Predicting Tags for Stackoverflow data

Sai Rahul Ponana | Sriram Hariharan Neelakantan | Chamanthi Aki

1. Abstract

Stack Overflow is a platform where people from disparate fields can drop their questions or problems about various domains of work as well as solve/answer the questions and queries. We obtained 10% of the Data from the Stack Overflow website with random questions, answers and the tags or domains of the field to which they belong. Based on this data we tried to predict the tags of the field of study for the new questions which are asked every day on the website and also predict the time taken by any user to answer the questions posted.We implemented various Machine Learning models and evaluated various metrics on them to find the best model.

2. Statement of contribution

- ➔ **Chamanti Aki** - Data Importing, data cleaning, data preprocessing, Tokenization, Lemmatisation, modelling the data using algorithms from the sklearn library, evaluation and hyper parameter tuning. Editing and adding content to the report and presentation.
- ➔ **Sriram Hariharan Neelakantan** - Data cleaning,Data preprocessing using NLTK package, encoding, modeling the data using algorithms from the sklearn library, evaluation and hyper parameter tuning.Editing and adding content to report and presentation.
- ➔ **Sai Rahul Ponnana** - Data cleaning, encoding, preprocessing methods like feature scaling, extraction etc. and worked on the regression analysis and model prediction.Editing and adding content to report and presentation.

3. Introduction

Stack Overflow serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down similar to Reddit and edit questions and answers in a fashion similar to a wiki.

Users of Stack Overflow can earn reputation points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up"vote on a question or an answer to a question, and can receive badges for their valued contributions, which represents a gamification of the traditional Q&A website. Users unlock new privileges with an increase in reputation like the ability to vote, comment, and even edit other people's posts.

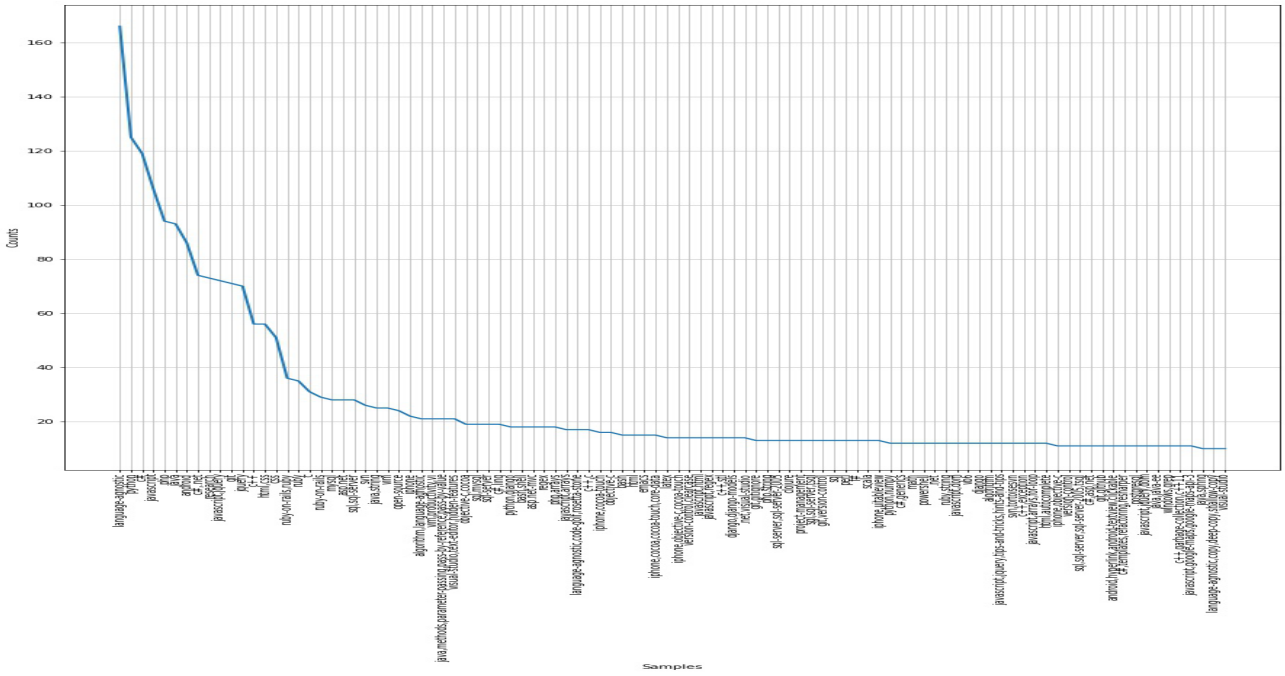
4. Feature Extraction and Preprocessing

- A peek into the Dataset:

The dataset is obtained from the Kaggle website. The dataset is obtained from the backend database of the website, so evidently the data is stored in the form of tables.

The dataset consists of only 10% of Questions and Answers and tags which are manually created from the Stack Overflow website and are organized as three tables namely:
questions.csv, answers.csv with 1.1 million rows of each and tags.csv

Fig.1. Frequency plot of tags:



The above plot shows the count of each tag for the top 100 common tags.

After eliminating unwanted columns and merging the three tables, the data table has the following columns:

• **Q_Id** - Double

• **Q_Score** - Integer

• **Title -** String

• **Question -** Sentences with HTML tags

• **A_Id** - Double

• **A_Score -** Integer

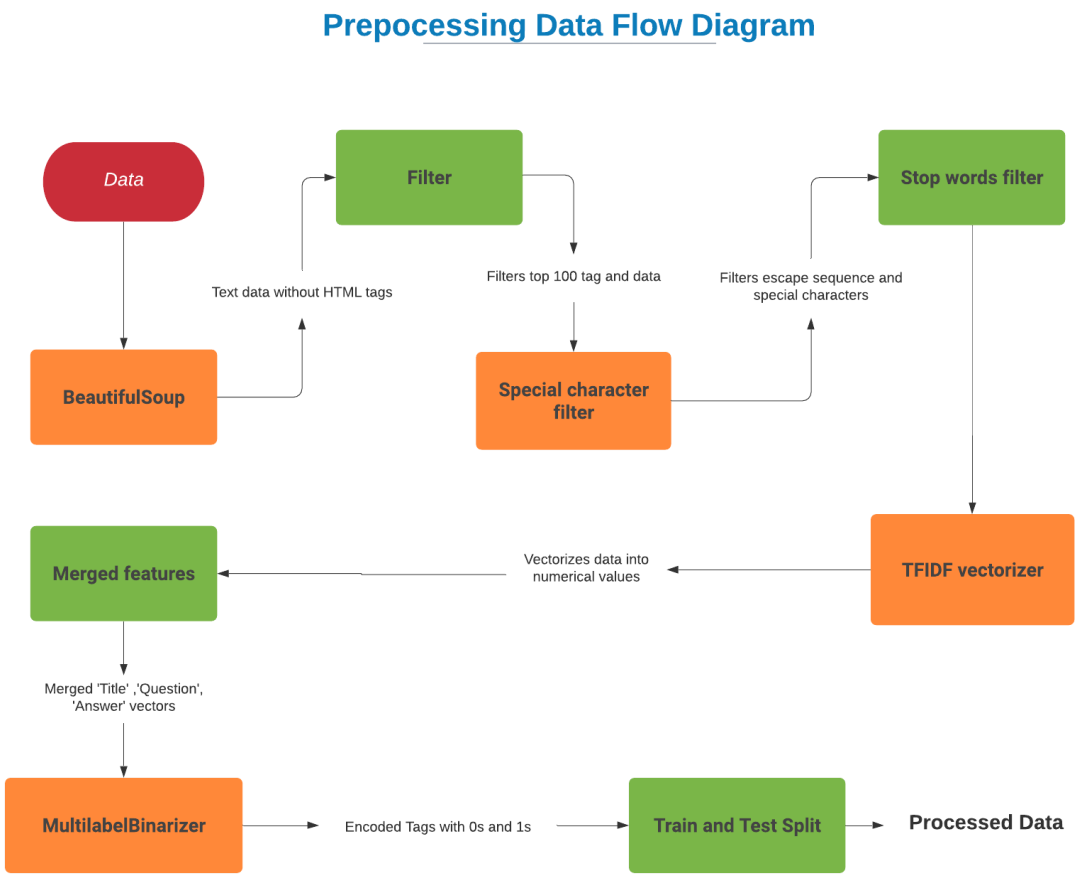
• **Answers -** Sentences with HTML tags

• **Tag** - list of strings

- Using NLTK Package:

We use the NLTK package to process data, transform data to fit various ML models to it. Using **BeautifulSoup** library to remove HTML tags from Questions, Answers and Title. Then, extract the top 100 most common tags and filter data with such tags. After filtering based on most occurring tags, we apply tokenizer and lemmatizer on the sentences to derive tokens of each word and to derive core words of every word. Removing escape sequence characters and special characters. Then remove stop words (‘a’,’the’,’is’,etc). **Using TFIDF vectorizer**, converted words into a set of numerical features (1000 assumed). Merged TFIDF vectors of Title,Questions and Answers. The **MultilabelBinarizer** library helps to change the target value to 0s and 1s. Then formed a Train and Test split.

Fig.2. Preprocessing (using NLTK package) Data Flow Diagram



For the regression problem,

- Pre-processing Steps to Predict Response Time:
  - Data Cleaning
  - Feature Addition
  - Label Encoding
- Data Cleaning: The following columns are used in addition to the columns used for the classification problem.
  - **Q_Creationdate** - Question Creation date

    - **A_Creationdate** - Answer Creationdate
- Feature Addition: New features/columns have been added:
    - **weekend** - if the question was asked on the weekend, we defined the column 1 or else 0
    - **day_half** - if the question was asked on the first half of the day, we defined the column 1 or else 0
    - New target column **Y** - Difference between close date and creation date.
- Label Encoding and Feature Scaling:
  Tags in the dataset have a string datatype, so we need to group them into numbers with LabelEncoder(). The numbers which we encoded are scaled in between a particular range of numbers. There are many scalers, we used MinMaxScaler() which has a range of 0 to 1.

5. Models:

Machine Learning Algorithms used in the Project:

- To predict the tags of the questions.
    - Logistic Regression
    - Stochastic Gradient Descent Classifier
    - Random Forest Classifier
    - Decision Trees Classifier
- Linear Regression – A simple linear regression and polynomial regression algorithm was used to predict the response time required to answer a question.

Building a model:

A. Classification Problem :
    a. We have split the pre-processed data into a train-test subsets with a proportion of 70%-30%.
    b. Used the train data directly to fit various models mentioned above.
    c. Predicted target labels for the test data subset to evaluate the models.
B. Regression Problem:
    a. The dataset from the classification problem contains many features and we need to eliminate the useless features for training in the case

of the Linear Regression. We take the regression summary of the whole data, eliminate the columns or select them according to the P-values. In this problem, we selected the columns for which the p-value is less than or equal to 0.05.

b. We split the whole data into train and test data with 30% test and 70% train data. Then, we build a linear regression model to the extracted data and features from the previous steps with LinearRegression() function.Train the training Data to fit the model and Predict the values using the predict() function in linear regression.

6. Experiments and Evaluation (Hyperparameter tuning and model evaluation strategies)

Logistic Regression:

- Parameters used are:
  - Number of maximum iterations : Values = [10,20,50,80,100]
- The model gives the same results for all values of maximum iterations.
- The metric values:
  - Accuracy: 37.7%
  - Recall: 0.99
  - Precision: 0.789
  - F1 score: 0.87
  - Log Loss error: 115
- By analysing the metrics we can conclude that Logistic Regression is not a good model. So, we implemented a Stochastic Gradient Descent Classifier.

Stochastic Gradient Descent Classifier:

Parameters used are:

- Regularisation parameter(alpha): Values = [0.001,0.01,0.1]
- Maximum Iterations : [100,200,300,500]

Fig.3. Best Metric values for SGDClassifier (Best hyper-parameter values)

```
Regularization parameter: 0.001, Maximum iterations: 500
------------------------------>
Clf:  SGDClassifier
Accuracy : 0.5553470919324578
Precision: 0.8394870116352685
Recall value : 0.9858534217849775
F1 score : 0.902860875644995
Log Loss : 77.82163336580606
---
```

Fig.4. Worst Metric values for SGDClassifier(Worst hyper-parameter values)

```
Regularization parameter: 0.001, Maximum iterations: 200
------------------------------>
Clf:  SGDClassifier
Accuracy : 0.4896810506566604
Precision: 0.7516036622165863
Recall value : 0.9953716366636598
F1 score : 0.8412341583533303
Log Loss : 100.15386008194461
---
```

The images show the best and worst combination of the hyper parameters that yield the best and worst model respectively. SGD Classifier performs better than simple Logistic Regression, but not much improvement. So, we implemented Decision Trees and Random Forests for better predictions.

Decision Tree Classifier:

No.of Features, Minimum split	2000,2	2000,3	2000,4	3000,2	3000,3	3000,4
Accuracy	0.72	0.71	0.739	0.737	0.737	0.726
Precision	0.87	0.86	0.88	0.876	0.89	0.87
Recall	0.92	0.91	0.89	0.90	0.91	0.89
F1 score	0.89	0.88	0.88	0.88	0.90	0.88
Log loss	67.98	65.8	64.7	64.9	63.3	66.9

- The best metrics :
    - Accuracy - 0.737
    - Precision - 0.89
    - Recall - 0.91
    - F1 score - 0.9
    - Log loss - 63.3
- Best hyper parameter values for best metrics obtained are as follows:
    - No.of Features - 3000
    - Minimum split - 3

Random Forest:

- We consider three hyper parameter here-
    - Number of features - [2000,3000]
    - Minimum split - [2,3,4]
    - Number of estimators - [300,400]
- The best combination of hyper parameters we got was:
    - Number of Feature - 3000
    - Minimum split- 2
    - Number of estimators - 400
- The metric values for the best combination of hyper parameters was
    - Accuracy - 0.752
    - Precision - 0.91
    - Recall - 0.89
    - F1 score - 0.899
    - Log loss - 61.3

- The least preferred model with **Number of Feature** - 2000, **Minimum split**- 4, **Number of estimators** - 300 has :
    - Accuracy - 0.71
    - Precision - 0.86
    - Recall - 0.87
    - F1 score - 0.864
    - Log loss - 74.7

8. Results and Conclusion

From the above results for various combinations of hyperparameters, we observed that decision tree classifiers and random forests are the best performers to classify the data into tags. However, this model is not the ultimatum of predicting the values, there are many alternatives and future scope one can work on to get more accurate and better models. We converted the timedelta to minutes, so trained and printed the outcome in minutes, or simply this is the minimum predicted response time in minutes for a particular question to be answered.

Github link : https://github.com/RAHULPONNANA/SML_Project/blob/main

9. Future Scope

We can implement Naive-Bayes and SVM classifiers also to compare the model to other methods. We can also implement feature reduction using PCA, LDA. For the response time prediction, we just used the most frequent and most specific tags, instead of which we can also use the activity specificity and frequency.

Sometimes the user must wait for longer than predicted, so it can be helpful if we predict the response time even before putting the question and using some advanced machine learning techniques, to reduce the error and increase the accuracy.

10. References

https://stackoverflow.com/
https://scikit-learn.org/stable/
https://www.kaggle.com/
https://towardsdatascience.com/
https://medium.com/