

Stackoverflow Dataset Report



Hans Walter
20180006
Feb.15, 2021

Introduction

Stackoverflow is a free to use, QA website where people can interact with fellow programmers and post their questions as well as answers. Therefore, the website is able to capture large amounts of data where we can create our own questions and as data scientists, be able to respond to them. For this specific instance, I decided to use machine learning in order to classify what programming language was being asked for, based on the related info about the question.

My main questions through this project were:

- Is there a correlation between for example, the length of the question and the various other parameters, to the framework that is being requested support for?
- How does Stackoverflow organize their data?
- What size of data are we dealing with?
- Is there room for improvement, for capturing data, organizing it, etc.

01 Data

GOOGLE BIG QUERY

The dataset was extracted using the Google Big Query platform. Due to the size of the data, it was not possible to provide the dataset through Kaggle directly. Therefore, the user could extract whatever data it needs through his/her own personalized queries.

The following tables were available in the database:

- badges
- comments
- post_history
- post_links
- posts_answers
- posts_moderator_nomination
- posts_orphaned_tag_wiki
- posts_privilege_wiki
- posts_questions
- posts_tag_wiki
- posts_tag_wiki_excerpt
- posts_wiki_placeholder
- stackoverflow_posts
- tags
- users
- votes


In our case we used posts_questions and posts_answers.

02

Data and Methods

QUERY

Query completed in 3.579 sec
7:33 PM

 Open query in editor

```
1 #standardSQL
2 SELECT
3   character_length(table1.title) as title_length, character_length(table1.body) AS question_length, table1.accepted_answer_id,
4   table1.creation_date as question_date, table1.tags, char_length(table1.tags) as tags_length,
5   table1.view_count, table2.creation_date as answer_date
6 FROM
7   `bigquery-public-data.stackoverflow.posts_questions` table1
8 LEFT JOIN `bigquery-public-data.stackoverflow.posts_answers` table2
9 ON (
10    table1.accepted_answer_id = table2.id
11 )
12 WHERE table1.post_type_id = 1 AND table2.post_type_id = 2
13 LIMIT 16000
```

| | |
|--------|---|
| Job ID | mlm-stackoverflow-ufm:US.bquxjob_3f754242_177a878aeaa |
| User | hansrwc734@gmail.com |

The importance of the query is that if done properly, it can save us time afterwards in organizing and cleaning data. The most important part of the query, was to be able to join both tables by id, which in this case table1(questions) connects to table2(answers) by answer_id. The question text wasn't downloaded because the download size became too large, due to the variant text lengths of each question. We gathered the title length, the question date, the answer date, the tags length and the view count on the corresponding questions.

With our dataset downloaded, we can start working directly with jupyter lab.

Methods

03

INITIAL ANALYSIS

It was important to check for missing values, as well to check how the data in each column behaved. It was surprising to find no missing data at all in the dataset, this demonstrates the good practices Stackoverflow enforces when the user wants to upload a question, ensuring no fields are left blank and perhaps ensuring a text length minimum.

```
so_df.isnull().sum()
```

```
title_length      0
question_length   0
accepted_answer_id 0
question_date     0
tags              0
tags_length       0
view_count        0
answer_date       0
dtype: int64
```

```
so_df.head()
```

| | title_length | question_length | accepted_answer_id | question_date | tags | tags_length | view_count | answer_date |
|---|--------------|-----------------|--------------------|-----------------------------|---|-------------|------------|-----------------------------|
| 0 | 53 | 2574 | 37738104 | 2016-06-09 20:10:52.550 UTC | google-analytics drupal-7 google-tag-manager u... | 80 | 522 | 2016-06-10 00:13:02.770 UTC |
| 1 | 27 | 535 | 37442587 | 2016-05-25 15:36:19.577 UTC | sql-server ssis enterprise sql-server-data-too... | 56 | 42 | 2016-05-25 16:12:18.980 UTC |
| 2 | 87 | 1946 | 37550734 | 2016-05-31 15:36:47.460 UTC | r | 1 | 56 | 2016-05-31 16:10:38.387 UTC |
| 3 | 53 | 418 | 37768366 | 2016-06-11 17:38:54.733 UTC | api oauth google-analytics google-analytics-ap... | 73 | 1361 | 2016-06-11 20:46:14.160 UTC |
| 4 | 57 | 1250 | 37594489 | 2016-05-24 09:32:21.660 UTC | permissions file-permissions tomcat8 permisio... | 77 | 2908 | 2016-06-02 14:17:18.903 UTC |

DAYS DIFFERENCE

In order to work with the question_date and answer_date, I converted the columns to date type variables so the day difference could be calculated.

```
so_df[['answer_date', 'question_date']] = so_df[['answer_date', 'question_date']].apply(pd.to_datetime)
so_df['hours_elapsed'] = (so_df['answer_date'] - so_df['question_date']).astype('timedelta64[h]')
```

| | title_length | question_length | accepted_answer_id | question_date | tags | tags_length | view_count | answer_date | hours_elapsed |
|---|--------------|-----------------|--------------------|----------------------------------|---|-------------|------------|----------------------------------|---------------|
| 0 | 53 | 2574 | 37738104 | 2016-06-09 20:10:52.550000+00:00 | google-analytics drupal-7 google-tag-manager u... | 80 | 522 | 2016-06-10 00:13:02.770000+00:00 | 4.0 |
| 1 | 27 | 535 | 37442587 | 2016-05-25 15:36:19.577000+00:00 | sql-server ssis enterprise sql-server-data-too... | 56 | 42 | 2016-05-25 16:12:18.980000+00:00 | 0.0 |
| 2 | 87 | 1946 | 37550734 | 2016-05-31 15:36:47.460000+00:00 | r | 1 | 56 | 2016-05-31 16:10:38.387000+00:00 | 0.0 |
| 3 | 53 | 418 | 37768366 | 2016-06-11 17:38:54.733000+00:00 | api oauth google-analytics google-analytics-ap... | 73 | 1361 | 2016-06-11 20:46:14.160000+00:00 | 3.0 |
| 4 | 57 | 1250 | 37594489 | 2016-05-24 09:32:21.660000+00:00 | permissions file-permissions tomcat8 permisio... | 77 | 2908 | 2016-06-02 14:17:18.903000+00:00 | 220.0 |

Methods

04

LABELS

Due to the numerous tags and frameworks that are present in the data, a filter was applied in order to obtain only the 10 most frequently asked about programming languages. Next, a simple label encoder was applied so it could be utilized afterwards for the classifier model. The encoder labels from 0 to n, depending on the size of the label dictionary.

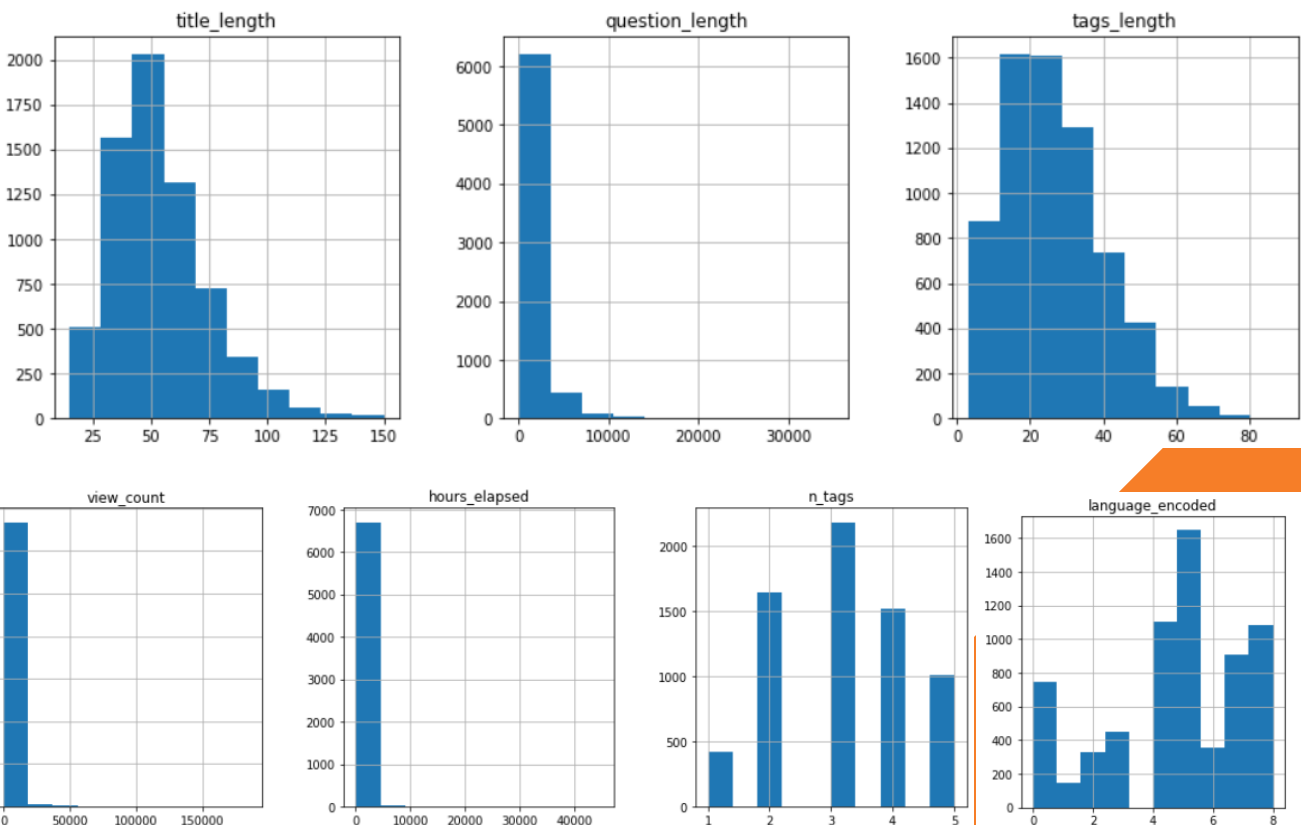
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
language_vector = clean_df['language']
clean_df['language_encoded'] = encoder.fit_transform(language_vector)

print(encoder.classes_)

['android' 'css' 'html' 'ios' 'java' 'javascript' 'jquery' 'php' 'python']
```

NORMALITY TESTING

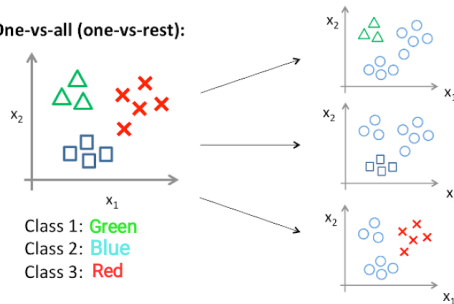
It was important to check previous to building the model, if our data was skewed. In our case, it was and some columns were more skewed than others. This is why a scaler was used in order to normalize the data. Skewed data can be difficult to deal with as it can make the model ineffective.



Methods and Classifiers

05

One-vs-all (one-vs-rest):



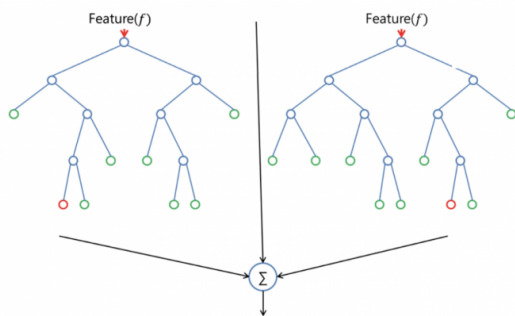
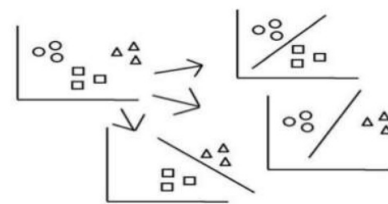
SDG OVA CLASSIFIER

By default the classifier works in a One versus All strategy, because the SDG classifier is used originally in binary classification.

SDG OVO CLASSIFIER

In order to use the One Versus One, we had to import a library of the same name from scikitlearn.

One-vs-One (OVO)

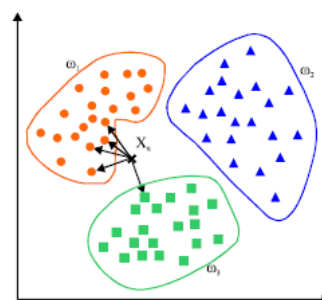


RANDOM FOREST CLASSIFIER

The classifier works with various decision trees, the features are then passed into each tree so that . The algorithm then searches for the label with "more votes" and provides it as the result.

KNN CLASSIFIER

The classifier works with various decision trees, the features are then passed into each tree so that . The algorithm then searches for the label with "more votes" and provides it as the result.



Training and Parameters

06

TRAINING AND TEST DATA

Splitting Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

The default splitting data process divides the dataset into 70% training and 30% testing. The `random_state` is simply used for reproducibility purposes.

N ESTIMATORS

Number of trees *Random Forest Classifier*

BOOTSTRAP

Random sampling of the dataset *Random Forest Classifier*

CROSS VALIDATION

Dividing the data into subsets of training and datasets, it can be considered as some sort of epochs also.

SCORING

For this instance we are using accuracy. However, it is not completely reliable. Therefore, we will be utilizing the method below.

PRECISION, RECALL AND F1

Precision: true positives divided by the sum of all true positives and true negatives.

Recall: correct positive predictions, ratio between true positives and all positives.

F1: combines precision and recall to give a general accuracy of the model.

Results

07

SDG OVA CLASSIFIER

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.69 | 0.05 | 0.09 | 216 |
| 1 | 0.00 | 0.00 | 0.00 | 54 |
| 2 | 0.13 | 0.17 | 0.15 | 111 |
| 3 | 0.07 | 0.23 | 0.11 | 137 |
| 4 | 0.17 | 0.13 | 0.15 | 331 |
| 5 | 0.29 | 0.40 | 0.34 | 485 |
| 6 | 0.00 | 0.00 | 0.00 | 105 |
| 7 | 0.21 | 0.30 | 0.25 | 253 |
| 8 | 0.19 | 0.08 | 0.11 | 339 |
| accuracy | | | 0.20 | 2031 |
| macro avg | 0.19 | 0.15 | 0.13 | 2031 |
| weighted avg | 0.24 | 0.20 | 0.18 | 2031 |

SDG OVO CLASSIFIER

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.23 | 0.26 | 0.25 | 216 |
| 1 | 0.00 | 0.00 | 0.00 | 54 |
| 2 | 0.00 | 0.00 | 0.00 | 111 |
| 3 | 0.00 | 0.00 | 0.00 | 137 |
| 4 | 0.24 | 0.10 | 0.14 | 331 |
| 5 | 0.32 | 0.65 | 0.43 | 485 |
| 6 | 0.00 | 0.00 | 0.00 | 105 |
| 7 | 0.24 | 0.52 | 0.33 | 253 |
| 8 | 0.24 | 0.08 | 0.12 | 339 |
| accuracy | | | 0.28 | 2031 |
| macro avg | 0.14 | 0.18 | 0.14 | 2031 |
| weighted avg | 0.21 | 0.28 | 0.21 | 2031 |

RANDOM FOREST CLASSIFIER

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.43 | 0.37 | 0.40 | 216 |
| 1 | 0.29 | 0.04 | 0.07 | 54 |
| 2 | 0.57 | 0.23 | 0.32 | 111 |
| 3 | 0.11 | 0.04 | 0.06 | 137 |
| 4 | 0.30 | 0.33 | 0.31 | 331 |
| 5 | 0.37 | 0.59 | 0.45 | 485 |
| 6 | 0.44 | 0.13 | 0.20 | 105 |
| 7 | 0.32 | 0.38 | 0.35 | 253 |
| 8 | 0.33 | 0.27 | 0.30 | 339 |
| accuracy | | | 0.35 | 2031 |
| macro avg | 0.35 | 0.26 | 0.27 | 2031 |
| weighted avg | 0.35 | 0.35 | 0.33 | 2031 |

KNN CLASSIFIER

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.26 | 0.38 | 0.31 | 216 |
| 1 | 0.06 | 0.04 | 0.04 | 54 |
| 2 | 0.19 | 0.17 | 0.18 | 111 |
| 3 | 0.08 | 0.06 | 0.07 | 137 |
| 4 | 0.23 | 0.24 | 0.23 | 331 |
| 5 | 0.37 | 0.46 | 0.41 | 485 |
| 6 | 0.12 | 0.06 | 0.08 | 105 |
| 7 | 0.25 | 0.23 | 0.24 | 253 |
| 8 | 0.24 | 0.18 | 0.21 | 339 |
| accuracy | | | 0.27 | 2031 |
| macro avg | 0.20 | 0.20 | 0.20 | 2031 |
| weighted avg | 0.25 | 0.27 | 0.25 | 2031 |

08

Conclusions

GENERAL

Even though we had a considerable amount of data to process, that doesn't necessarily mean that it was enough to have a good model. We could observe that to begin with, the data was skewed and that the correlation matrix showed us that there was little to no connection between the features and the label. We can conclude effectively that with our current set of features we cannot classify which framework/programming language is being asked for in the forums. Our Random Forest classifier model theoretically is doing better than guessing(it gave us a 35% accuracy score), by which if we used a guessing method we would have a 10% of getting the right programming language based on the features presented(10 encoded results, which gives a 1/10 of guessing the right one). The results also provide us with the insight that there are various levels of difficulty as well as simplicity within each program. In a way, we could say that there is no real difference between frameworks when it comes to complications, questions, etc. Notice that the original dataset had 16,000 observations, for which it might be possible at the time of filtering, affected in some way the final result.

REFLECTION ON STACKOVERFLOW DATA

The company does an excellent job of storing huge amounts of data (approximately 27 million records) in an organized fashion. However, accessing the data was not very user friendly. The documentation as to how to use Google Big Query or download the corresponding library in Python to do the queries was a bit ambiguous at times. Nevertheless, once you have access, the only limit the user has is the download size at times. I believe the company already has enough information about their users as well as the questions and answers, it's a good historical record for many programming enthusiasts searching for help.

LEARNING CURVE

Part of the process is learning which models to use for each type of problem, as well as to be able to tune them properly to get better results. A combination of continuous learning as well as practice, will ensure a better workflow in the future. Not only to make better models, but also take into account the type of questions we are asking ourselves and if they are the right ones to make.

References

- Dataset from <https://www.kaggle.com/stackoverflow/stackoverflow>
- F-score. (2019, May 17). Retrieved February 18, 2021, from <https://deepai.org/machine-learning-glossary-and-terms/f-score#:~:text=The%20F%2Dscore%2C%20also%20called,positive'%20or%20'negative'.&text=It%20is%20possible%20to%20adjust,recall%2C%20or%20vice%2Dversa.>
- Multiclass classification & Cross Validation - Machine Learning with TensorFlow & scikit-learn video from <https://www.youtube.com/watch?v=5KyH6v8oKNQ&t=1423s>
- Purva Huilgol Trainee Data Scientist at Analytics Vidhya. Pursuing Masters in Data Science from the University of Mumbai. (2020, December 29). Precision vs recall: Precision and recall machine learning. Retrieved February 18, 2021, from <https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/>
- Sklearn.neighbors.kneighborsclassifier¶. (n.d.). Retrieved February 18, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
- Sklearn.ensemble.randomforestclassifier¶. (n.d.). Retrieved February 18, 2021, from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- Sklearn.linear_model.sgdclassifier¶. (n.d.). Retrieved February 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
- Stochastic Gradient Descent, Clearly Explained!!! video from <https://www.youtube.com/watch?v=vMh0zPT0tLI&t=320s>
- *NOTE: I referenced and was guided by the jupyter notebooks provided by Christian Medina to support my work.