Software Developer's Productivity Prediction using Deep Learning Based on GitHub Public Repositories

Md Rezwan Hassan Khan
Department of Computer Science
Carleton University
Ottawa ON CA
Email: mdrezwankhan@cmail.carleton.ca

***Abstract*—The software industry is a growing business. The more it will grow, the more software developers will be in demand and more software teams will be created to make products in this industry. However, till now no exact methods, metrics, tools are created that can measure the performance of a software team and individual developers. This research proposed a quantitative analysis where we discussed 12 important metrics that are correlated with the productivity of developers. We also proposed a machine learning productivity prediction model that can forecast the future productivity trend based on these metrics. We have used Long short-term memory(LSTM)which is a recurrent neural network to build this prediction model. The data-set we have used for this research were 4 public GitHub repositories. By considering each repository as a development team our LSTM model can predict future development team's productivity. Baidu's and TensorFlow team's productivity will go up in the near future whereas it will go down for Home Assistant's development team and the productivity level will remain unchanged for Magneto's development team. Lastly, we also suggest individual analysis that can improve the performance of each development team based on the model's performance**

I. Introduction

According to the 2019 Government of Canada report, the Canadian Private sector spent 7.5 billion dollars in ICT's Research and Development (1). This statistic shows that big companies are spending billions for making this software, hiring the best developers, trying their level best to innovate and become more efficient. Surprisingly, they are making this investment without any productivity forecast when it comes to software development. This raises two important questions :

1) Is there any way to track how well this software is performing?
2) What is the productivity of a software development team/developers?

Up until now, no valid tracking model has been made that can solve these answers. Usually, in economic terms productivity is the measure of output compared to its input. But it is impossible to measure productivity in Software Development in this way due to many external and internal factors. However, there are many important metrics such as code metrics that can evaluate software developers' performance. Another issue in the Software Development industry is that team leaders and project managers of a software development company often face difficulties monitoring the performance of their software developer because they don't know metrics and ways it needs to measure it(2). A lot of research has been done in this area but no standards tools were created to measure the productivity of developers and till now researchers and developers are in search of answers that can measure the productivity of a developer.

According to our research and previous studies, two research questions can help us measure a software developer's productivity.

- **RQ1**: What are the metrics that can be used to measure the performance rate of a developer?
- **RQ2**: How can we build an AI model that can automatically predict a developer's performance based on past data?

Solving these two scientific questions can help us find the metrics that can evaluate the performance of a software developer and with these data, we can build a machine learning model that can be considered as a productivity measuring tool that may help to identify the performance level of a software developer/teams. We proposed a solution for research question 1 by identifying the 13 most important quantitative metrics that are related to software developer's productivity. Out of these 13 metrics, we took 7 metrics to define our recurrent neural network prediction model that can forecast the future productivity of a software development team's productivity based on public GitHub repositories.

The remaining parts of the paper are designed as follows. Section 2 talks about the related work like this paper. Section 3 discussed the methodology, section 4 explained our analysis, section 6 deals with threats validity and the paper concludes with sections 7 and 8 with the conclusion and future work.

II. Related Work

In this section,we have discussed similar work like ours that work with similar data to analyse the challenges in the development industry.

How does Working from Home Affect Developer Productivity? A Case Study of Baidu During the COVID-19 Pandemic

They have collected 4000 records of 139 developers for 138 working days from different size projects with different creation times. Each record in the dataset has several metrics that represent the activities of a developer in one day. Due to the security policy of Baidu, the numeric metrics are standardized. Besides this, they also examine factors like size, age, language used in a project that has a correlation between productivity of developers(3).

A Deep Dive on the Impact of COVID-19 in Software Development

The paper analyzes developers' productivity before and after pandemic through a statistically time series analysis showing changes in 10 metrics behavior over time determining the productivity change. For that, they have selected 100 GitHub repositories from January 2019 to May 2020. Along with that, they surveyed 279 developers for a better understanding of the productivity, code quality, well-being of software developers in this pandemic. They have gone through five selection criteria to identify their 100 GitHub project and retrieve ten important metrics from those repositories either from cloning git repositories or using REST API. Afterward, they performed a time series analysis by dividing the data set into six-time. Next, they try to find whether these metrics have a positive or negative impact. The author performed a pairwise two-sample t-test to check whether there is any statically significant difference in the mean value of each metric between windows July 2019-December 2019 and January 2020 to May 2020. If the value is less than 0.05 mean it has a significant difference or else vice versa. This paper state that the pandemic didn't reduce the productivity of the developers and code quality. They found 12 observations that can help developers and companies to become more productive and both physically and mentally healthy(4).

How to Evaluate the Productivity of Software Ecosystem: A Case Study in GitHub

This paper tries to define the productivity of software ecosystems, find the factors of productivity and build a model that can measure the productivity of software ecosystems and evaluate their model on real-world open software ecosystems like GitHub and Stack Overflow. They find out various active communication in the open-source software ecosystem define productivity. They have divided productivity into two parts(Primary and secondary). Factors that affect the productivity of the software ecosystem are project popularity, development language, number of participants, their activities like pull requests, and so on. Based on these factors, they have first created a linear regression model to find the relation between the number of participants and productivity. Afterward,real discovery approach was taken to find the relations between productivity and other factors. After that, they analyze their model with different ecosystems of platforms like Stack Overflow and GitHub(5).

Public Git Archive: a Big Code data-set for all

The paper describes how they created a public GitHub repository archive that contains 182014 git repositories. This paper also explains the pipeline of the whole procedure. The author makes this publicly available that includes the source code of each project and important information. It is one of the biggest publicly available GitHub data-sets(6).

Stock Prediction Based on LSTM under Different Stability

The authors discuss in their paper, Stock Prediction Based on LSTM under Different Stability, that linear models for stock price prediction cannot accurately utilize historical data compared to non-linear models. The authors explain how LSTM includes a time component within its network without encountering the vanishing gradient problem, affirming its usefulness for time series analysis.The LSTM was conducted using Tensor Flow in Python and the Root Mean Squared Error (RMSE) was chosen for a performance measure. The LSTM contained 10 hidden layers and a time-step of 5(7).

III. Methodology

A. Description of Data Set

As private repositories are not available for our analysis, we used GitHub public repository for our research. For our initial analysis and demo project, we are taking 4 Github public repositories. According to the site HackerNoon, they have classified the top 100 most valuable GitHub Repositories Out of 96 million public Git repositories(8). From these 100 repositories, we have selected four public repositories from the different software domain industry as shown below.

1) Baidu: Deep Learning FrameWork
2) Tensorflow: Machine Learning FrameWork
3) Magneto: E-commerce
4) Homeassistant: Home Automation

This Github's public repositories were not presented in a form of a dataset. So to extract these data, GitHub Rest API was used. We are going to solve these two questions based on four public repositories from GitHub. In step 1, we have extracted all data such as commits, pull requests, and so on from the four repositories using GitHub Rest API and save these as CSV files.

B. Data Mining

GitHub Rest API contains all the git-related information. We needed the knowledge of python and a few libraries like pandas, NumPy, and config to extract these data. We save our GitHub username and access token in a .py file which we named config.py. After that, we search for the corresponding API endpoint for commits, pull requests, branches, and issues from Github API Documentation(9) . For example , API endpoint for commit is "GET /repos/:owner/:repo/commits" After that we have provided Github Repositories name and its owner's name as input parameter as "url = api + '/repos///branches?page=&per_page=100'.format(owner, repo, page_number)". And then we retrieve all the commits using this URL and few already defined functions to extract the commits. We follow the same procedure to extract information like

TABLE I
METRICS USED TO MEASURE PRODUCTIVITY OF A SOFTWARE DEVELOPERS/TEAMS

Metrics
1. Number of Active Contributors
2. Number of New Developers
3. Number of Branches
4. Number of Created Pull Requests
5. Number of Closed Pull Requests
6. Pull Request created and closed per month
7. Number of Commits
8. Number of Created Pull Requests Comments
9. Number of Updated Pull Requests Comments
10. Number of Bug-x commits
11. Size of the project
12. Language used
13. Age of the size

branches, pull requests, and issues. After that, we save each information as a different CSV file. So for each repositories, the data we extracted and converted into CSV files are :

- Commits.csv
- Branches.csv
- Pull Request.csv
- Issues.csv

We are retrieving data for the recent six to eight years for each repository. We have used implementation by Xhentilo Karaj (9) to extract GitHub data.

C. Proposed solution for Research Question 1 and Initial Data Pre-Processing :

In step 2, we tried to find the metrics having a high correlation with the developer's productivity. For this, we have used defined metrics from previous research papers(3; 4) where they claimed that these are the best quantitative metrics that can measure the developer's productivity as shown in Table 1.

Next, we defined each metric base on our GitHub Repository data which is explained below. For initial analysis and demo projects, we worked with 7 metrics out of these 13 metrics. Before defining the metrics, we have to clean and pre-process our data frame. We dropped all the unnecessary columns that are not needed and added a new column that converts the date column into pandas date format using the to_datetime function. We then added three more columns that show the day, month, year of the date columns separately.

1) **Number of Created Pull Requests**
   We can find all the pull requests from our pull request data frame. We then extract pull requests basis on their creation date. For this, we have used the index number, columns "created a year" and created a month. Based on this, we found the number of pull requests created in each month of every year as shown in Figure 1.
2) **Number of Closed Pull Requests**
   We have used the same methodology to find the number of updated pull requests like we did to find the number of created pull requests. Instead of using created year and created month columns, we have used updated year and updated month columns to find all metrics according to each month of each year.
3) **Pull Request created and closed per month**
   First, we found commits created and updated in the same month and follow the same procedure to find the number of commits created and updated in the same month according to each month of each year.
4) **Number of Commits**
   We have used the index number and commits creation date to extract all commits in each month of every year.
5) **Number of Created Pull Requests Comments**
   First of all, we try to find all created pull requests that have comments in them. For this, we shrink our original dataset with only index, creation year, creation month, and pull request's body columns. And we drop all the commit message's rows that were empty using panda's built-in function dropna. After that, we found out all the created pull requests with messages in them for each month of every year.
6) **Number of Updated Pull Requests Comments**
   We follow the same methodology we use to find the number of created pull requests with comments. Instead of using the columns; created year and creation month, we have used updated year, updated month; the remaining procedure was the same.
7) **Number of Bug-Fix commits**
   To find the commit related to bug fix, we searched words like error, bug, fix and defect in commit message (10) and filter out all these commits that have these words. Afterward, we find the total number of bug fix commits in each month of every year.

Month	2016	2017	2018	2019	2020	2021
Jan	0	327	1093	777	198	259
Feb	0	383	644	614	148	168
March	0	530	953	881	200	12
April	0	338	1022	457	484	0
May	0	544	919	252	202	0
June	0	632	1076	227	164	0
July	0	900	787	160	248	0
Aug	66	1191	827	197	427	0
Sep	205	1247	878	296	472	0
Oct	67	1164	717	344	305	0
Nov	451	994	1016	206	362	0
Dec	634	840	922	273	329	0

Fig. 1. Number of Commits for Baidu in each month from the year 2016 to 2021.

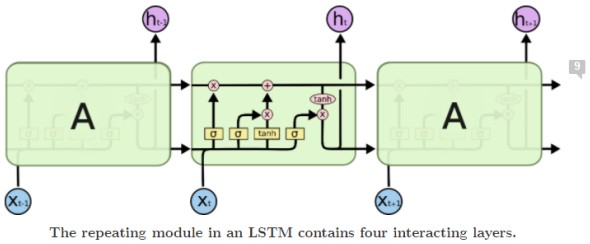

Fig. 2. The Repeating Module in an LSTM Contains Four Intereacting Layers.

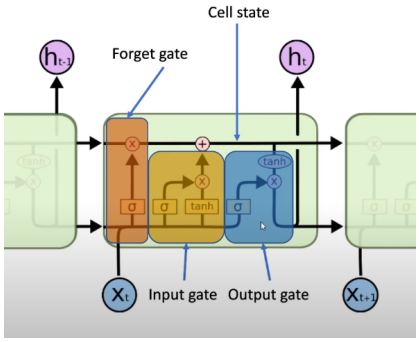

Fig. 3. LSTM cell state and different Gates.

D. Proposed solution for Research Question 2 :

We have proposed a time series analysis where we have used LSTM to forecast how each metric is changing with time; is it increasing with time or is it decreasing with time? We have divided our dataset into train and test datasets and try to see how our model is forecasting on unseen data, in our case the unseen data will be our test data for every 7 metrics. Our prediction and forecasting models were implemented using the Long Short Term Memory (LSTM) model. LSTM is a unique version of a Recurrent Neural Network (RNN) that works well with sequence data which is useful for prediction problems(10). LSTM was chosen over other traditional regression models because of its ability to remember past information. So, time series analysis works best for this kind of data where there is a correlation with the variable itself and LSTM is perfect for doing a time series analysis because it remembers information for long periods as shown in the figure. Figure 2 illustrates LSTM's cell structure. A cell in an LSTM model can be divided into three parts, forget gate, input gate and output gate(11; 12). The forget gate decides which information the model should remember and also decides to discard information. It is composed of a sigmoid function with for its input function. Thus, the output of forget gate is either one or zero. If the output is one, then no information is lost or forgotten, and it is continued through the cell state. The input gate decides what new information LSTM will save in the cell state. It is composed of both a sigmoid and tanh function. Where, sigmoid layers decide which data is stored and tanh.distribute a weight to each output value that was passed through the sigmoid layer of input function. The output gate decides what to output to the following cell and is composed of a sigmoid function and tanh function. Where the tanh function gives a weight to the output in the range between negative one to one. The following Figure depicts the three gates within a cell Implementation of this code utilized the TensorFlow and Keras libraries in Python by Krish Naik (13) and Sreeni(14). A sequential API was used to create the LSTM models using Keras. The LSTM model contains layers with 50 units. Given that this model used more than one LSTM layer, the return sequence was set to "True" allowing for a three stacked LSTM. The timestep was set to four and a dropout layer was added to reduce overfitting. Lastly "Adam" was used as an optimizer and the Root Mean Squared Error (RMSE) was used to compute the loss in the models. The epoch was set to 100 with a batch size of 64 and the training and validation loss were plotted. From the Keras library, the function model.predict() was used to evaluate how each mode works for our datasets. Each model splits the entire dataset into a training and testing dataset, where the training dataset consists of the first months and testing dataset is the remaining end months. The model created in this study is Metric Forecasting Model : We run our LSTM model separately on each 7 metrics. The input variable for the model is the metric itself and the output is also that metric itself. For example , when we run our LSTM model on the number of commits metric, the model considers the number of commits of the previous fours days with the output value being the fifth day's number of commits as we set the time step to 4.

IV. Case Study Analysis

The model created in this study is Metric Forecasting Model is described below: We run our LSTM model separately on every seven metrics. The input variable for the model is the metric itself, and the output is also that metric itself. For example, when we run our LSTM model on the number of commits metric, the model considers the number of commits of the previous fours days with the output value being the fifth day's number of commits as we set the time step to 4.

For all four public git repositories, we run our model for each metric and forecast how each metric will change in the future. The RMSE values were calculated for each company per metric for each LSTM model. Given the limited space in this paper, Baidu will be exemplified in the prediction graphs while the remaining graphs of the other companies will be included in the appendixes.

1) Model Evaluation: The training dataset comprised sixty-five percent of the entire dataset, while the remaining thirty-five is the testing dataset. Figure 4 shows the plotted training and testing loss for Baidu's Commit's LSTM model, while Figure 5 shows Baidu's pull request comment training and testing loss.

In Figure 4, the testing loss curve is slightly below the training loss curve that implies that the model may be underfitting the data. For figure 5, however, the difference between the two lines is wider that indicates the model has overfitted. Thus, the

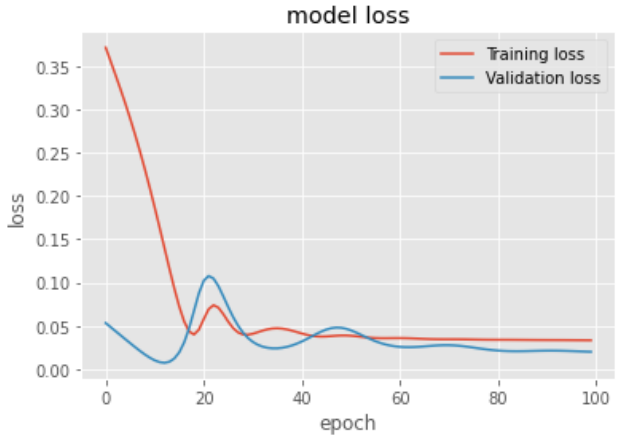

Fig. 4. Baidu's Commits 's Training and Validation loss.

Fig. 5. Baidu's Created Pull Request 's Training and Validation loss.

model is doing well with training data compared to test data. However, the testing error decreases with the increased number of epochs which reduces the wider overfit. We have analyzed each LSTM model for each metric. However, due to limited space, we included it in the appendix.

In addition to visual representations of the model's performance, the root mean squared error (RMSE) values were calculated to determine accuracy(7). The RMSE evaluates the amount of error in a model and whether the value is small or large depends on the units of the dependent variable, y, in the model. In our models, the dependent variable is the metric. A low RMSE is desired for accuracy. The formula for the RMSE is as follows, where y is the predicted value, $\hat{y}$ is the actual value, and n is the number of observations as shown in the equation.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y} - y)^2}{n}}$$

The RMSE values were calculated for both training and testing datasets for each metric for each GitHub repository. Figure 6

	Number of Commits	Created Pull Requests	Closed Pull Requests	Pull Request created and closed per month	Created Pull Requests Comments	Updated Pull Requests Comments	Number of Bug-x commits
Training Rmse	724.89	1.15	42.5	n/a	0.79	4.85	43.56
Testing Rmse	421.39	19.75	33.34	n/a	15.45	20.36	45.81

Fig. 6. RMSE values of the training and testing datasets for all metrics for Baidu.

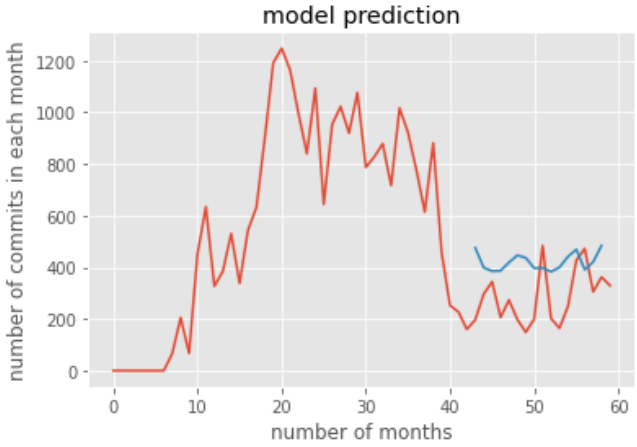

Fig. 7. Baidu's Commits 's LSTM Testing Prediction .

displays the RMSE values of the training and testing datasets for all metrics for Baidu.

2) Baidu's Commits 's LSTM Model Forecasting: For Baidu, out of 7 metrics, our forecasting shows the productivity of the team is going up based on the number of created

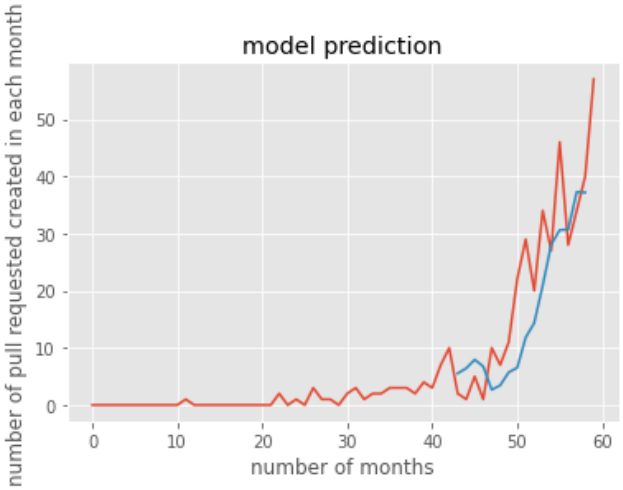

Fig. 8. Baidu's Created Pull Request 's LSTM Testing Prediction .

TABLE II
BAIDU'S OVERALL PRODUCTIVITY BASED ON THESE 7 METRICS

Metrics	Productivity
Number of Commits	Going down
Number of Created Pull Requests Comments	Going Up
Number of Updated Pull Requests Comments	Going Up
Number of Created Pull Requests	Going Up
Number of Updated Pull Requests	Going Up
Pull Request created and closed per month	Highly over fitted(invalid)
Number of Bug-x commits	No effect

TABLE III
TENSOR FLOW'S OVERALL PRODUCTIVITY BASED ON THESE 7 METRICS

Metrics	Productivity
Number of Commits	Going Up
Number of Created Pull Requests Comments	Going Up
Number of Updated Pull Requests Comments	No effect
Number of Created Pull Requests	Going Up
Number of Updated Pull Requests	No effect
Pull Request created and closed per month	No effect
Number of Bug-x commits	Going Down

pull requests, created pull request comments, updated pull request, and updated pull request comments. Number of Bug-x commits didn't have any changes/minor changes over time. And pull requests created and closed per month forecasting model was invalid as the model was results were incorrect. As mentioned above, a similar analysis is performed on our remaining 3 GitHub repositories; the results are shown below in 3 separate Tables.

For Tensorflow, out of 7 metrics, our forecasting shows team productivity is going up based on the number of commits, created pull requests, and created pull request comments. Three metrics didn't have any changes/minor changes over time. Metric reducing the productivity of the team is due to the decrease of bug-fix commits.

TABLE IV
HOME ASSISTANT'S OVERALL PRODUCTIVITY BASED ON THESE 7 METRICS

Metrics	Productivity
Number of Commits	Going up(Incorrect)
Number of Created Pull Requests Comments	Going Down
Number of Updated Pull Requests Comments	No effect
Number of Created Pull Requests	Going Down
Number of Updated Pull Requests	No effect
Pull Request created and closed per month	No effect
Number of Bug-x commits	Going Down

TABLE V
MAGNETO'S OVERALL PRODUCTIVITY BASED ON THESE 7 METRICS

Metrics	Productivity
Number of Commits	Going down
Number of Created Pull Requests Comments	Going up
Number of Updated Pull Requests Comments	No effect
Number of Created Pull Requests	Going Up
Number of Updated Pull Requests	No effect
Pull Request created and closed per month	No effect
Number of Bug-x commits	Going Down

TABLE VI
FUTURE PRODUCTIVITY FORECASTING OF 4 DEVELOPMENT TEAMS BASED ON 7 METRICS

Repositories	Developer's Team Productivity
Baidu	Will go up
Tensorflow	Will go up
Home Assistance	Will go down
Magneto	Will remain unchanged

For HomeAssistance, out of 7 metrics, our forecasting shows team productivity is going down based on the number of created pull requests, created pull request comments, and the number of bug-fix commits. The number of commits forecasting should have gone down but due to less amount of data, it's not giving good accuracy. The number of Updated Pull Requests, Number of Updated Pull and Pull Requests created and closed per month were invalid as there was no data available till 2020. Overall, three metrics were going down, few pull requests were updated and the rate of updating a pull request that was created in the same month was very low. So we can say that the productivity of this development team will go down in the future.

For Magneto, out of seven metrics, our forecasting shows neither team productivity is going up nor going down based on four metrics. Out of these four metrics, the number of created pull requests and number of pull requests is going up, and the number of commits and number of bug-fix commits is going down. Other remaining metric was unchanged and was unable to make any verdict for this development team productivity.

3) Implications: For Tensorflow, developers should focus on dealing with the bugs and fix them more often to improve productivity. Overall, Tensorflow productivity will go up in the future as the number of metrics with productivity trend going up than metrics with productivity trend going down are more in number.

For Baidu, developers should focus on committing more often. Overall, Baidu's productivity will go up in the future as the number of metrics with productivity trend going up than metrics with productivity trend going down are more in number.

For HomeAssistance, developers should focus more on updating pull requests. There are very few pull requests created and updated in the same month, so developers should focus more on quickly updating the pull request. Overall, HomeAssistance productivity will go down in the future as the number of metrics with productivity trend going up than metrics with productivity trend going down are less in number.

For Magneto, developers should focus more on increasing the commits and fixing bugs more often. Overall, Magneto's productivity will remain constant as the number of metrics with productivity trends going up is equal to the number of metrics with productivity trends going down.

4) Comparison of this proposed solution with other related approaches: The proposed solution for this work is a noble work. Thus, we couldn't compare our solution with any other previous work. However, two papers mentioned in our

Our Proposed Solution	Other's Proposed Solution
1. We have used LSTM setting our time step to 4 , to monitor how the productivity is changing over a given period of time. 2. Productivity Comparison tools : LSTM Model Prediction on test set 3. Build a prediction model that can predict future productivity of software development teams 4. Final Outcome : Build a productivity machine learning prediction model that can predict future productivity based on 7 metrics	1. They grouped the dataset into a 2 time period to see how productivity is evolving from one time period to the next one. 2. Productivity Comparison tools : Wilcoxon rank sum test, Pairwise two-sample t-test and cliff delta 3. Did not proposed any prediction model 4. Final Outcome : Statistical analysis of each metric between two time period to analyse productivity changes

Fig. 9. Our Proposed Solution Vs Other's Proposed Solution.

literature review had some similarities with our works regarding the procedure to find the difference between developer's productivity that change over time shown below in Figure 9.

V. Threats To Validity

A. Internal Threats

Internal validity concerns factors that could have influenced our results. Out of metric, we were only able to define 7metrics from our dataset as the data we will need to define these metrics were missing from public GitHub repositories. This threat can be eliminated if we can work with the private software repository of a running software development team. However, we were able to define the necessary metrics which make this research valid and aligned with similar studies by other researchers who explained these metrics in their work. Lstm works well only when there is enough data to train. However, in this study, the are some metrics that we have selected have little or no changes with time. As a result, our LSTM model was not working very well due to insufficient data. To avoid this threat, instead of considering the whole dataset, we have just considered the year where significant changes occurred with time. And based on this we can forecast its future condition. After making this slight change, our LSTM model accuracy improved drastically. It can also improve more if we go for a day to day changes of each month instead of month to month analysis of each year which we are planning to include in our future work

B. Construct Threat

Construct validity considers the relationship between theory and observation. We are considering our productivity based on forecasting of 7 metrics. We assumed if the summation of the number of metrics with positive productivity is more than the summation of the number of metrics with negative productivity trends, this development team productivity will improve in the future. It raises a threat to our research as we are using our assumptions for our final prediction. To avoid this threat, we mentioned that this model prediction is only base on seven metrics. The main reason we conduct this research was to find a way that predicts future developer's productivity, we can add or modified metrics later. Secondly, we didn't mention exactly how much the productivity is changing; if the change is very big or small, we didn't back our prediction with an exact numerical value. To avoid this threat, we manually checked out predicted values and checked our predicted graph, and based on these two results we came up with our final analysis.

C. External Threats

Threats to external validity concern the generalization of our findings. In this study, we extracted data from GitHub public repositories. We could not consider the private software repositories for our analysis because datasets were not available publicly. However, we believe that our study allows the generalizability since GitHub is one of the most popular forums where developers put their whole work, update it, and work there with other developers. So this works in a similar way like developers work in private software repositories. Lastly, another threat to the validity of this paper is we only considered 12 metrics that are co-related to developer's productivity. But in reality, there are other important factors too that affect a developer's productivity. So we tried to avoid this threat by working with already defined metrics from recent academic research papers that examined metrics related to developer's productivity.

VI. Conclusion

In this paper, we analyze the 12 most important metrics which are co-related with developer's team productivity. This paper also proposed a recurrent neural network developer's team prediction model that forecasts the future productivity of a software team based on 7 defined metrics. Our model was trained and tested on 4 public GitHub repositories on every 7 metrics separately. We find out that Baidu's and TensorFlow team productivity will go up in the future whereas it will go down for Home Assistant's development team and the productivity level will remain unchanged for Magneto's development team based on 7 metric' productivity. After this, we also suggest individual analysis that can improve the performance of each development team based on the model's performance.

VII. Future Work

This paper recommends the following to improve future work:

1) Mining 100 GitHub repositories and extending the research for high-quality analysis.
2) Increasing the number of metrics related to productivity and considered them as important attributes to build prediction models.
3) Include a filtering process when selecting a public repository. For example, we need repositories that are old and active in their operation for better accuracy of the LSTM model.
4) Include model forecasting on a future unseen dataset that is not part of training as well as test dataset.
5) Defining the level of productivity of each development's team.

6) Introducing qualitative analysis such as interviewing the developers who were involved in public GitHub repositories we have selected for predicting their productivity.
7) Applying similar studies on private repositories or software development team data for better prediction and analysis.
8) Training model on day-wise sample data instead of month-wise data points.

The authors would like to thank...

REFERENCES

[1] S. Sector and Telecommunications, "Canadian ict sector profile 2019," Jun 2020. [Online]. Available: https://www.ic.gc.ca/eic/site/ict-tic.nsf/eng/h$_it07229.html$

[2] E. Oliveira, E. Fernandes, I. Steinmacher, M. Cristo, T. Conte, and A. Garcia, "Code and commit metrics of developer productivity: a study on team leaders perceptions," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2519–2549, 2020.

[3] L. Bao, T. Li, X. Xia, K. Zhu, H. Li, and X. Yang, "How does working from home affect developer productivity?–a case study of baidu during covid-19 pandemic," *arXiv preprint arXiv:2005.13167*, 2020.

[4] P. A. d. M. S. Neto, U. A. Mannan, E. S. de Almeida, N. Nagappan, D. Lo, P. S. Kochhar, C. Gao, and I. Ahmed, "A deep dive on the impact of covid-19 in software development," *arXiv preprint arXiv:2008.07048*, 2020.

[5] Z. Liao, X. Qi, Y. Zhang, X. Fan, and Y. Zhou, "How to evaluate the productivity of software ecosystem: A case study in github," *Scientific Programming*, vol. 2020, 2020.

[6] V. Markovtsev and W. Long, "Public git archive: a big code dataset for all," in *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018, pp. 34–37.

[7] F. Qian and X. Chen, "Stock prediction based on lstm under different stability," in *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, 2019, pp. 483–486.

[8] O. published by Ake Gaviar on, "Githubs top 100 most valuable repositories out of 96 million." [Online]. Available: https://hackernoon.com/githubs-top-100-most-valuable-repositories-out-of-96-million-bb48caa9eb0b

[9] X. Karaj, "Introduction to git data extraction and analysis in python," Dec 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-git-data-extraction-and-analysis-in-python-e7e2bf9b4606

[10] J. Brownlee, "A gentle introduction to long short-term memory networks by the experts," Feb 2020. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[11] "Understanding lstm networks." [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[12] "An introduction to rnn and lstm," Oct 2020. [Online]. Available: https://www.youtube.com/watch?v=Mdp5pAKNNW4

[13] krishnaik06, "Stock market forecasting," May 2020. [Online]. Available: https://github.com/krishnaik06/Stock-MArket-Forecasting/blob/master/Untitled.ipynb

[14] "python for microscopists." [Online]. Available: https://github.com/bnsreenu/python$_for_microscopists, journal = GitHub, author = Bnsreenu$