Towards Understanding Developers' Machine-Learning Challenges: A Multi-Language Study on Stack Overflow

Alaleh Hamidi*, Giuliano Antoniol*, Foutse Khomh*, Massimiliano Di Penta†, Mohammad Hamidi‡
*Polytechnique Montréal, †University of Sannio, ‡University of Tehran
{alaleh.hamidi, giuliano.antoniol, foutse.khomh}@polymtl.ca, dipenta@unisannio.it, mohamad.hamidi@ut.ac.ir

***Abstract*—Machine Learning (ML) is increasingly being used as an essential component of modern software systems. Also, the maturity of the adopted techniques and the availability of frameworks have changed the way developers approach ML-related development problems. This paper aims at investigating, by analyzing Stack Overflow (SO) posts related to ML, how the questions about ML have been changing over the years, and across six different programming languages. We analyzed 43,950 SO posts in the period 2008-2020, studying (i) how the number of ML-related posts changes over time for each programming language, (ii) how the posts are distributed across different phases of a ML pipeline, and (iii) whether posts belonging to different languages or phases are more or less challenging to address. We found that some programming languages are fading while others are becoming more popular in ML development. While model-building questions are the most discussed in general, the level of challenges posed by the other phases of the ML pipeline appears to be language-dependent. Results of this work could be used to better understand ML challenges in different programming languages, and, possibly, to improve ML tutorials related to different languages.**

***Keywords*-stack overflow; machine learning; machine learning Life cycle; programming language;**

I. Introduction

Machine learning (ML) is increasingly becoming an essential component of modern software systems. The demand for industry adoption of ML has increased. A recent NewVantage Partners Big Data and Artificial Intelligence (AI) Executive Summary shows that 92% of organizations are increasing their pace of investment and 62% of them have already seen measurable results from their investments in big data and ML [1].

The growth in ML adoption produces new challenges for software developers as reported in several studies [2]–[4]. One consolidated way to study challenges developers face in a given field is to analyze posts on Question and Answer (Q&A) forums such as Stack Overflow (SO) [5], [6]. Some studies report an increasing number of ML-related posts on SO [5]–[7]. This indicates that, in recent years, ML has attracted growing interest among software developers, especially after 2015 [8].

However, developers may face different problems when coping with ML in different programming languages. That is, they may be constrained by the specific project/application to use a language that may or may not be suitable for ML-intensive development. While previous work [7] has studied the increasing attention received by ML from developers, there is little knowledge about its evolution and variation across programming languages, and whether different ML phases receive more attention or create more challenges.

In this work, we systematically study what kind of problems developers encounter in each stage of the ML development life cycle when using a specific programming language. We analyze 43,950 posts of SO dated from 2008 to 2020, and related to ML, to answer the following research questions:

- **RQ1**: How does the number of ML-related posts related to different programming language change over the years?
- **RQ2**: How are posts distributed across different phases of a ML pipeline, and how does this change over time?
- **RQ3**: To what extent are posts belonging to different languages and different phases answered and after how long?

We consider six programming languages, including popular ones (e.g., C/C++, Java, and Python), as well as languages used in scientific/statistical computation (e.g., R).

As one may expect, Python has the lion-share as a programming language. Languages such as R are still very popular, while ML-related questions for other languages such as C, C++, or Matlab tend to increase less, or not increase at all. Concerning the ML phases to which questions refer, building, evaluation and requirement are the most frequently encountered concerns, when we consider posts with three or more stars. Model building alone accounts for about one-third of posts. The building and evaluation phases are the sources of most concerns, they account for more than 50% of concerns, but it is decreasing in recent years likely because of the availability of high-level frameworks, e.g., PyTorch or Keras for Python.

This work's results can inform the research community about the language-specific ML challenges to investigate the solutions and improve the languages' capabilities. Besides, ML developers and tutorial creators can utilize this work to get a deep perspective of ML practitioners' language-related challenges to provide more comprehensive PL libraries and ML tutorials.

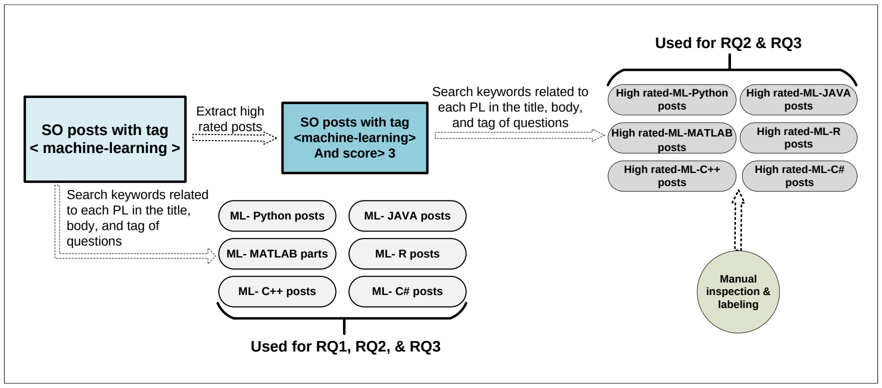
Used for RQ2 & RQ3
SO posts with tag < machine-learning >
Extract high rated posts
SO posts with tag <machine-learning> And score> 3
Search keywords related to each PL in the title, body, and tag of questions
High rated-ML-Python posts
High rated-ML-JAVA posts
High rated-ML-MATLAB posts
High rated-ML-R posts
High rated-ML-C++ posts
High rated-ML-C# posts
Search keywords related to each PL in the title, body, and tag of questions
ML- Python posts
ML- JAVA posts
ML- MATLAB parts
ML- R posts
ML- C++ posts
ML- C# posts
Manual inspection & labeling
Used for RQ1, RQ2, & RQ3

Figure 1. Data analysis methodology

II. Methodology

The *goal* of our study is to analyze SO posts related to ML, characterizing them from different perspectives. We want to understand how the number of such posts evolves over the years, how this varies among different programming languages, how posts are distributed across the different ML phases, the extent to which posts receive a response, and the time taken to provide these responses. The *perspective* is that of educators interested in understanding ML topics that are challenging to ML developers. They can leverage this information to improve training materials introducing these topics. Researchers could also leverage the information, to develop better tools and analysis techniques to support ML developers during challenging phases of the development life cycle of ML systems. The *context* consists of SO posts related to ML, dated between 2008 and 2020, and belonging to different programming languages.

Figure 1 provides an overview of our methodology, described in the following.

A. Context and Data Collection

The popularity of SO and its large amount of ML-related questions, makes it a representative data source for our study. We consider 43,950 SO questions posted between July 2008 and December 2020. The first step in our study is to identify the subset of SO questions that represent the ML-related challenges. We followed a snowball sampling approach similar to Alshangiti et al. [7]. We started with the "machine-learning" tag and extended the list of ML-related tags based on relevance and co-occurrence. More specifically, for each tag expanding, we inspected the list of top 25 tags that occurred with that tag and chose the ones related to ML. We repeated this process until we obtained the top 50 most-used ML-related tags (reported in Table I).

As Table I shows, the "machine-learning" tag includes the greatest number of ML-related questions. Note that this tag can cover the topic of other ML-related tags as well. We collected the questions with "machine-learning" tag, using Stack Exchange Data Explorer, which is an open-source tool to run queries against public data from the Stack Exchange network [9].

Table I
Top 50 most commonly used Machine Learning Tags

Tag	Occur.	Tag	Occur.
machine-learning	43,953	neural-network	17,391
supervised-learning	460	conv-neural-network	6709
unsupervised-learning	541	recurrent-neural-network	2300
reinforcement-learning	1805	rnn	668
prediction	2304	deep-learning	19,804
regression	7522	image-recognition	1287
linear-regression	4966	object-detection	3230
non-linear-regression	570	sentiment-analysis	1629
nls	545	cluster-analysis	5391
classification	6850	hierarchical-clustering	1013
classifier	545	pca	2282
multilabel-classification	670	autoencoder	1155
multiclass-classification	560	word2vec	1865
document-classification	217	word-embedding	811
text-classification	1405	tf-idf	1157
logistic-regression	2995	rfe	113
svm	4270	feature-engineering	296
svmlight	98	feature-selection	1213
decision-tree	2139	feature-extraction	1458
random-forest	2902	cross-validation	2144
naivebayes	950	confusion-matrix	791
perceptron	436	precision-recall	352
dbscan	496	scikit-learn	22015
knn	1409	tensorflow	65,213
k-means	2991	r-caret	1864

Table II
2008 - 2020 SO ML posts with more than three stars per language manually verified

	Manually-Verified Posts					Posts Per
Lang.	Nbr.	Correct	Misclassified	Reassigned	Lost	Language
C#	47	42	5	4	1	42
C/C++	128	111	17	14	3	114
Matlab	169	153	16	14	2	157
Java	133	119	14	4	10	126
R	320	285	35	26	9	289
Python	3389	3367	22	0	22	3411
Other						47
Overall	4186	4077	109	62	47	4186

B. RQ1: Post classification across programming languages

Before automatically classifying posts across programming languages, we performed some textual preprocessing. Since the body of each question may contain source code, before starting to search for keywords, we preprocessed the content of the post using regular expressions matching and removed the content between the `<code>` and `</code>` tags. This preprocessing step is necessary because the code in the post can contain the searched keywords as the name of some variables like C and R, therefore introduce false positives.

To determine the ML posts related to each programming language, we used the language name as a keyword to search in the title, question body, and tag of each post from our initial dataset of ML-related questions. Since some posts could contain questions related to more than one programming language, we count such posts in the datasets of each relevant programming language separately. After removing the source code, we pruned out HTML tags.

We validated the reliability of such a categorization manually inspecting a subset of them, and, in particular, all posts having more than three stars, because these posts will be used for the analyses of RQ2 and RQ3. Results of this manual analysis are reported in Table II, showing how many posts were misclassified and how many reassigned. When a post was reassigned, it was reassigned to a different language, including the *Other* language category (i.e., not one of the analyzed languages), thus the column *Total* is the results of manual reassignment and cannot be obtained by adding the number on the same row.

In our sample of 4186 posts, we found a total of 109 misclassifications (about 3%), out of which 62 were manually re-assigned. The remaining 47 posts were discussing other programming languages and though not strictly relevant we kept and are reported as *Other*.

We then computed the proportion of questions asked for each programming language, from 2008 to 2020. We also computed the proportion of accepted questions for each programming language during the same period. Moreover, to determine whether there is a trend in the questions for each programming language, we apply the non-parametric Mann-Kendall test for trend detection [10], [11] and the Sen's slope statistic [12]. Specifically, we first apply the Mann-Kendall test to reject the null hypothesis H_0 : *There is no significant trend in the time series.* The Mann-Kendall test is a measure of association between an increasing time series, automatically generated, and the observed time series. It computes the S statistic, which is closely related to the τ correlation, i.e., it is based on the ranks of the samples and indicates the trend direction, i.e., if positive, the trend increases (conversely, it decreases). Then, we compute the Sen's statistic [12] to gain an insight of the slope size [12]: the higher the value, the greater the slope. In a nutshell, the slope is the median of the deltas between consecutive points. Being based on the median, Sen's slope (also known as Theil-Sen estimator [12]) is robust to outliers.

Finally, we use the Lanzante statistic [13] which, given a time series, determines whether it has points in which the slope significantly changes, and if yes indicates where.

To address RQ1, we report and discuss the post trends over the years among the different programming languages, using the aforementioned statistical procedures to identify the presence of trends.

C. *RQ2: Classification of questions into phases*

In this research question, we classify posts according to the ML phases defined by Amershi et al. [14]. The analysis has been conducted manually (see Table II) and, to consider more representative and high-quality posts, we consider those having a score greater than three stars (i.e., four or five stars). This selection also makes the manual inspection more feasible. While we are aware that excluding some (low-rated) posts might somewhat bias our dataset, our intent is to look at very meaningful (highly-rated) posts that can be considered as good representatives of the various phases. Low-rated posts may be useless, trivial, or even wrong, and may produce a misleading representation of the post distribution along phases.

Note that such a filtering was not performed to address RQ1, because, in that case, we were simply interested to gather an overview of the posts across programming languages and over the year. We manually inspected 4186 high-rated ML posts to categorize them and consider the accuracy of the programming language classification.

Two ML researchers independently classified and labeled 100 randomly-selected questions. They were asked to assign only one label to each question. We computed the Cohen's k [15] inter-rater agreement, which resulted equal to 0.88, which is an almost perfect agreement.

This substantial level of agreement means that the two researchers shared a common understanding of the posts, therefore, for the remaining dataset, we divided it into two groups and tasked each of the researchers to manually classify one group.

During the manual inspection, each rater performed two tasks (1) determined the correctness of programming language assignment to post (which for RQ1 was only done on a sample, and (2) by reading the post content, assigned the post to one or more ML phases as they were defined by Amershi et al. [14].

Amershi et al. [14] suggest a nine-phase ML workflow informed by prior experiences developing AI applications at Microsoft. Some phases are data-oriented like collection, cleaning, and labeling. But other stages are model-oriented (e.g., model requirements, feature engineering, training, evaluation, deployment, and monitoring). We determined our ML categories based on this workflow to classify and label the questions. Next, we describe these categories as they were interpreted by the raters when manually classifying SO posts:

Model Requirement (MR): The questions under this category are related to considering the implementation feasibility of features with ML, looking for appropriate libraries, choosing the most appropriate models for a given problem, identifying relevant and representative data, and installing required infrastructure, e.g., packages.

Data Collection & Preprocessing (DCP): All the questions related to data collection and preprocessing are placed in this category; integrating datasets, removing inaccurate, noisy or duplicate records and outliers from the dataset, data adoption into suitable data format required, data transformations (like normalization, min-max scaling, and data format conversion), and balancing the classes.

Feature Processing (FP): This category includes questions related to data labeling and feature engineering; assigning ground truth labels to each record, encoding data, extracting and selecting informative features to reduce data dimensionality.

Model Building (MB): The questions under this category are related to identifying the training and test dataset, creating the ML model using the APIs, training the model, tuning the model parameters, storing models to disk, and loading them to use later.

Model Evaluation (ME): These questions are about evaluation method selection, evaluating the performance of a model, visualizing the behavior of the model, and interpreting the output of a model.

Model Deployment (MD): The questions related to deploying the application on suitable devices or platforms, model reuse (like conversion of a model trained using one library and then using the trained model for prediction in an environment using another library), getting different results from one ML model implemented by different platforms (when using similar setting), and robustness are placed in this category.

Model Monitoring (MM): This category includes the questions related to monitoring the functionality of ML applications for performance and potential errors, during real-world executions.

Not Related to ML Challenges (NO): This category includes questions that cannot be placed under any of the above categories.

All Phases (all): Includes posts related to all ML development phases: model requirement, data collection, data preprocessing, feature processing, model building, model evaluation, model deployment, and monitoring.

To address RQ2, we report the distribution of the manually-classified posts along phases, and how this varies between programming languages.

D. RQ3: Analysis of how posts were answered

To address RQ3 we analyzed, for different programming languages and for different ML phases, the percentage of posts that received an accepted answer, and, if this happens, the distribution of response time (in hours). That is, we replicate the analysis conducted by Alshangiti et al. [7], but (i) considering the impact on different programming languages separately, and (ii) by using the phase classification taxonomy of Amershi et al. [14]. As also done for RQ2, and since posts could not receive an accepted answer because they are trivial/uninteresting/misleading, we only consider posts having a score greater than three.

In summary, we report (i) the number and percentage of posts without confirmed answers (ii) for posts that have been answered, boxplots depicting their distribution along programming languages and along ML phases. Also, to statistically compare different programming languages and different phases we use the (non-parametric) Dunn's test [16] using the Benjamini-Hochberg correction [17] because of multiple comparisons being performed.

Table III
OVERALL NUMBER OF POSTS PER LANGUAGES

	With accepted answers	Without accepted answers	Total	Without accepted answers %
C#	150	215	365	59
C/C++	272	294	566	52
Matlab	589	636	1225	52
Java	622	823	1445	57
R	1074	1616	2690	60
Python	9591	12,663	22,254	57

III. STUDY RESULTS

This section reports the study results, aimed at addressing the research questions formulated in Section II.

A. RQ1: How does the number of ML-related posts related to different programming language change over the years?

Table III reports the overall number of posts, along with the number and ratio of accepted ones. As the table shows Python and R play the lion-share role while others such as C# and C/C++ have a lower number of posts.

The role of Python is not surprising, given the availability of many data science-related libraries (e.g., pandas, numpy, PyTorch, or Scikit-learn) but, more importantly, the hype of deep learning techniques [18], and the adoption of related libraries. In that case, Python represents an almost compulsory choice as the support in other programming languages is relatively limited.

The case of R is also interesting to discuss. R has a long-term tradition in terms of usage by the scientific community coupled with a large community of developers and a consolidated infrastructure (i.e., the CRAN archive). While nowadays Python libraries such as the ones mentioned above make available almost any statistical procedure, R remains popular for researchers who got used to it (especially working in areas beyond computer science), and is a valid choice for those having to use very specific statistical procedures that may not be available in Python yet.

Table IV and Figure 2 show the overall evolution of posts over the years, and its trend per programming language, respectively. As it is possible to notice from the plots in

Table IV
OVERALL NUMBER OF POSTS OVER THE YEARS PER LANGUAGE WITH AND WITHOUT ACCEPTED ANSWERS

	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total
C#	0	5	9	9	22	11	14	22	33	42	65	70	63	365
C/C++	0	3	14	33	41	48	49	85	88	70	48	40	47	566
Matlab	2	11	24	39	108	157	161	168	183	149	120	54	49	1225
Java	0	6	28	60	112	115	123	171	177	191	136	135	191	1445
R	0	2	12	28	61	119	153	225	438	404	374	360	514	2690
Python	0	4	46	83	170	314	441	819	1855	3330	4366	4423	6403	22,254

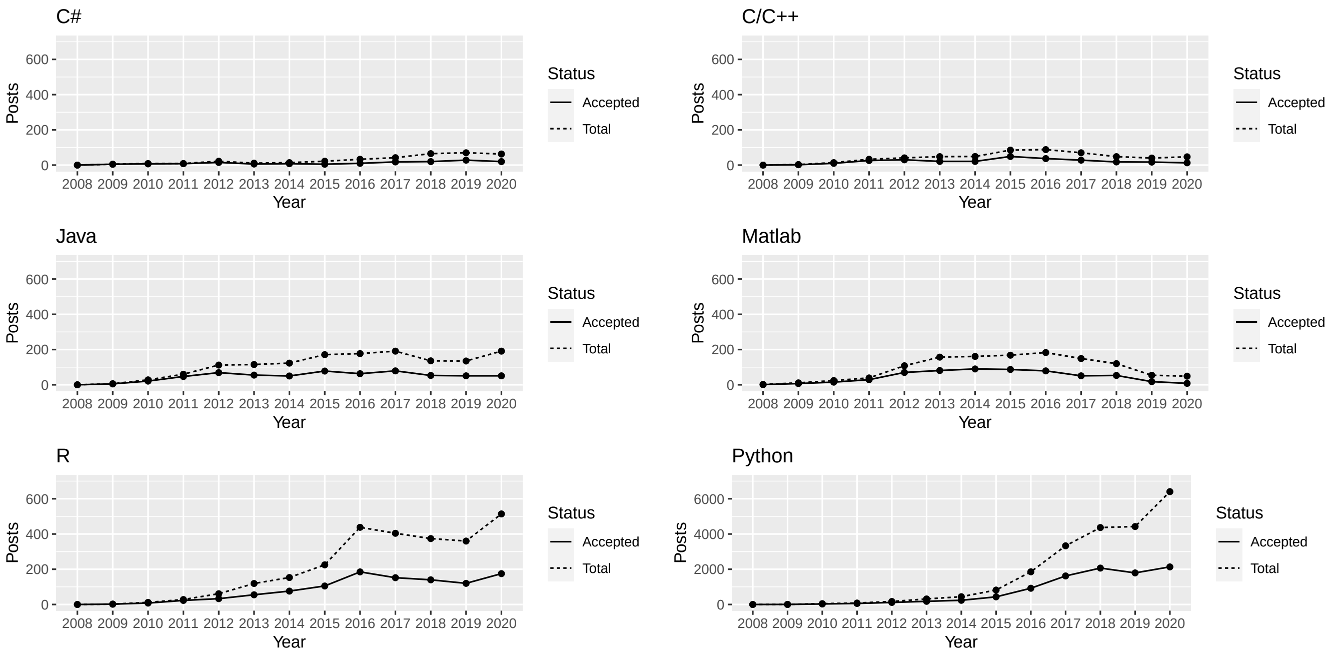

Figure 2. Evolution of total and accepted posts over the years. Note: Python has a different scale.

Figure 2, there is a consistent trend within two groups of programming languages. On the one hand, we have Java, R, and Python where we can clearly notice an increasing trend. On the other hand C#, C/C++ and Matlab have either a lower increasing trend (if any) or even a decrease (Matlab). It is unclear why this happens, although the only difference between Matlab and the other languages is that the former is part of a commercial product (by MathWorks), although Matlab-like (e.g., GNU Octave) open source solutions exist. However, it may also be the case that Matlab targets a different category of developers, e.g., including industrial developers working in automation control and especially in the automotive domain, where the adoption of ML solutions is rapidly increasing especially to support self-driving or driving assistance features. One possible explanation is that, in the context of industrial projects, developers may target internal sources of knowledge rather than SO.

Consistently to what we observed in terms of the number of posts, Python and R have a consistent and steeper increase, plus they do not exhibit a plateau (nor a decrease) in the last two/three years.

We confirmed this observation by applying the Mann-Kendall Test to determine whether our time series dataset has an increasing or decreasing trend. We also applied the Sen's slope statistics [10]–[12] to estimate the slope of the trends. The results of these analysis are reported in Table V and Table VI, respectively. As for the Mann-Kendall test,

Table V
OVERALL POSTS MANN-KENDALL TREND TEST - H_0 : SLOPE=0

Language	p-value	S	VarS	τ
C#	<e-04	68	266	0.88
C/C++	0.02	39	267	0.50
Matlab	0.07	30	268	0.38
Java	0.0001	63	267	0.81
R	<e-4	66	268	0.85
Python	<e-05	78	268	1

Table VI
OVERALL POSTS TREND ANALYSIS: SEN'S SLOPE

Language	p-value	Slope
C#	$<$ e-04	5.50
C/C++	0.02	5.41
Matlab	0.07	10.0
Java	0.0001	16.28
R	$<$ e-04	42.77
Python	$<$ e-05	455.52

Table VII
2014 ON: TREND ANALYSIS: SEN'S SLOPE

Language	p-value	Slope
C#	0.02	9.5
C/C++	0.13	-7.6
Matlab	0.03	-23.8
Java	0.44	3.5
R	0.2	49.66
Python	0.002	1018.5

Table V reports the test p-value, the S statistics (the higher S, the steeper the trend is expected), the S variance (VarS), and the τ coefficient. Table VI reports the Sen's trend analysis p-value and slope.

The Mann-Kendall test confirms the presence of trends for all languages but Matlab. Sen's slopes vary evenly between the minimum of about five (for C# and C/C++) and the maximum of about 450 for Python.

Finally, we applied the Lanzante statistics [13] to further confirm the presence of a trend and divide the likely point of change. Lanzante statistics identified two possible change points in 2012 and 2015. Also in this case, languages can be broadly divided into two groups. On the one hand, C/C++ and Matlab have a change point in 2012, while the other languages exhibit a change point in 2015. The latter has a very clear explanation, i.e., the introduction of very popular frameworks for deep learning, such as TensorFlow and Keras (both released in 2015) or PyTorch, released in 2016. Indeed, if we consider two sub-series up to 2014 and from 2014 on, we obtain the trend analysis results shown in Table VII, where only Python is increasing.

Results show an increasing trend for Java, R, and Python, with the latter having the steepest increase. The rapid increase of Python posts in 2015–2016 and the delta in 2019-2020 may be explained by the surge of Python frameworks, as well as by the increasing adoption of ML in research and practice.

B. RQ2: How are posts distributed across different phases of a ML pipeline, and how does this change over time?

Table IX reports the distribution of posts (among the manually analyzed ones) for the ML various phases defined by Amershi et al. [14], and Table VIII the percentage of SO posts per phases and language. Note that since a post may belong to multiple categories, the sum of percentages does not add up to 100.

Looking at Table IX, the difference in terms of relative importance of phases is evident. The majority of questions in our dataset are related to model building (MB), model evaluation (ME), and model deployment (MD) phases. Model requirement (MR) seems to be a concern for C# and C/C++ (because many C# and C/C++ developers are looking for the equivalent of Python libraries in these languages), while data collection (DC) and feature processing (FP) posts are frequent in C#. We can conclude that MB, ME, and MD are the main sources of concerns.

Table X reports the Python distribution of posts per phase and years. Similar tables for other languages are omitted due to space reasons. Given the conclusion of RQ1, Python is indeed the most interesting language to consider for an in-depth discussion. Table X shows an almost steady increase for MB, MD, and ME posts. Moreover, MB, ME, and MR account for about 50% and more of concerns, and this throughout the years. Overall, for the other programming languages, we observed similar trends.

Table XI reports links to examples belonging to different phases.

MR posts address a variety of specific topics related to model requirements. Post P1 asks: *There is an infinite stream of 4 possible events. What ML algorithm is the best to predict what the next event will be, based on the order that events have come in in the past? And how long of a history that the predictor should maintain? And if more than one event can happen simultaneously on each round, does that change the solution? Python or C libraries are nice, but anything will do.* The developer was not sure what ML tool fits such a prediction problem regardless of the language. Also, adapting to more complex learning and improving it performance could be an issue, e.g., the P2 issue *Recently I've been reading a lot about Q-learning with Neural Networks and thought about updating an existing old optimization system in a power plant boiler composed of a simple feed-forward neural network approximating an output from many sensory inputs.* As another example for C#, the post P3 is asking *I'm trying to tackle the classic handwritten digit recognition problem with a feed forward neural network and backpropagation, using the MNIST dataset...I finished writing it some time ago and been debugging since, because the results are quite bad. At its best the network can recognize 4000/10,000 samples after 1 epoch and that number only drops on the following epochs, which lead me to believe there's some issue with the backpropagation algorithm.* Here, in the end, the problem was not of a better algorithm, but rather, an input rescaling problem, thus more a matter of pre-processing.

Data collection, pre-processing and feature processing are a concern too. Consider post P6, where the problem was a missing rescaling of the inputs, causing the Deep Neural Network to produce inconsistent results. Similarly, P5, P7 and P8 cope with re-shaping, rescaling, and feature space reduction.

In summary, developers asking questions are concerned about ML algorithm and their implementation, given the problem requirements at hand. Since languages such as C, C++, or C# where MR questions occur more (in percentage) are not languages specifically targeting ML or statistics (as instead R, Matlab, and also Python thanks to its

Table VIII
PERCENTAGE OF ML POSTS (WITH MORE THAN THREE STARS) PER PHASE AND LANGUAGES

	MR	DCP	FP	MB	ME	MD	MM	NO	ALL	Unclear
C#	26	29	3	43	-	3	-	3	15	-
C/C++	33	10	3	38	7	24	-	8	-	-
Matlab	13	11	6	66	10	3	-	6	2	-
Java	1	12	7	48	3	27	2	4	6	2
R	10	15	8	46	22	13	2	4	1	-
Python	19	13	18	47	33	12	2	5	1	5
Other	5	7	3	19	-	12	-	10	-	55

Table IX
DISTRIBUTION OF PHASES OF ML POSTS WITH MORE THAN THREE STARS FOR DIFFERENT LANGUAGES

	MR	DCP	FP	MB	ME	MD	MM	No	all	Unclear	Total
C#	9	10	1	15	0	1	0	1	5	0	42
C/C++	31	9	2	36	6	23	0	7	0	0	114
Matlab	17	15	8	90	13	4	0	8	2	0	157
Java	1	13	8	56	3	31	2	4	6	2	126
R	23	35	19	114	54	30	4	8	2	0	289
Python	429	291	384	1046	734	270	39	108	1	109	3411
Other	2	3	1	8	0	5	0	4	0	24	47
Overall	512	376	423	1365	810	364	45	140	16	135	4186

Table X
PYTHON NUMBER OF POSTS (WITH MORE THAN THREE STARS) PER PHASE AND OVER THE YEARS

	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total
MR	0	3	17	30	26	40	28	44	74	65	55	28	19	429
DCP	0	0	7	4	6	15	18	32	44	65	60	26	14	291
FP	0	0	3	4	8	26	28	40	67	88	72	33	15	384
MB	0	0	12	15	34	50	42	86	186	242	231	105	43	1046
ME	0	0	6	9	27	36	37	56	156	171	144	72	20	734
MD	0	0	1	3	9	13	11	25	46	52	65	36	9	270
MM	0	1	1	1	1	4	3	1	5	8	4	7	3	39
NO	0	0	2	1	6	4	2	6	24	28	18	14	3	108
ALL	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Unclear	0	0	1	5	4	3	8	8	20	30	13	14	3	109
Overall	0	4	50	72	121	192	177	298	622	749	662	335	129	3411

frameworks), such questions likely originate from developers working on a domain-specific application, which need to use ML but do not have specific ML-related expertise. Indeed, languages like C, C++ are used, for example, in the development of embedded systems or Internet-of-Things (IoT) applications, where ML may find a meaningful usage. Other areas of application include video games, where C++ and C# are used as languages.

MB questions vary from very specific questions, for example, P9:*What is the parameter "max_q_size" used for in "model.fit_generator"*, related to the evaluation of deep learning models in Python, to more implementation-oriented issues, e.g., P10 *While tuning the hyperparameters to get my model to perform better, I noticed that the score I get (and hence the model that is created) is different every time I run the code despite fixing all the seeds for random operations. This problem does not happen if I run on CPU* and e.g., P11 for Matlab: *I am using gradient descent to minimize the cost function to implement Logistic Regression. I wrote a function in Matlab that returns both the cost and the gradient of each parameter evaluated at the current set of parameters. My cost function works, but the gradient function does not. What's wrong?* Finally, P12 concerns with language-specific issues occurring when porting Deep Neural Networks on Android. We conjecture that MB questions like the ones discussed exhibit an increase also considering the growing adoption of deep learning algorithms. ME is also an often encountered concern. Consider the Python posts P13 and P14: *... I have a working network that is training, but the minibatch loss is at about 425 right now and the accuracy at 0.0, and for the LSTM MNIST example code (linked) the minibatch loss was about 0.1 and the accuracy about 1.0. My hope is that if I can change the activation function to use the SoftMax function, I can improve results...* and *After using "sklearn.linear_model.LogisticRegression" to fit a training data set, I would like to obtain the value of the cost function for the training data set and a cross validation data set." How can I evaluate cost function for scikit learn LogisticRegression?* Similar doubts about evaluation are raised with other languages, e.g., a Java developer asked

Table XI
EXAMPLES OF SO POSTS FOR DIFFERENT PHASES

Phase	Language	Link
MR	Python	P1: https://stackoverflow.com/questions/2524608/machine-learning-algorithm-for-predicting-order-of-events
		P2: https://stackoverflow.com/questions/40158232/updating-an-old-system-to-q-learning-with-neural-networks
	C#	P3: https://stackoverflow.com/questions/56365587/backpropagation-algorithm-giving-bad-results
DCP	Python	P5: https://stackoverflow.com/questions/52184142/keras-sequential-dense-input-layer-and-mnist-why-do-images-need-to-be-reshape
	R	P6: https://stackoverflow.com/questions/9316794/am-i-using-the-wrong-data-type-with-predict-nnet-in-r
FP	Python	P7: https://stackoverflow.com/questions/28254824/feature-space-reduction-for-tag-prediction
	Matlab	P8: https://stackoverflow.com/questions/40560139/matlab-feature-selection
MB	Python	P9: https://stackoverflow.com/questions/36986815/what-is-the-parameter-max-q-size-used-for-in-model-fit-generator
		P10: https://stackoverflow.com/questions/50744565/how-to-handle-non-determinism-when-training-on-a-gpu
	Matlab	P11: https://stackoverflow.com/questions/38853370/matlab-regularized-logistic-regression-how-to-compute-gradient
	Java	P12:https://stackoverflow.com/questions/38910471/tensorflow-retrained-inception-v3-model-crashes-on-android
ME	Python	P13: https://stackoverflow.com/questions/37796595/tensorflow-lstm-rnn-output-activation-function
		P14: https://stackoverflow.com/questions/35956902/how-to-evaluate-cost-function-for-scikit-learn-logisticregression
	Java	P15: https://stackoverflow.com/questions/21674522/get-prediction-percentage-in-weka-using-own-java-code-and-a-model
MD	Python	P16: https://stackoverflow.com/questions/42945509/keras-input-shape-valueerror
		P17: https://stackoverflow.com/questions/34175174/extract-features-using-pre-trained-tensorflow-cnn

How can I get prediction percentage in Weka using my own Java code and a model? (P15).

We observed that model deployment (MD) touches a large variety of issues from operating systems to the availability of pre-trained models. In P16 the concern is an error due to the use of 32 versus 64 bits software *I made an image classifier with Keras using Theano as a Backend and a Sequential model. When I run my script on Windows 7 32 Bit, I get an error but If I run it on Elementary OS 64 Bit, it runs without any problem. Is there such a big difference between different Operating Systems?* , while in P17 the concern is the availability of a model ready to be deployed *when one does not have enough data to train the CNN, I may expect this to outperform a pipeline where the CNN was trained on few samples. I couldn't find a pickle file (or similar) with a pre-configured CNN feature extractor. Do such pre-trained networks exist and where can I find them. Where could I find a CNN+weights.*

Model requirement, building, evaluation, and deployment are the most frequently encountered concerns. Model building and evaluation challenges for Python started decreasing after 2018 that may be because of emerging new frameworks like Keras that facilitate ML development.

C. RQ3: To what extent are posts belonging to different languages and different phases answered and after how long?

As shown in the last column of Table III, the ratio of posts without accepted answers for all languages is between 52% and 60%. If we consider the sample of posts with more than three stars, as shown in the last column of Table XII, such a percentage lowers between 21% and 41%, with a noticeable exception (Java) of only 4%. It must be noted that the number of such posts considered for Java is relatively small (126, as from Table IX), all highly-rated posts may get accepted. Confirming the findings of RQ1 (where we have observed an increase of Python and R posts), R and Python have higher percentages of not accepted posts than other languages. This is also the case for C# which, however, similarly to Java, has a very small number of high-rated posts (42 as from Table IX). While there could be many reasons for such observation, it is possible that the greater adoption of Python and R for ML generates more questions, but also more controversial answers. Languages such as Matlab and C/C++, which are more specialized and tied to specific usages and needs in the context of ML (e.g., speed, see for instance the Darknet project[1]) or the need to integrate into simulation engines (e.g., Matlab).

When looking at the distribution across phases, RQ2 findings are confirmed also in terms of "controversial" posts that received no answers. First, we can see a relatively high percentage of posts with no accepted answers for MR, but also (confirming results of Alshangiti et al. [7]) for DCP which is however lower for Python than for C#, C/C++, and R. This can be explained because Python has, with respect to those languages (including also R), much more data parsing/cleaning facilities thanks to the wide availability of libraries. Interestingly, this also happens for FP, MB, and ME, despite R has a quite variety of facilities for feature processing, and model building, and to some extent for model evaluation. However, the availability in Python of recent frameworks, including PyTorch, Scikit-learn, or Keras makes the life of ML developers easier.

MD seems to be less of a concern for Matlab and R. For the former, as previously discussed the language is quite mature in fields related to automation, and therefore deployment mechanisms may be clear. Also, often Matlab source code is translated into C before being deployed, therefore challenges arise (as it can be seen in the table) for that language instead. R is more used for experimentation than for production, and this could explain a relatively lower percentage of unaccepted questions there. Finally, MM does not seem to be an issue. There could be a variety of

[1]https://pjreddie.com/darknet/

Table XII
PERCENTAGE OF ML POSTS (WITH MORE THAN THREE STARS) WITH NO ACCEPTED ANSWER PER PHASE AND LANGUAGES

	MR	DCP	FP	MB	ME	MD	MM	NO	ALL	Unclear	Overall
C#	45	40	100	27	-	100	-	0	0	-	34
C/C++	42	34	0	42	17	44	-	43	-	-	40
Matlab	30	27	13	16	24	26	-	38	50	-	21
Java	0	16	0	2	0	4	0	0	0	0	4
R	40	46	37	44	47	30	0	13	50	-	41
Python	37	23	26	30	31	36	16	38	100	41	31
Other	100	34	100	50	-	40	-	26	-	34	41

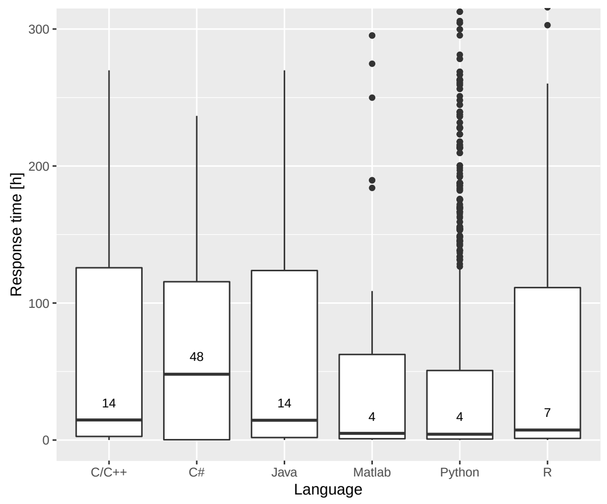

Figure 3. Distribution of post response time by language.

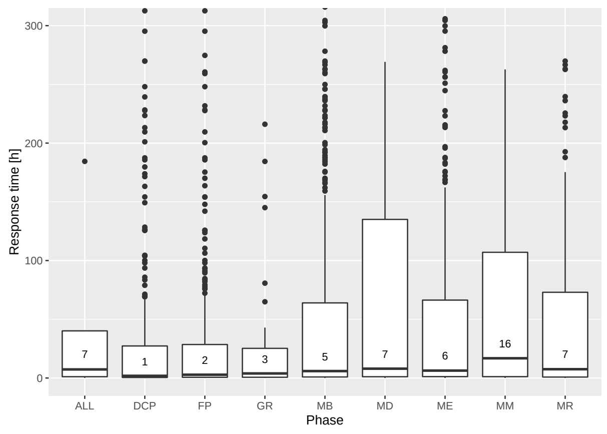

Figure 4. Distribution of post response time by phase.

explanations, including the fact that ML is not applied yet in contexts where monitoring is a crucial concern.

Figure 3 depicts boxplots (positive outliers truncated to ease readability) showing the distribution of response times (in hours) for ML posts related to different programming languages. A Dunn's test with Benjamini-Hochberg correction indicates that Python has the shortest response time than other languages, with a statistically significant difference (p-value< 0.05) with all other languages but Matlab. This, unsurprisingly, confirms the high popularity of Python for ML, and consequently the availability of developers able and willing to quickly provide answers on SO.

Figure 4 shows, instead, boxplots related to response times related to different phases. Results indicate that posts related to data collection and processing (DCP), feature processing (FP), and those related to all phases (ALL) have significantly shorter response time (Dunn's test p-value< 0.05) than others. DCP and FP are less specific to the technicality of ML algorithms, therefore they are likely to receive attention also from non-experts, but just people knowledgeable about how to extract and clean data from given sources, as well as from domain experts. Model Monitoring (MM) is, for example, a very specific feature which not all ML developers may take care of, therefore the reaction time appears to be longer.

Model requirements, data collection/processing, and model building are the phases generating more questions without accepted answers than others. Python has, in general, fewer questions without accepted answers than R. Generic questions, and those related to data collection and processing or feature processing are answered faster than others.

IV. THREATS TO VALIDITY

Threats to *construct validity* concern the relationship between theory and observation. These are mainly related to imprecision in our measurements. Querying SO for "machine learning" posts may miss relevant posts. Also, the programming language classification can be error-prone. This threat has been mitigated in RQ1 by manually analyzing a sample of posts having a high rate, as well as manually checking all data for RQ2. Also, the classification for RQ2 can be subjective and error-prone. We mitigated this threat by having two raters separately assessing a sample of 100 posts before completing the task independently, and computing their Cohen k inter-rater agreement. Finally, the number of SO posts is only a proxy of the developer base, but it may rather represent a technology's popularity among developers.

Threats to *internal validity* concern factors internal to our study that can influence our results. While we discuss possible reasons for the trends of ML questions among programming languages (RQ1) and phases (RQ2), these are only possible interpretations, and there might be other reasons why such trends occur.

As for RQ3, while previous work has used the extent to which questions were answered and the time needed to answer a question as indicators of ”challenging” questions [7], we cannot really claim a causal-effect relationship between them. This is because there might be many other reasons why a question receives (or does not get) answers, and why a question is answered immediately or after a long time. To partially (mitigate) this threat in RQ3, we consider posts having a high score, which at least are less likely to receive low attention because they are not considered particularly interesting nor relevant by developers.

Threats to *external validity* concern the generalizability of our findings. Since our study focuses only on SO posts, results are limited only to the challenges that some developers state on SO. Developers may use other specific forums, as well as internal mailing lists of their projects/organizations. To generalize the results of our study, additional studies on other sources should be conducted. Also, while we analyzed popular programming languages, including the most widely used language for ML (Python) and languages used for scientific software (Matlab and R), we may have excluded interesting discussions related to other languages.

V. Related Work

Amershi et al. [14] studied ML development activities in Microsoft projects, classifying, as we have discussed in section II, ML development into nine phases. Based on the analysis conducted, Amershi et al. also distilled some best practices for ML development. We take inspiration from Amershi et al. work, as we study how SO posts are distributed along ML phases, and how this varies over the years and across programming languages.

Mostly related to ours is the work of Alshangiti et al. [7]. They studied the extent to which ML posts on SO are challenging to be answered, by considering the proportion of unanswered posts among six phases of a ML process. Complementary to Alshangiti et al., we look at the ML-related posts on SO also from the perspective of considering different programming languages, trends over the years, and using a phase classification proposed by Amershi et al. [14].

Islam et al. [5] conducted a manual inspection on 3,243 highly-rated SO posts related to ten ML libraries and classify these questions into seven typical categories of an ML pipeline to determine the correlation between the library and the phase. However, Islam et al. only study posts related to some ML libraries, therefore their results cannot be generalized to the whole body of ML challenges. In our study, we look at ML posts related to six different programming languages.

Bangash et al. [19] studied SO posts about ML to understand which ML topics are significantly more discussed than others. To this aim, they applied Latent Dirichlet Allocation (LDA) [20] on the textual content of SO posts to categorize them. Our work differs from Bangash et al., because we fully rely on a manual analysis to classify posts into ML phases.

Other studies analyzed interests and challenges of big data developers [21], the refactoring of ML-intensive systems [22], ML API misuses [23], and the root causes of ML-framework bugs [24]–[26].

Patel et al. [27], [28] conducted an early study on the difficulties faced by software developers in the adoption of ML techniques. Yang et al. [29] interviewed non-expert ML developers to understand the challenges they encounter when developing ML applications. This study shows that the most important challenges are related to model performance issues, unexpected errors, and the understanding of ML algorithms. With respect to the aforementioned interview-based studies, our analysis focuses on a different data source, i.e., SO posts. Also, our analysis specifically deals with (i) the comparison of different programming languages, (ii) the classification of posts into phases, and (iii) the extent to which posts belonging to different phases and languages are answered or not, as well as the time required to receive an answer. That being said, the results of our study indirectly highlight challenges that developers are facing with ML for different languages and different phases. Also, the measurements of RQ3 are, similarly to Alshangiti et al. [7], a proxy of the extent to which posts are challenging to be answered.

VI. Conclusion

In this paper, we report the results of an empirical study that examined the characteristics of questions asked by developers about machine learning (ML) development. Through a quantitative and qualitative analysis of Stack Overflow posts from 2008 to 2020, and related to ML, we observe that the number of ML-related posts has changed over time for different programming languages. More specifically, there is an increasing number of posts related to machine learning development, especially for Java, R, and Python. The majority of highly-rated ML-related questions concern issues related to model requirement, model building, and evaluation. Model building alone accounts for about one-third of all the studied posts. Questions about ML development in Python are usually answered much faster than questions related to ML development using other programming languages. This is not surprising given the current popularity of Python language in the ML community. However, given the important amount of questions asked by developers about ML development using other programming languages, educators and tool creators should ensure that enough training resources and tool support are available to support ML development using languages other than Python. Our data and results are available at an online Appendix [30] to ensure our findings' reproducibility.

References

[1] New Vantage Partners (NVP), *Big data and AI executive survey 2019: How big data and AI are accelerating business transformation*, 2019. [Online]. Available: http://newvantage.com/wp-content/uploads/2018/12/Big-Data-Executive-Survey-2019-Findings-122718.pdf

[2] D. Sculley, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, *Machine learning: The high-interest credit card of technical debt*, 2014.

[3] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, December 2015, pp. 2503–2511. [Online]. Available: https://dl.acm.org/doi/10.5555/2969442.2969519

[4] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "What's your ML Test Score? a rubric for ML production systems," in *Reliable Machine Learning in the Wild - 30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016, pp. 1–5. [Online]. Available: https://research.google/pubs/pub45742/

[5] M. J. Islam, H. A. Nguyen, R. Pan, and H. Rajan, "What Do Developers Ask About ML Libraries? A Large-scale Study Using Stack Overflow," in *arXiv:1906.11940v1*, 2019, pp. 1–13. [Online]. Available: https://arxiv.org/abs/1906.11940

[6] A. Joorabchi, M. English, and A. E. Mahdi, "Text mining stackoverflow: Towards an Insight into Challenges and Subject-Related Difficulties Faced by Computer Science Learners," *Journal of Enterprise Information Management*, vol. 29, no. 2, pp. 255–275, 2016.

[7] M. Alshangiti, H. Sapkota, P. K. Murukannaiah, X. Liu, and Q. Yu, "Why is developing machine learning applications challenging? a study on stack overflow posts," *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–11, 19-20 September 2019.

[8] A. McIntosh, S. Hassan, and A. Hindle, "What can android mobile app developers do about the energy consumption of machine learning?" *Empirical Software Engineering*, vol. 24, pp. 562–601, 2018.

[9] J. Atwood, *Introducing Stack Exchange Data Explorer*, 2010. [Online]. Available: https://stackoverflow.blog/2010/06/13/introducing-stack-exchange-data-explorer/

[10] K. W. Hipel, *Time series modelling of water resources and environmental systems*. Elsevier Amsterdam; New York, 1994.

[11] C. Libiseller and A. Grimvall, "Performance of partial mann–kendall tests for trend detection in the presence of covariates," *Environmetrics*, vol. 13, no. 1, pp. 71–84. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/env.507

[12] P. K. Sen, "Estimates of the Regression Coefficient Based on Kendall's Tau," *Journal of the American Statistical Association*, vol. 63, p. 1379–1389, 2012.

[13] J. Lanzante, "Resistant, robust and non-parametric techniques for the analysis of climate data: Theory and examples, including applications to historical radiosonde station data," *International Journal of Climatology*, vol. 16, pp. 1197–1226, 1996.

[14] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software Engineering for Machine Learning: A Case Study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, Canada, 25-31 May 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8804457

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

[16] O. J. Dunn, "Multiple Comparisons among Means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[17] Y. Benjamini and Y. Hochberg, "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[18] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[19] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, and K. Ali, "What do developers know about machine learning:a study of ML discussions on StackOverflow," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, Montreal, Canada, 25-31 May 2019, pp. 260–264. [Online]. Available: https://ieeexplore.ieee.org/document/8816808

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[21] M. Bagherzadeh and R. Khatchadourian, "Going big: A large-scale study on what big data developers ask," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 432–442. [Online]. Available: https://dl.acm.org/doi/10.1145/3338906.3338939

[22] Y. Tang, R. Khatchadourian, M. Bagherzadeh, R. Singh, A. Stewart, and A. Raja, "An empirical study of refactorings and technical debt in Machine Learning systems," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, Madrid, Spain, 22-30 May 2021, pp. 238–250. [Online]. Available: https://ieeexplore.ieee.org/document/9401990

[23] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online q&a forum reliable?: a study of api misuse on stack overflow," in *ICSE '18: Proceedings of the 40th International Conference on Software Engineering*, May 2018, pp. 886–896. [Online]. Available: https://dl.acm.org/doi/10.1145/3180155.3180260

[24] Y. Zhang, Y. Chen, S.-C. Cheung, Y. Xiong, and L. Zhang, "An empirical study on TensorFlow program bugs," in *In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018)*, New York, USA, July 2018, pp. 129–140. [Online]. Available: https://dl.acm.org/doi/10.1145/3213846.3213866

[25] M. J. Islam, G. Nguyen, R. Pan, and H. Rajan, "A comprehensive study on deep learning bug characteristics," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, USA, August 2019, pp. 510–520. [Online]. Available: https://dl.acm.org/doi/10.1145/3338906.3338955

[26] M. J. Islam, R. Pan, , G. Nguyen, and H. Rajan, "Repairing Deep Neural Networks: Fix Patterns and Challenges," in *ICSE '20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, June 2020, pp. 1135–1146. [Online]. Available: https://dl.acm.org/doi/10.1145/3377811.3380378

[27] K. Patel, J. A. Fogarty, J. A. Landay, and B. L. Harrison, "Investigating statistical machine learning as a tool for software development," in *CHI '08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2008, p. 667–676. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1357054.1357160

[28] K. Patel, J. A. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learning as a tool for software development," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, July 2008, p. 1563–1566. [Online]. Available: https://dl.acm.org/doi/10.5555/1620270.1620333

[29] Q. Yang, J. Suh, N. Chen, and G. A. Ramos, "Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models," in *DIS '18: Proceedings of the 2018 Designing Interactive Systems Conference*, June 2018, pp. 573–584. [Online]. Available: https://dl.acm.org/doi/10.1145/3196709.3196729

[30] A. Hamidi, G. Antoniol, F. Khomh, M. Di Penta, and M. Hamidi. Towards understanding developers' machine-learning challenges: A multi-language study on stack overflow – replication package. [Online]. Available: https://github.com/plmlchallenges/ml-pl-challenges