure for analysis and further processing. The first step after loading the data is to process it, split the input from the target variable, split the data into testing and training and then apply TFIDF on each set. The testing data is 33% of the original data set.

Refinement

The algorithms out-of-the-box without any optimization did not preform as well as expected. Therefore, different parameters were tested in order to find the best parameters possible.

IV. Results

Model Evaluation and Validation

The final model seems be reasonable and align with solution expectations.

The result of NB as a baseline seems to perform somewhat well. The accuracy is 0.67, precision 0.68, recall is 0.68, and F1-score is 0.68. This is done with training time of 0.3 seconds.

While as expected SVM performed much better. The accuracy is 0.71, precision 0.72, recall is 0.72, and F1-score is 0.72. This is done with training time of 13 seconds.

Justification

The result produced by SVM is stronger than the baseline (NB). The is probably due to the fact that NB assume that the features are independent. SVM is also much better when it comes to more complex linear data.

V. Conclusion

In this project, I attempted to use machine learning to predict the programming language of a given code snippet. Platforms such as Stackoverflow are challenges daily with many users miss-tagging or forget to tag their questions with the appropriate programming language. This kind of issue reduce user interaction and many times prevent the question being directed or recommended to the right sub community within the platform. It also makes it difficult for users to search for relevant questions and create a great amount of duplicate questions. The solution proposed utilized Stackoverflow dataset and has been developed with good result. In this project, three well known algorithms were tested NB and SVM. The final result shows SVM accuracy of 71% while NB had an accuracy of 67%. I initially assumed that NB will not out-perform SVM due to its underlining assumption of independent features.

Improvement

Additional work can be done to collect more data and find a way to detect multiple programming languages within the same question. I think the final solution can be set

as the new benchmark for future work. Especially if more advanced algorithms such as neural network can be used.

References

https://data.stackexchange.com/stackoverflow/query/edit/1092843

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://en.wikipedia.org/wiki/Precision_and_recall

https://en.wikipedia.org/wiki/F1_score