SOQDE: A Supervised Learning based Question Difficulty Estimation Model for Stack Overflow

Sk. Adnan Hassan*, Dipto Das†, Anindya Iqbal*, Amiangshu Bosu‡, Rifat Shahriyar* and Toufique Ahmed*
* Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
† Department of Computer Science
Missouri State University, Springfield, MO, USA
‡ Department of Computer Science
Wayne State University, Detroit, MI, USA
Email: adnan20049@gmail.com, dipto.cse.buet@gmail.com, anindya@cse.buet.ac.bd,
amiangshu.bosu@wayne.edu, rifat@cse.buet.ac.bd, toufiqueahmed@cse.buet.ac.bd

***Abstract*—StackOverflow (SO), the most popular community Q&A site rewards answerers with reputation scores to encourage answers from volunteer participants. However, irrespective of the difficulty of a question, the contributor of an accepted answer is awarded with the same 'reputation' score, which may demotivate an user's additional efforts to answer a difficult question. To facilitate a question difficulty aware rewarding system, this study proposes *SOQDE* (Stack Overflow Question Difficulty Estimation), a supervised learning based Question difficulty estimation model for the StackOverflow. To design SOQDE, we randomly selected 936 questions from a SO datadump exported during September 2017. Two of the authors independently labeled those questions into three categories (basic, intermediate, or advanced), where conflicting labels were resolved through tie-breaking votes from a third author. We performed an empirical study to determine how the difficulty of a question impacts its outcomes, such as number of votes, resolution time, and number of votes. Our results suggest that the answers of a basic question receive more votes and therefore would generate more reputation points for an answerer. Due to less incentives relative to efforts spent by an answerer, intermediate and advanced questions encounter significantly more delays than the basic questions, which further validates the need of a model like SOQDE. To build our model, we have identified textual and contextual features of a question and divided them into two categories—pre-hoc and post-hoc features. We observed a model based on *Random Forest* achieving the highest mean accuracy (67.6%), using only answer-independent pre-hoc features. Accommodating answer-dependent post-hoc features, we were able to improve the mean accuracy of our model to 75.2%.**

***Index Terms*—StackOverflow, Prediction model, Question Difficulty, Reputation**

I. Introduction

Stack Overflow (SO) is the most popular community Q/A site among the programmers. Since more than 92% of the questions are answered in a median time of 11 minutes [1], SO has become the go-to place for the programmers to find solutions for technical difficulties. Programmers irrespective of their expertise levels, from novice learner to expert professionals, participate in constructive discussions to generate one or more solutions for each posted issue. The user posting a question can mark only one solution as "accepted" . Apart from the questioner, other registered users of the SO can upvote or downvote a question or answer to indicate its quality (e.g., research effort, clarity, and usefulness) so that clear and useful answers receive more upvotes and vice-versa. However, SO awards the same number of reputation points (i.e., 10) for each accepted answer irrespective of the difficulty level of the question, although an 'Advanced' question may require significantly more efforts from an answerer than a 'Basic' question. Since the number of beginner or intermediate level programmers is larger than the number of experts, a 'Basic' question is likely to receive more views than an 'Advanced' question. Hence, the answer to a 'Basic' question may receive more upvotes, points and consequently yield more reputation scores to the answerer. On the other hand the occurrence of a difficult issue may be less frequent among the programmers to generate lower number of views than a basic question. Therefore, the current reputation model of SO is biased towards rewarding the answerer of a basic question with more points, although the solution of a basic question may require significantly less efforts than the difficult ones. Hence, the answerers of SO may find less incentives to spend the extra efforts to answer difficult questions, which may eventually cause many difficult questions remaining unanswered. Moreover, SO awards various levels of privileges to its users based on their reputations and badges. Since the users gains access to various SO community services based on their reputation scores [1], a flat awarding system that ignores the difficulty level of the questions may prevent expert users focusing on difficulty questions from accessing those services.

To reduce its bias towards the easy questions, SO may introduce variable rewards to the answers based on the difficulty level of a question. This model may motivate SO users to spend potential extra efforts to answer difficult questions. Moreover, SO may introduce new badges or privileges to motivate the users who answer difficult questions. Such a reward model may not only reduce the resolution times for difficult questions but also reduce the number of unanswered questions. However, a this variable reward model would re-

[1] https://stackoverflow.com/help/whats-reputation

978-1-7281-1970-0/18/$31.00 ©2018 IEEE
DOI 10.1109/APSEC.2018.00059

quire a reliable question difficulty estimation (QDE) technique, which currently do not exist. Such a QDE model may be also helpful to route difficult questions to expert users (i.e., reputation point) to reduce resolution intervals for difficult questions.

Prior research on SO have investigated various areas such as: answer quality estimation, user expertise or rating measurement, and question routing. [2], [3]. However, the problem of QDE i.e., categorizing SO questions based on its' difficulty levels has received little attention. Liu *et* al. [4] was the first to propose a competition-based model to estimate the difficulty of a question by exploiting pairwise comparisons of users and questions. A subsequent work by Wang *et* al. [5] improved the Liu *et* al.'s model by incorporating the textual descriptions of the questions. However the dependence on the answers to predict the difficulty of a question limit the applicability of those two models [4], [5]. Hence, this research aims to develop, *SOQDE* (**S**tack**O**verflow **Q**uestion **D**ifficulty **E**stimation), a reliable model to automatically determine the difficulty level of SO questions. This model will ensure well-deserved additional reputation scores to the users who ask or answer difficult questions and reduce the occurrences of their extra efforts going unnoticed. We aim to develop two supervised learning based models: i) a pre-hoc model that does not depend on the characteristics of the answers, and ii) a post-hoc model that leverages the answers of a question to improve its accuracies over a pre-hoc model.

On this goal, we manually labeled randomly selected 936 SO questions on Java String, Inheritance and Multi-threading. Two of the authors independently labeled each of the questions as 'Basic', 'Intermediate' or 'Advanced' based on a pre-defined rubrics. We measured inter-rater reliablitty of the manual labeling process with Cohen's Kappa, which was measured as $\kappa = 0.73$. Conflicting labels were later resolved by a tie-breaking vote from a third author. We classified features selected from each question into two categories.

- **Pre-hoc:** Features those are available immediately after a question has been posted on the SO.
- **Post-hoc:** Features those are available only after a accepted solution has been posted for a SO question.

We uses this dataset to train and validate supervised learning models using 10 popular algorithms. The best performing model (using Random Forest) during our 10-fold cross validations achieves 67.1% mean accuracy with pre-hoc features and 75.2% mean accuracy with both pre-hoc and post-hoc features. To encourage replication of this study, we have made the dataset used in this study publicly available at: [2].

In summary, the primary contributions of this paper include:

- We have manually labeled an validated the difficulty levels of of 936 SO questions.
- We have empirically investigated how the difficulty levels of SO questions may impact its' resolution times, view counts, and number of votes.
- We have applied information-gain based feature selection models to identify three pre-hoc and seven post-hoc features that are useful to estimate the difficulties of SO questions.
- We have developed SOQDE, a supervised learning based model that achieves satisfactory performance with only pre-hoc features and is improved further with post-hoc features. Although the pre-hoc model is less accurate than the post-hoc model, it can be used as soon as a question has been posted on the SO. Our SOQDE model can be useful not only to introduce a variable reward system based on question difficulty but also assist the routing of a question based on its difficulty level.
- We have empirically investigated 10 popular supervised learning algorithms that may be useful for the QDE context.

The remained of this paper is organized as following. Section II provides an empirical investigation of the difficulties of SO questions. Section III details our research methodology. Section IV evaluates the performance of different models on our training dataset. Section V, discusses the implications of this study. Section VI analyzes the threats to validity of our study. Section VII, provides an overview of the related prior research on SO. Finally, Section VIII concludes the paper.

II. Empirical Investigation of Questions' Difficulties

In this section, we discuss our approaches for dataset selection, manual labeling process and an empirical investigation of the difficulties of SO questions.

A. Harnessing Stack Overflow Data

In this study, we selected SO questions on Java, since it is one of the most popular programming languages as of 2018. According to a survey [6] conducted by Stack Overflow, 41.6% back-end developers and 54.2% mobile developers prefer Java making it as one of the top three languages used by programmers. Java is also the most popular language among students, with 51.1% of the students using it. It is also the second most popular technology on Stack Overflow. In this study, we randomly selected, total 936 SO questions (i.e., 312 questions from each of the following on three Java topics) *Strings*, *Threads*, and *Inheritance*). We selected these three topics since they are among the most popular Java topics on SO.

For the selection process, we used the SO data dump [7] published during September 2017. We imported the data dump in to a MySQL database to facilitate an user friendly query interface. Although, we only considered questions regarding three popular Java (i.e., Strings, Threads and Inheritance), our dataset generation methodology followed in our work should work for any topics from SO.

B. Training Set Generation

To manually label the questions, we had discussion sessions to develop a rubrics to rate the difficulty of each questions into one of the following three categories:

[2]https://goo.gl/rgYp9i

1) **Basic:** A *Basic* question can be answered with the help of beginner level books or basic API documentation. Basic questions also include the simple problem-solving questions and comparison of basic functions of two different languages. We considered it from the viewpoint of learners as they usually start with Basic questions and try to solve them going through the text/reference books and/or API documentation.
2) **Intermediate:** An *Intermediate* question requires a relatively deeper understanding of the language to answer. When a beginner goes through all the problems of Basic questions, the next thing that will come to his/her mind is how to solve those problems efficiently considering the time complexity or memory usage of different.
3) **Advanced:** An *Advanced* question usually deals with hard problems where the solution needs in-depth programming knowledge or logical thinking. We also considered questions that require the understanding of internal language structure or compiler as advanced.

Table I provides an overview of our rubrics and example questions for each of the three categories. Using these rules as guidelines, authors Adnan and Dipto independently labeled each of the 936 selected questions. During this labeling process, they rated each question solely based its title and description. we used Cohen's kappa coefficient [8] to measure the inter-rater agreement with is $\kappa = 0.738$, indicating a substantial agreement[3]. The disagreements between the two coders was resolved using a tie-breaking vote from the author Anindya. Based on these steps, we labeled the 936 selected questions to generate our training dataset. Among these 936 questions, 598 (63.88%) questions were labeled as *basics*, 244 (26.06%) questions were *intermediates*, and the remaining 94 (10.04%) questions were *advanced*.

C. Empirical Investigation

Figure 1 shows the average number of votes received by questions from various difficulties. These results indicate that accepted answers from both *Basic* level and *Intermediate* level questions received significantly higher votes than the answers of *Advanced* level questions. It further validates our point that users have less motivation to take the toil of answering *Advanced* level questions. Hence, *Advanced* level questions are likely to take longer times to get an acceptable answer. Figure 2 supports that claim as the average resolution time for questions grows with its' difficulty level, especially from *Intermediate* to *Advanced* level of questions. On average, Advanced questions require 10 times longer than the *Intermediates* to get an answer.

Similarly, Figure 3 shows that *Basic* and *Intermediate* questions generate significantly higher number of views than the *Advanced* questions. Therefore, even the non-accepted answers of the *Basic* questions get more attentions as well as generate higher reputation scores for an answerer. As a result, answering Basic questions not only saves a lot of efforts from a answerer, but also those questions provide higher chances of generating reputation scores on the SO. These results justifies the motivation of designing a question difficulty aware rewarding system for the SO and further validates the necessity of a QDE model.

[3]Landis and Koch [9] suggest interpreting Kappa values < 0 as no agreement, 0-0.20 as slight, 0.21-0.40 as fair, 0.41-0.60 as moderate, 0.61-0.80 as substantial, and 0.81-1.0 as almost perfect agreement.

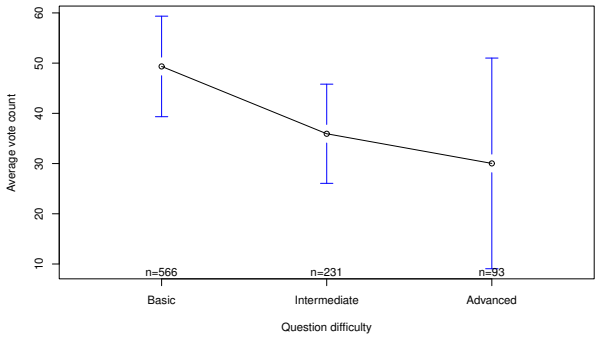

Fig. 1. Question Difficulty vs. Vote for Accepted answers

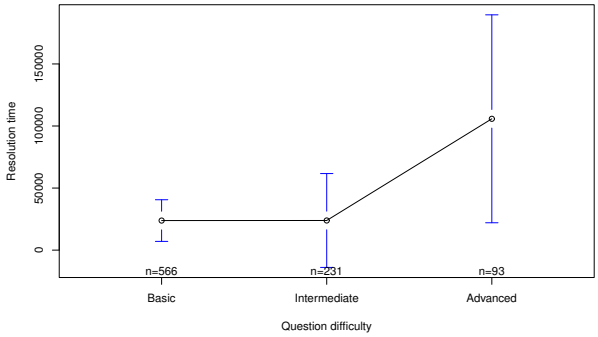

Fig. 2. Question Difficulty vs. Resolution time

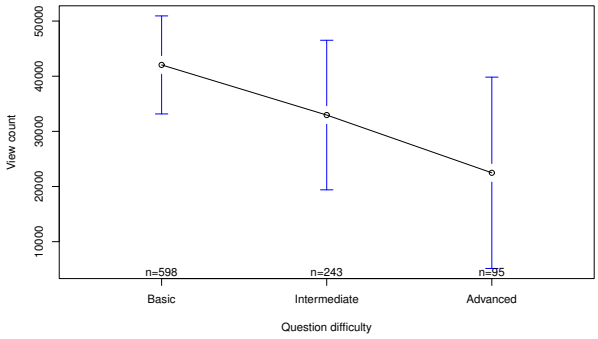

Fig. 3. Question Difficulty vs. Number of Views

TABLE I
CHARACTERISTICS OF QUESTIONS OF DIFFERENT CATEGORIES

Category	Rules	Examples SO Questions
Basic	Questions that can be answered with simple built-in functions/API documentation/beginner level books.	Adding a space after a specific character in a string
	Questions with comparison between constructs/functions of two languages (for better understanding of the language or for learning a new language)	Get integer difference between string just like strcmp
	Questions with simple problem-solving.	String Merge Sort Implementation
Intermediate	Questions that require a relatively deeper understanding of the language to answer, for example Why type questions.	(Resolved) Why are two threads calculating slower than one?
	Questions where the questioner knows about the answer/solution of the question/problem but wants to know more efficient answer/solution.	Algorithm to find the length of longest suffix between a String and a prefix of the String
	Questions related to time complexity, memory usage or other different resource usages of a system/solution.	Finding the Number of Times an Expression Occurs in a String Continuously and Non Continuously
	Questions that require answers with conceptual reasoning/underlying philosophy of any programming construct/API or syntax/design principle.	@Autowired doesn't work if applied to a bean shared between two threads
Advanced	Questions that deal with hard/critical problems where solution needs in-depth programming knowledge or conceptual/logical thinking.	Java - Multiple selectors in multiple threads for nonblocking sockets
	Questions that require advanced in-depth knowledge of internal language structure.	Default method returns true for a while, and then returns false? (Possible JVM bug)
	Questions that deals with infrequently/rarely used framework/API.	Determining which locks are most contended?

III. METHODOLOGY

To estimate the difficulty level of SO questions, we preprocessed our training dataset and selected various textual and contextual features of each question in our dataset. In this section, we briefly describe the preprocessing and feature selection processes.

A. Preprocessing

1) Tokenization and stemming: We separated the body of the questions. We used a tokenizer to parse each text into a list of words. Finally, we applied the Snowball Stemmer [10] to convert each word to its stem.

2) Stop-word removal: Many stop-words (usually non-semantic words such as articles, prepositions, conjunctions, and pronouns) do not play a significant role to express the difficulty level of a question. Popular natural language processing tools such as NLTK [11] and Stanford CoreNLP [12] provide lists of stop-words. We leverage the stop-word list from CoreNLP to exclude stopwords from our dataset.

3) Code snippet removal: Many of the SO questions have code snippets in their body. Code snippets usually consists of identifiers and keywords. The users do not often use the same words to represent the same thing. This is known as the vocabulary problem [13]. For this reason, SOQDE removes these code snippets. We parsed the body of the questions and removed all the code snippets with $< code >$ tag. We did not require any preprocessing for punctuation marks as the tokenization step excludes those.

4) Feature vector generation: We computed TF-IDF (Term Frequency - Inverse Document Frequency) [14] to extract the features for classification. To reduce potential overfitting, we excluded words that are present in less than three questions. We used WEKA [4] to generate feature vectors.

B. Feature Selection

After generating the training dataset, we derive some features to be used in the supervised learning process. To avoid overfitting we turned continuous variable features such as fastest response time, median response time, and question body into categorical ones and put those into four bins or intervals: low, medium, high, very high. Following the guidelines to select features offered by Yang *et* al. [15] and Joachims *et* al. [16], we excluded features that are either redundant or irrelevant and can thus be removed without incurring a significant loss of information [17]. This feature selection method also helps reducing over-fitting and memory requirement of the classifier. Moreover, it will give an estimation of the importance of each feature for our model. We used information gain [15] measure to rank the features. Information gain is a measure of the reduction in entropy of the class variable (Labeling) after the value for the feature is observed and is widely used for feature selection.

Table II presents the features we used, their nature with respect to received answers, ranking according to information gain, and explanations why it may be useful to predict question difficulty. To select features, we used the select attribute [18], [19] feature from WEKA.

[4]http://www.cs.waikato.ac.nz/ml/weka/

TABLE II
LIST OF FEATURES WITH A RATIONALE FOR EACH FEATURE'S INCLUSION

Feature Name	**Pre-hoc or Post-hoc**	**Rank**	**Rationale**
Classification Result with tf-idf	Pre-hoc	1	Words in the document with a high tf-idf score occur frequently in the document and provide the most information about that specific document.
Question Body Size	Pre-hoc	4	Basic questions can be very straight-forward and concise. On the other hand, Intermediate and Advanced questions need description of the background knowledge that the question asker has and the context of the problem. This characteristics may come helpful in identifying question difficulty.
Reputation of the Questioner	Pre-hoc	8	We can expect that expert users, i.e. the users with high reputation are likely to ask Advanced questions whereas programmers of a less expertise and reputation are probable to ask Basic or Intermediate questions.
Fastest Response time to the Question	Post-hoc	2	If a question is basic it is likely to receive the first answer to it very quickly whereas if a question is Advanced one it is likely to receive its first answer after a longer time comparing with the basic questions.
Median Response time to the question	Post-hoc	3	Questions with fastest answer may favor those questions which serve a small purpose that last for a short period of time. The questions that represent a longer period of discussion with a good number of answers as responses and a reasonable median time among those answers indicate their insightfulness which is helpful to identify the Intermediate and Advanced questions.
Favorite Count of the question	Post-hoc	5	Favorite counts number shown on a question indicates how many community users marked it as 'favorite' under their profile that lets one find it easily later.
Score of the Question	Post-hoc	6	The score of the question implies the difference between upvotes and downvotes of a question indicating the extent of research effort, clarity and usefulness of the question which is often absent in new users. If we consider that new users in Stack Overflow are likely to be inexperienced then the questions that lacks in clarity and research effort, are also likely to be basic questions.
View Count of the Question	Post-hoc	6	the number of highly skilled programmers is likely to be less than the number of programmers in Basic or Intermediate level. Hence, the number of questions aroused by the advanced programmers are less likely to be searched by the programmers with basic skill set. Therefore, the Advanced questions should receive fewer views than the basic questions do.
Comment Count of the Question	Post-hoc	9	Users' interaction under a question in the comment section shows users' interest to discuss on the problem. These also show the usefulness of a question and may bear information about the question's difficulty.
Answer Count of the Question	Post-hoc	10	Generality of a topic can be estimated from the answer count on a question. If a question receives many answers it is likely to be related to a basic concept.

IV. PERFORMANCE EVALUATION

In this section, we empirically validate our SOQDE model based on 10-Fold cross validations of ten commonly used supervised learning algorithms.

After data preprocessing data and feature selection, we noticed our data as imbalanced with almost two-third question (63.88%) belonging to the basic category. Therefore, we applied the SMOTE [20] oversampling technique to increase the ratios of *Intermediate* and *Advanced* questions. Next, we have applied ten commonly used supervised learning algorithms that are commonly used for text classification in software engineering. Our list of algorithms include:

1) K-Nearest Neighbour (KNN) [21];
2) Random Subspace [22];
3) Bayesian Network [23];
4) Naive Bayes [24];
5) AdaBoost with Decision Stump [25];
6) Decision Tree [26];
7) Random Forest [27];
8) LogitBoost [28];
9) Support Vector Machines (SVM) [29]; and
10) Simple Logistic Regression [30].

We validated each of the algorithms using 10-fold cross-validations, where the dataset was randomly divided into 10 groups and each of the ten groups was used as test dataset once, while the remaining nine groups were used to train the classifier. Each of the 10-fold cross-validation was repeated 100 times and the mean performances of the models were computed. We also tuned the hyper-parameter of the classifiers to avoid overfitting and ensure better generalization. The primary goal of parameter tuning is to reduce the difference between cross-validation error and separate test set error. In the

TABLE III
PERFORMANCE OF DIFFERENT CLASSIFIERS USING ONLY TEXTUAL FEATURES

Classifier	Basic			Intermediate			Advanced			Mean Accuracy
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	
Adaboost	0.642	0.983	0.777	0.000	0.000	0.000	0.150	0.032	0.052	0.631
Bayesian Network	0.641	0.990	0.778	0.000	0.000	0.000	0.083	0.011	0.019	0.634
KNN	0.639	0.824	0.720	0.247	0.160	0.195	0.429	0.032	0.059	0.572
Decision Tree	0.646	0.821	0.723	0.257	0.144	0.185	0.300	0.126	0.178	0.574
Logitboost	0.670	0.925	0.777	0.380	0.144	0.209	0.316	0.063	0.105	0.634
Naive-Bayes	0.701	0.478	0.569	0.256	0.276	0.265	0.184	0.516	0.271	0.429
Random Forest	**0.652**	**0.977**	**0.782**	**0.429**	**0.062**	**0.108**	**0.600**	**0.032**	**0.060**	**0.643**
Random Subspace	0.639	0.992	0.777	0.000	0.000	0.000	0.286	0.021	0.039	0.636
Simple Logistic	0.645	0.955	0.770	0.310	0.053	0.091	0.444	0.042	0.077	0.628
SVM	0.679	0.753	0.714	0.289	0.239	0.261	0.222	0.168	0.192	0.560

TABLE IV
PERFORMANCE OF DIFFERENT CLASSIFIERS ON PRE-HOC FEATURES

Classifier	Basic			Intermediate			Advanced			Mean Accuracy
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	
Adaboost	0.498	0.610	0.548	0.000	0.000	0.000	0.391	0.753	0.515	0.445
Bayesian Network	0.582	0.686	0.630	0.525	0.348	0.418	0.582	0.671	0.623	0.570
KNN	0.644	0.637	0.640	0.613	0.570	0.591	0.726	0.803	0.763	0.658
Decision Tree	0.607	0.624	0.616	0.578	0.521	0.548	0.728	0.789	0.758	0.633
Logitboost	0.568	0.699	0.627	0.514	0.342	0.410	0.600	0.639	0.619	0.565
Naive-Bayes	0.582	0.686	0.630	0.525	0.348	0.418	0.582	0.671	0.623	0.570
Random Forest	**0.650**	**0.659**	**0.654**	**0.620**	**0.584**	**0.602**	**0.760**	**0.800**	**0.779**	**0.671**
Random Subspace	0.583	0.763	0.661	0.621	0.391	0.480	0.691	0.684	0.688	0.619
Simple Logistic	0.576	0.699	0.631	0.532	0.364	0.432	0.625	0.666	0.645	0.579
SVM	0.561	0.719	0.630	0.508	0.329	0.400	0.561	0.719	0.630	0.572

TABLE V
POST-HOC CLASSIFIERS' RESULTS

Classifier	Basic			Intermediate			Advanced			Mean Accuracy
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	
Adaboost	0.497	0.610	0.548	0.000	0.000	0.000	0.392	0.753	0.515	0.445
Bayesian Network	0.590	0.694	0.638	0.556	0.389	0.458	0.613	0.679	0.644	0.589
KNN	0.769	0.624	0.689	0.675	0.765	0.717	0.771	0.868	0.817	0.734
Decision Tree	0.642	0.669	0.655	0.611	0.574	0.592	0.727	0.734	0.730	0.654
Logitboost	0.594	0.697	0.642	0.546	0.426	0.479	0.637	0.642	0.640	0.593
Naive-Bayes	0.589	0.694	0.637	0.554	0.383	0.453	0.608	0.679	0.642	0.587
Random Forest	**0.731**	**0.746**	**0.738**	**0.710**	**0.706**	**0.708**	**0.841**	**0.821**	**0.831**	**0.752**
Random Subspace	0.635	0.776	0.698	0.658	0.496	0.566	0.774	0.747	0.760	0.676
Simple Logistic	0.606	0.689	0.645	0.560	0.444	0.495	0.663	0.695	0.679	0.609
SVM	0.586	0.699	0.638	0.556	0.418	0.477	0.658	0.668	0.663	0.598

following, we briefly discuss our parameter tuning for different algorithms.

1) For **Adaboost** classification algorithm, we used 100 estimators and decision stump as base classifier.
2) We used α = 0.5 as simple estimator and *hill-climbing* as local search algorithm for **Bayes Network**.
3) We applied **KNN)** with K = 5. Its performance decreases with increased number of neighbors.
4) For **Decision Tree**, confidence factor = 0.25 was chosen.
5) We applied **LogitBoost** with shrinkage = 0.3. The model overfits if we increase the shrinkage further.
6) For **Random Forest**, we used 100 estimators and restricted the maximum depth of the trees to 50. We did not observe any notable improvement with increased number of estimators or depths.
7) We applied **SVM** with $\epsilon = 1e-12$ and tolerance parameter = 0.001.
8) We applied **Simple Logistic** with heuristic stop = 50, the heuristic for greedy stopping while cross-validating the number of LogitBoost iterations is enabled.
9) For **Random Subspace**, we used subspace size = 0.5 and Reptree as base classifier.

We have performed classifier selection at first on feature vector generated with TF-IDF. The performance of different classifiers on ITF-IDF is shown on Table III, which indicates that the random forest based models achieved the highest mean accuracies. After that, we used the classification result of this model as a feature for next steps of classifications as shown in Table II.

Table IV and Table V show accuracy, precision, recall, and F-measure for all the classifiers mentioned above using 10-fold cross validation with pre-hoc features and post-hoc features respectively.

We take mean accuracy as the dominant performance metric. After comparing results for all the classifier mentioned above with 10-fold cross validation, we found that Random Forest classifier achieved the highest mean accuracy with both pre-hoc features and post features, i.e., 0.671 and 0.752 respectively. For Random Forest, mean Precision, Recall and F-measures were recorded as 0.669, 0.671, and 0.669. For pre-hoc features and 0.753, 0.752, and 0.752 for post-hoc features, respectively. Therefore, we consider Random Forest based models as the best performing model for SOQDE.

V. Implications of the Study

This study on the difficulty level categorization has several implications to the users of Stack Overflow such as:

1) Benefiting the users' teaching material collection from Stack Overflow.
2) Benefiting the users' learning process from Stack Overflow.
3) Providing helpful code examples to learners.
4) Routing questions to generate better answers.
5) Introducing a question difficulty aware variable reward system.

The following subsections details these potential applications.

A. Benefiting the users' teaching material collection from Stack Overflow

Nowadays, teachers/instructors of programming courses often take examples from SO posts to demonstrate real-world issues and sometimes for practice exercises. The level of students varies from sophomore year to graduate level. Hence, it is very useful to have a classification of posts/discussions that match the level of the target audience. Manually identifying them matching desired difficulty level is cumbersome as numerous entries are being posted in every minute and few of them are of a particular level, e.g. Advanced. Thus, the developed SOQDE tool is expected to benefit teachers select suitable examples more conveniently and appropriately.

B. Benefiting the users' learning process from Stack Overflow

The users on SO are of different expertise levels. Their grasp of a particular topic may vary. While learning a particular thing from SO, s/he may be confused about the topics and solutions available to the specific problem the user is facing. There may exist many solutions to a problem that troubles a user. Efficient and better solution may require the user to have knowledge on advanced topics whereas the same problem may be solved with basic concepts in a less efficient and less appreciated way. If a user has to use a significant amount of time in identifying SO posts that match his/her level, s/he may get discouraged and the learning process is hampered. Therefore categorizing the questions according to their difficulty levels may help a lot to improve the learning practice of the users of SO.

C. Providing helpful code examples to learners

People often prefer to take help through viewing code examples especially in case of learning how to use new APIs or a new language [31]. Often *basic* questions on SO are accompanied by some code snippet or code examples. The person who is attempting to answer the question often mentions a single API along with a line of code or a block of code in order to simplify the use of that API. This can be very helpful for the people who like to rely upon code examples while learning.

D. Routing questions to generate better answers

When a new question is asked in Stack Overflow, it is routed to some potential users who are experts on the topic so that answers are posted quickly. On the common topics, there are many users capable of answering, however, with different depths of knowledge. The users' capability is represented by his/her rating. If the difficulty level of a question is identified, it may be routed to the user with a rating that is supposed to match the difficulty level of the question. Consequently, experts will be relieved of dealing with trivial/simple questions. On the other hand, new users may be encouraged by being asked to reply to questions matching his/her level.

E. Introducing a question difficulty aware variable reward system

As answering Basic questions is eventually equal or more rewarding than answering Intermediate or Advanced ones, users of SO community are likely to get less incentives to answer Non-basic (Intermediate or Advanced) questions putting some extra efforts. To motivate users of SO community to answer more difficult questions, SO may introduce a variable reward based system considering difficulty level of the question concerned. SO may also introduce new badges for answering Intermediate or Advanced questions. Consequently, response time of difficult question is likely to be reduced and overall impact of SO on desired audience will increase. To implement such smart mechanism, SOQDE is expected to be of use.

VI. Threats to Validity

In this empirical study, there can be several threats that can challenge the validity of this study. There are four common threats to validity [32]. They are:

1) Internal validity
2) External validity
3) Construct validity
4) Conclusion validity

All these are discussed in the following subsections.

A. Internal validity

The primary threat to internal validity is choosing a programming language and tags. We chose to work with a dataset about Java Strings, Inheritance and Threads. Though questions about these topics or having *Java* as tags with the corresponding questions are very much common on Stack Overflow, they do not necessarily cover all the possible topics or questions available here. However, characteristics related to the difficulty of a topic may not be related to the language or topic highly. We think this threat is minimal for the following reasons:

- Java, being a very much popular, vast, and mature programming language, covers a lot of things common to other programming languages.
- Java unifies many of the programming philosophies such as object orientation and structured programming which brings many modern concepts under the same hood.
- We did not use any feature specific to Java to train our models.

Therefore, we believe that our internal threats to validity are not a matter of concern.

B. External validity

The consideration of difficulty levels of programming related questions may vary greatly from person to person according to their expertise level. However, our dataset was labeled by the first two authors who are computer science and engineering graduates who have hands-on experience in application development with Java. Tie-breaking of their conflicts was done by the third author who is a faculty member having experience of teaching Java. Hence, we may say that the annotation of the questions being biased by one's expertise level is minimal. Another threat may come from the impact of age of the raters. According to a survey conducted by Stack Overflow [6], we know that average developers' age is 29.6 years with a median of 27 years. Figure 4 shows the graphical representation of Stack Overflow survey participants' age. In our case, persons who labeled the dataset initially are aged between 22 to 36 years with an average of 30 years which is very close to that of the general users of Stack Overflow. This similarity implies that our methodology is not vulnerable to possible external threats.

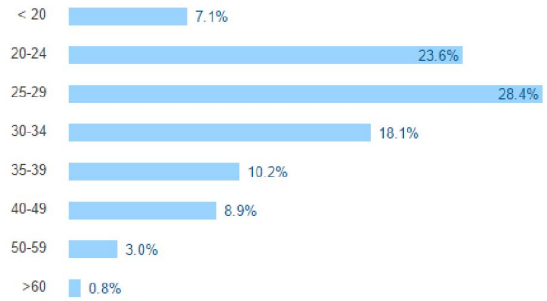

Fig. 4. Developers' profile: **Age** [Source: http://StackOverflow.com/insights/survey/2016]

C. Construct validity

In absence of any dataset available for our context, we had to build our own using the standard methodology as discussed in Section III. In short, we had the data labeled by the first two authors independently. The third author settled the disagreements independently. Thus, the preparation of the dataset is not likely to suffer from any bias. This dataset consisting of 936 questions performed well under different classifiers with accuracy up to 75.2% considering post-hoc features and 67.1% considering pre-hoc features only. We used 10-fold-cross validation while validating the performance. The first two authors had a high agreement in their independent labeling (Cohen's Kappa = 0.738). In case of disagreement, it was settled by the third author. This kind of majority voting for data labeling is a standard technique if there is no existing dataset for the problem scenario. Therefore, we can expect that there is not construct validity concern in our experiment.

D. Conclusion validity

The used dataset was collected from SO data-dump from late 2017. This dataset reflects the current trends of SO very well. The size of our dataset is close to the sizes of other studies recently done on SO data using manual labeling. For the imbalanced nature of dataset, we used oversampling using SMOTE [20] to remove any majority bias. We have adopted

widely used implementations of the commonly used machine learning techniques for this study. Therefore, our study does not have any serious threat to conclusion validity.

VII. Related Works

In the learning process, the learners of programming languages or even programmers take resort to tutorials and Q/A sites. The number of such kind of existing tutorials and Q/A sites is pretty huge. Among the technical tutorial and Q/A sites, Code Project and Stack Overflow are noteworthy. To analyze the interactions in these sites and to design make them more helpful to users, the researchers are carrying out various studies on the usage pattern and different characteristics of learning. Some studies such as [33] have tried to correlate the search trends on Google and Stack Overflow and show that technical terms searched and asked have strong correlation over time. Searching and asking for newer, specific technical terms have a stronger correlation, compared with older, general technical terms.

When the official documentation of any product is sparse or even non-existent, answers on Stack Overflow becomes a substitute for official documentation. To measure the interest of users on a different topic, research has been conducted on what kinds of questions are asked on Q/A websites for programmers, which questions are answered, and how best answers are selected in [34]. It found several categories of questions that are asked on Stack Overflow frequently. This study found that "How-to" questions are asked most frequently. Other categories of questions asked on Stack Overflow include about discrepancy, development environment, error, conceptual, code review, and novice inquiries. Another study [31] discusses the use of code examples while learning a new programming language or API. It shows that programmers often depend on code examples to support their learning.

A good amount of effort has been put to investigate the contents of questions on Stack Overflow. Arora et al. [35] aim to flag questions good or bad on Stack Overflow. Basically what they did is to assess if a question asked conforms to the standard of Stack Overflow. If a question matches the expected standard, it is marked as good and bad otherwise. Jiarpakdee et al. [36] discover the relation between question quality and effective features of that question. Here, by saying effective features, they mean positive sentiment, negative sentiment and politeness. They also used textual, community-based information extracted from the dataset they used. They found that though effective features alone cannot build a model good enough to predict question quality, one of the mentioned effective features - politeness ranking second important feature of the model, as a group can improve the performance of a model to identify the question quality. Baltadzhieva et al. [37] studied upon the individual terms or words used in writing the title and body of the question and studied to which extent do these terms can help to predict the probability of a question to receive an answer and the score that the question may have. Studying two indicators of question quality: question score and number of answers on that questions they came to a decision that their model performed better when terms were included.

A study [38] on Stack Overflow tries to augment API documentation from the insights in Stack Overflow posts. They applied a novel supervised machine learning approach SISE that uses the sentences as features. This study shows a way to consider Stack Overflow metadata and parts-of-speech tags to significantly improve extraction processes. Another study by Toba et al. [39] studies on the quality of the answers provided by the users on CQA sites. They proposed a hybrid hierarchy-of-classifiers framework to model QA pair with a view to identifying high-quality answers. They also found out a number of novel features to distinguish answers' quality. Rocha et al. [40] showed the feasibility of automatically generated tutorials. They developed and evaluated several methods to generate tutorials for APIs with the contents of Stack Overflow. They also organized those contents according to their complexity level that had better result regarding the generation methodology.

Competition based difficulty level estimation of the questions was done in [4]. This study compares users and questions. They adopted some assumptions. They assumed that the difficulty level of a question is higher than that of the expertise level of the user who asks the question in general. They also assumed that the user who provides the accepted answer has a higher expertise level than the user who posted the question. This is a hard assumption. Also, possible absence of answers at the early stage after a question is posted is beyond the applicability of this approach. An improvement over this is proposed in [5]. They added a textual description along with the user-question comparison.

Gamification incentives for badge recommendation have been discussed in [41]. This study develops a badge recommendation model based on item-based collaborative filtering which recommends the next achievable badges to users. The model calculates the correlation between unachieved badges and users previously awarded badges. They evaluated their model with the data from Stack Overflow question-answering website to examine if the recommendation model can recommend proper badges in an existing real community. However, they did not consider the difficulty of questions to recommend the badges.

VIII. Conclusion

In this paper we have presented SOQDE, a supervised learning based model to estimate the difficulty level of programming questions in Q/A sites such as Stack Overflow that can work both with and without submitted answers. We have used supervised learning techniques by developing a standard annotated dataset, tried different classifiers, and selected the best one in terms of accuracy and F-measure. The measured difficulty level of questions is likely to help design a smart reward model for the users that would consider the difficulty level of question. Thus it will attract the users answer difficult questions putting extra effort and get rewarded for it. Also,

estimation of the difficulty level of the questions will aid question routing by improving response time of the questions.

REFERENCES

[1] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and Hartmann, "Design lessons from the fastest q&a site in the west)," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 2857–2866.

[2] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu, "Searching questions by identifying question topic and question focus." in *ACL*, vol. 8, 2008, pp. 156–164.

[3] M. A. Suryanto, E. P. Lim, A. Sun, and R. H. Chiang, "Quality-aware collaborative question answering: methods and evaluation," in *Proceedings of the second ACM international conference on web search and data mining*. ACM, 2009, pp. 142–151.

[4] J. Liu, Q. Wang, C.-Y. Lin, and H.-W. Hon, "Question difficulty estimation in community question answering services," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 85–90.

[5] Q. Wang, J. Liu, B. Wang, and L. Guo, "A regularized competition model for question difficulty estimation in community question answering services," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1115–1126.

[6] "Stack overflow developer survey results," http://stackoverflow.com/insights/survey/2016, accessed: 2017-04-06.

[7] A. Bacchelli, L. Ponzanelli, and M. Lanza, "Harnessing stack overflow for the ide," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*. IEEE Press, 2012, pp. 26–30.

[8] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[9] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[10] M. F. Porter, "Snowball: A language for stemming algorithms," 2001.

[11] S. Bird and E. Loper, "Nltk: the natural language toolkit," in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 2004, p. 31.

[12] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[13] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, 1987.

[14] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, 2003, pp. 133–142.

[15] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, 1997, pp. 412–420.

[16] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *Machine learning: ECML-98*, pp. 137–142, 1998.

[17] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro *et al.*, "Application of high-dimensional feature selection: evaluation for genomic prediction in man," *Scientific reports*, vol. 5, 2015.

[18] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1289–1305, 2003.

[19] P. C. Rigby and M. P. Robillard, "Discovering essential code elements in informal documentation," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 832–841.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[21] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[22] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[23] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[24] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[25] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[26] J. R. Quinlan, "Learning efficient classification procedures and their application to chess end games," in *Machine learning*. Springer, 1983, pp. 463–482.

[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[29] B. Schölkopf and C. J. Burges, *Advances in kernel methods: support vector learning*. MIT press, 1999.

[30] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, no. 1-2, pp. 167–179, 1967.

[31] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming q&a in stackoverflow," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. IEEE, 2012, pp. 25–34.

[32] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[33] C. Chen and Z. Xing, "Towards correlating search on google and asking on stack overflow," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 1. IEEE, 2016, pp. 83–92.

[34] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 2011, pp. 804–807.

[35] P. Arora, D. Ganguly, and G. J. Jones, "The good, the bad and their kins: Identifying questions with negative scores in stackoverflow," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM, 2015, pp. 1232–1239.

[36] J. Jiarpakdee, A. Ihara, and K.-i. Matsumoto, "Understanding question quality through affective aspect in q&a site," in *Proceedings of the 1st International Workshop on Emotion Awareness in Software Engineering*. ACM, 2016, pp. 12–17.

[37] A. Baltadzhieva, G. Chrupala, G. Angelova, K. Bontcheva, and R. Mitkov, "Predicting the quality of questions on stackoverflow." in *RANLP*, 2015, pp. 32–40.

[38] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 392–403.

[39] H. Toba, Z.-Y. Ming, M. Adriani, and T.-S. Chua, "Discovering high quality answers in community question answering archives using a hierarchy of classifiers," *Information Sciences*, vol. 261, pp. 101–115, 2014.

[40] A. M. Rocha and M. A. Maia, "Automated api documentation with tutorials generated from stack overflow," in *Proceedings of the 30th Brazilian Symposium on Software Engineering*. ACM, 2016, pp. 33–42.

[41] R. Gharibi and M. Malekzadeh, "Gamified incentives: A badge recommendation model to improve user engagement in social networking websites."