The Impact of AI powered code completion in the software engineering field.

Niklas Reini

Haaga-Helia University of Applied Sciences

Thesis

2021

Bachelor of Business administration

Abstract

Author(s)
Niklas Reini

Number of pages and appendix pages
16 + 2

Abstract

Artificial intelligence is becoming smarter everyday and is now starting to find its way into programming through tools called automatic code completers. These tools vary in complexity, from completing the last words of a common function to generating hundreds of lines of code from a typed out description.

Currently AI powered code completers are useful for programmers, and will replace some of the more simple tasks in the future. However, since software engineering is an ever evolving field, programmers will always have some sort of job, even along side of artificial intelligence.

This thesis aims to resolve the extent to which an easily accessible AI powered code completion tool can impact the software engineering field. To find out the capablilty of a tool like this, a case study utilizing experimentation is conducted using an AI powered tool called Tabnine. The results show that in more simpler tasks, the tool can be extremely helpful and speed up the process of coding.

Keywords
AI (artificial intelligence), machine learning, neural network, data poisoning, auto completion.

Table of contents

1 Introduction

Artificial intelligence has become one of the biggest buzzwords in the last decade, it is becoming a bigger part of society and is slowly intertwining itself into everyone's careers. With the possibilities of deeper automation that AI provides, people are worried about simple jobs being replaced. However, now even jobs like software engineering could be at risk, or are they? (De Cremer, D & Kasporov, G. 2021)

Software engineering is an ever-evolving field which seems like the last place that AI would threaten. Software engineers are the ones who created AI in the first place.
Even though Artificial intelligence is the creation of programming, there are now AI created specifically for code automation. And this is not some special secret tool that only people at the top level can use, anyone could install an extension on a text editor like Visual Studio Code Editor, also commonly known as VSCODE that automatically completes one's code. One example of this is the "Tabnine" extension. (Tabnine. 2020.) (Microsoft. 2022)

Now, to what extent can AI replace programmers? This is an interesting phenomenon to assess, especially due to the irony from the creators of AI being replaced by what they created themselves. This subject is important to research because we need to always be one step ahead of AI, we need to be able to predict what kind of impact it can have on our future. Computers can do thousands of tasks at once, hence the thought of artificial intelligence being able to program by itself should sound Intimidating.

The question at hand here is about to what extent artificial intelligence can replace the jobs of average programmers, not if it will replace them totally. For this reason, the first objective of this thesis is to nullify the fact that AI will replace programmers totally, at least in the next decades. The job of a programmer will also need to be heavily researched to understand what kind of tasks there are to be replaced. Also, current as well as upcoming code automation AI will be subject to strong research to establish what is currently happening and what will happen in the next couple of decades. There is not much existing information of these tools, thus existing research as well as research created for this project shall be used. Experiments with AI created solely for this project will be also utilized.

2 Background

Artificial intelligence powered automatic code completion in coding has not been researched for as long as other subjects on artificial inteligence e.g., AI replacing profession in other fields. A major factor is that artificial intelligence started rising as an idea ever since the first successful Ai was created in 1951. (Copeland, B. 2022.) And almost 50 years later in 1996 it was used in code completion with the invention of IntelliSense. (Microsoft. 2016). Therefore, at least a surface level understanding of AI principles is a needed achievement.

This thesis will investigate to what extent AI or “code completers” can replace the job of a software engineer. To answer this question three areas, need to be researched, how AI operates, how much AI code completion tools can accomplish, and what kind of tasks and roles a software engineer has. The least time and resource consuming, however thorough part of the research will be on how AI operates, this is because, this thesis does not aim to explain how AI powered code completion tools work on the deeper level, the focus is more on how they can complete everyday tasks for programmer. An understanding of the big picture is far more important. Deep knowledge on the most complex mechanics of AI is outside the scope of this paper. After establishing a simple base line knowledge on how AI works, experimentation can be done with an AI powered code completion tool. This can then be put in contrast with the role and tasks of the average software engineer, consequently allowing us to find out the extent to which AI powered code completers can replace the tasks of programmers.

2.1 Artificial Intelligence

Artificial Intelligence is when computers operate like humans. Normally computers and machines do not think or learn, they do exactly what they are told to do. This works well with simple tasks like calculating numbers, however, some tasks are too complex for regular algorithms. This is where AI comes in, it allows for computers to learn from trying again and again. For AI to work, it needs to be trained, this is done by giving the computer a large amount of data where it is given examples of the tasks and the correct results. As a simple example, the data could be two different pictures where one is a fruit and one is a vegetable, from the given correct answers, the computer learns over time which one is the fruit, and which one is not. It learns this even though it has no instructions on how to differentiate a fruit and a vegetable. The reason why this simple method works is because computers can go through millions of these examples in minutes, resulting in them being smarter at what they do than humans. (CSU Global. 2021.)

What separates different AI is how they take the data they are given; this is where machine learning comes in. According to Columbia State University machine learning is “A specific application of AI that lets computer systems, programs, or applications learn automatically and develop better results based on experience” (CSU Global. 2021.)

The pattern recognition of AI is a result of neural networks. A neural network is a key player in what powers AI, it mimics the neurons in a human brain. As seen below in Figure 1 which demonstrates the layers of a neural network, the neural network is built up of nodes which send and receive data, each of them has a specific value, which if reached the node will pass data. The neural network is the result from all the learning the AI has done from the data it has been given (IBM Cloud Education. 2020.)

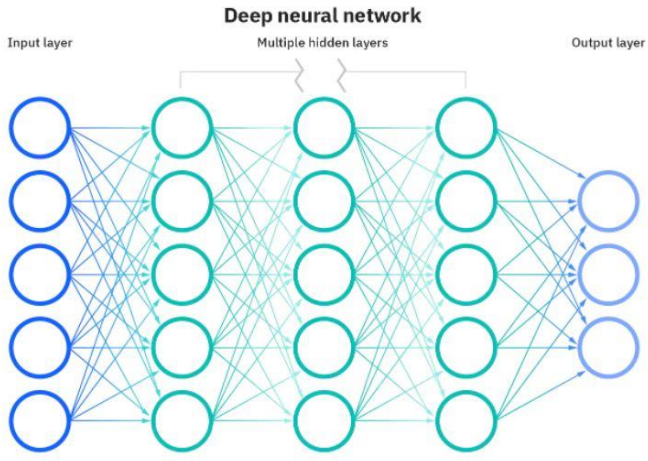


Figure 1 Neural network (IBM Cloud Education. 2020.)

2.2 Safety

The way that autocompletion knows what the most likely outcome is, starts with a huge amount of data. This data is most often gathered from open-source code repositories, this is a smart way of gathering lots of data for free, however it does come with a security concern. Anyone can add code to these repositories which is then used to train the AI, therefore anyone could tamper with how the AI learns. If the learning method of the AI is tampered with enough it could result in autocompletions with malicious intent. These in turn could pose a security risk for the company, this method of malicious intent is known as a data poisoning attack. (Schuster, R & Song, C. 2021.)

Another type of poisoning attack is a model poisoning attack, this is where the attacker affects the learning model directly instead of the inputted data. This requires a more difficult access, because only trusted programmers can change the model, however the risk outweighs the possibility. (Schuster, R & Song, C. 2021.)

According to Usenix, a computing systems administration “Powerful natural-language models improve the quality of code autocompletion but also introduce new security risks”. (Schuster, R & Song, C. 2021.) From this it can be gathered that currently, safety is not a huge risk in AI autocompletion, however, as these tools get smarted and complete a larger amount of code, the risk will increase exponentially, therefore safety needs to be considered even in these early stages of AI in coding.

There is huge risk that comes with artificial intelligence in general, in terms of safety and being able to control it. The risk of having suggested code that has been tampered with poisoning attacks from the outside, can lead to drastic changes in the outcome of the software being developed. Software controls so much of our lives that is not to be taken lightly. Perhaps even backdoor access could be created from these attacks. Big companies working with projects of high importance will most likely hesitate in adopting this technology, therefore slowing the progress of AI integrating into the job of a programmer. (Schuster, R & Song, C. 2021.)

3 Theoretical background

The theoretical background explains that AI will not fully replace programming and will work alongside software engineers to help in their tasks. Thus, below are some explanations of the crucial research required to fully answer the question at hand; To which extent can AI powered code completion replace tasks and even parts of a software engineers' job?

To get an initial overview of AI in coding, and all things AI, a thesis from a previous student of Haaga-Helia; "Artificial Intelligence and auto-generation of code" (Cheng, J. 2018) will be studied. From this thesis we can gather that AI uses machine learning to learn and neural networks allow this to be possible, neural networks mimic how a humans brain solves problems.

Another piece of research that will be crucial to answering the question is a paper from the International Journal of Information Technology and Language Studies; "The Future of Software Engineering by 2050s". (Zohair, L. 2018.) From this paper we learn what the field of software engineering will be like in the future. The need for applications and updates to those applications is rising and rising, this is a result of everything in our world being digitized and automated this paper covers how that will impact the work of software engineers in the future. This is a vastly important part of the research because it is required to understand exactly what the AI will be possibly replacing. If there is a deep understanding of the capabilities of Artificial Intelligence, but no understanding of the job of programmers, the question at hand cannot be answered.

Because neural networks are a building block of how artificial intelligence operates. This project will be using a research article from Hindawi talking about how a neural network-based support for code completion can help programmers "A Neural Network Based Intelligent Support Model for Program Code Completion" (Rahman, M & Watanobe, Y & Namakura, K. 2020.) From This research we gain an understanding of how a deep neural network-based support model for code completion can create a huge advantage for software engineers, especially students studying programming. This could be done by detecting more errors than what normal code completion tools without neural networks can.

It would not be wise to approach this paper without thinking of possible safety concerns. One possible safety concern that could hinder the progress of this automatic code completion taking over programming is data poisoning. This project will be using Usenix's research on how poisoning attacks can affect the completed code made by these AI code completion tools. "You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion" (Schuster, R & Song, C. 2021.) Code completion tools utilizing neural networks can be extremely vulnerable to poisoning attacks. Poisoning is done by adding unwanted data to the training process of AI, this will result in a direct change of the outputted code from the AI. This is crucial part to my research because safety in the IT world is taken very seriously, especially with AI. This could heavily impact the research phenomenon of how much AI can replace the job of software engineers. Will laws be put in place to completely put a halt on this?

It is of importance to also get a look at not just AI in general but the tools it powers. To get a more specific look at a code completion tool, research on a tool called Dompletion will be done. it is a DOM-Aware code completion tool made for JavaScript. "Dompletion: DOM-Aware JavaScript Code Completion" (Bajaj, K & Pattabiraman, K & Mesbah, A. 2014.)From this article we can learn how creating a code completion tool that is aware of the Document object model can be challenging. The technique proposed here analyses JavaScript code and provided completed code that uses DOMs APIS to interact with it. The precision of this tool to provide accurate completed code is 90% This article is on the older side of the research, but an important one to get the big picture of how code completion works, what it needs as an input and what challenges come with it. After achieving a fundamental understanding of AI tools in coding, the experimentation for this paper can be started.

4 Research method

This thesis uses a case study which includes experimenting with an existing AI code completion tool. Case studies are crucial to the research as they allow to get an overall understanding of the big picture so that further research can be efficient. Case studies as defined by Rhee, Y. s “an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident…[and] relies on multiple sources of evidence” Rhee, Y. (2004). AI code completion experimentation is conducted so that a thorough understanding of of what they achieve can be gathered. These experiments are coupled with a look at what’s to come soon. GitHub Co-pilot is an upcoming AI code completion tool which is more powerful than the ones featured in the experiments, therefore I t is a perfect gauge to assess how the AI will progress forwards. (GitHub, Inc. 2021)

To understand the capabilities of AI code completers, experimentation will be conducted, these experiments will be conducted with Tabnine. It is a modern code completion tool powered by artificial intelligence, released in 2018. The experiments conducted with Tabnine will take an input from the programmer and notes will be taken on how complex the resulting code is, and how much of the intended function or algorithm it completes. To get quantitative data from these experiments, the number of characters used for each experiment is recorded as well as the number of autocompleted characters. As a result, we get the completion percentage of each experiment (Tabnine. 2018.)

To begin, a simple “hello world” type experiment will be conducted. In programming “hello world” is used to describe the first piece of code that is written when learning or using something new. This piece of code is always simple and short, most often the resulting code will print out the words “hello world”, hence the name.

The experiments will continue to get more complex, to the point where a clear pattern can be seen, or the code completion tool is not providing a significant advantage. Or even starts hindering the job of the programmer. The programming language used for these experiments will be JavaScript, this will not be changed throughout the experiments as it would tamper with the legitimacy of the results.

5 Experimentation

Listed below in table 1 are the experiments that will be conducted for this paper, on the left side are the names, in the middle is the code that is required to achieve the results and lastly on the left is the number of characters needed.

Experiments	Expected code in JavaScript	Expected characted amount
Hello world	console.log("hello world");	27

Multiplication	var number1 = 1;
	var total = number1 * 5;
	console.log(total);
		63

Loop Array	var myArray = [0,1,2,3,4,5,6,7,8,9,10];
	for (var i = 0; i < myArray.length; i++){
	console.log(myArray[i]);
	}
		117

Addittion	var number = 1;
	var secondNumber = 2;
	var thirdNumber = 3;
	var total = number + secondNumber + thirdNumber;
	console.log("total");
		127

Duplicate Array	function duplicateArray(){
	var array = [1,2,3,4,5,6,7,8,9,10];
	var newArray = new Array(array.length);
	for (var i = 0; i < array.length; i++){
	newArray[i] = array[i];
	}
	for (var i = 0; i < newArray.length; i++){
	console.log(newArray[i]);
	}
	}
	duplicateArray();
		306

Table 1 Experiments

6 AI autocompletion

Before conducting any experiments with AI autocompletion, we must have a thorough understanding of how they operate. Different code completion tools use AI to a varied extent, some use deep learning, meanwhile some use a simple model of commonly used functions and algorithms. The Tabnine tool was chosen for this paper because it fits in the middle of code completion tools that use little to no AI and ones that use it extensively. The way Tabnine actually works is that it learns from a list of commonly used lines of code, but also trains to be better while the programmer or the whole team writes code. The more you write with the tool the better it gets. For the commonly used pieces of code the tool is reading what the programmer is typing and making calculations on which one is the most likely to be what the programmer needs (Tabnine. 2020). Below in figure 2 is an example of the suggestions that these AI auto completers can provide, this is specifically from the duplicate array experiment. From just typing “fo” we get the grayed-out suggestion that can be completed by pressing tab. It correctly assumed that the programmer wanted to loop through the list of numbers.

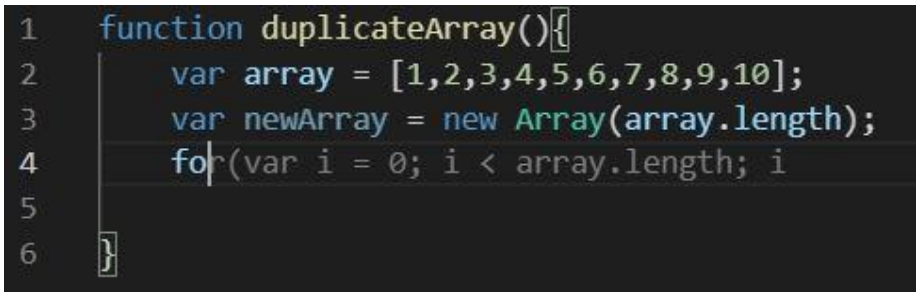


Figure 2 AI autocompletion

7 Results

The five different experiments yielded fairly similar results, however, there is some crucial differences that can be seen. The lowest completion rate was 48.82%, this means that the completion tool wrote almost half of the code. The highest completion rate on the other had was 74.36%.

The result table below (table 2) shows the total number of characters that is needed to write the code for each experiment as well as how many of said characters were not needed to be written by the programmer but were automatically completed by the Tabnine tool. Lastly it also shows the percentage of the work that the auto completion tool wrote.

Table 2 experiment results

Experiments	**Expected characted amount**	**Autocompleted characters**	**AI completetion perentage**
Hello world	27	15	55.56%
Multiplication	63	31	49.21%
Loop Array	117	87	74.36%
Addittion	127	62	48.82%
Duplicate Array	306	181	59.15%

Starting with the simplest experiment; The “Hello world” experiment. As the programmer started typing “co..” the AI already suggested “console” This being a result of the console function in JavaScript being an extremely common tool for developers. After autocompleting “console” and typing “.” just behind it the autocomplete already suggests the whole code “console.log(“”)” Now we just need the “hello world” which surprisingly enough comes up as a suggestion after typing just the letter “h”. The hello world experiment had the third highest completion rate of 55.56%.

The multiplication experiment is where we see a change of behaviour, this is a result of having named variables. The name of the variables can be anything that the programmer decides, for an example a number variable could be just named “number” or “i” for integer. The AI only starts to give good predictions after naming multiple variables, this is because the AI starts to notice a pattern in the programmer’s naming scheme. In the beginning of the experiment the AI only suggests completions for “var” which stands for variable. In the end when typing “co” the AI suggests “console.log()” and when typing the letter “t” as the parameter the suggested variable is total. As a consequence of having more than half of the function being variables and programmer’s chose values, the completion rate falls to 49.21%.

The highest completion rate of 74.36% was with the Loop Array experiment. In this experiment the programmer is trying to print out the numbers in a list of 10. This is done with a loop function which goes through every number in the list and uses console.log to print it out. Programmers use the name array instead of list to not confuse it as another variable type. As the programmer starts with creating the array the AI suggests an array based on the name of the variable being “myArray”. When the programmer starts to type “1,2…“the Ai suggests the completed list of 10 numbers. In the next step we start to see how powerful this tool can be, when the programmer types “fo” the AI suggests the entire piece of code required for a loop ”for (var i = 0; i < myArray.length; i++)”. Next the programmer types “con” and the tool provides a suggestion for “console.log(myArray[i]);” which is all the programmer needs to finish the task. Here we see once again that the tool works well when the program consists of commonly used code and not many variable names.

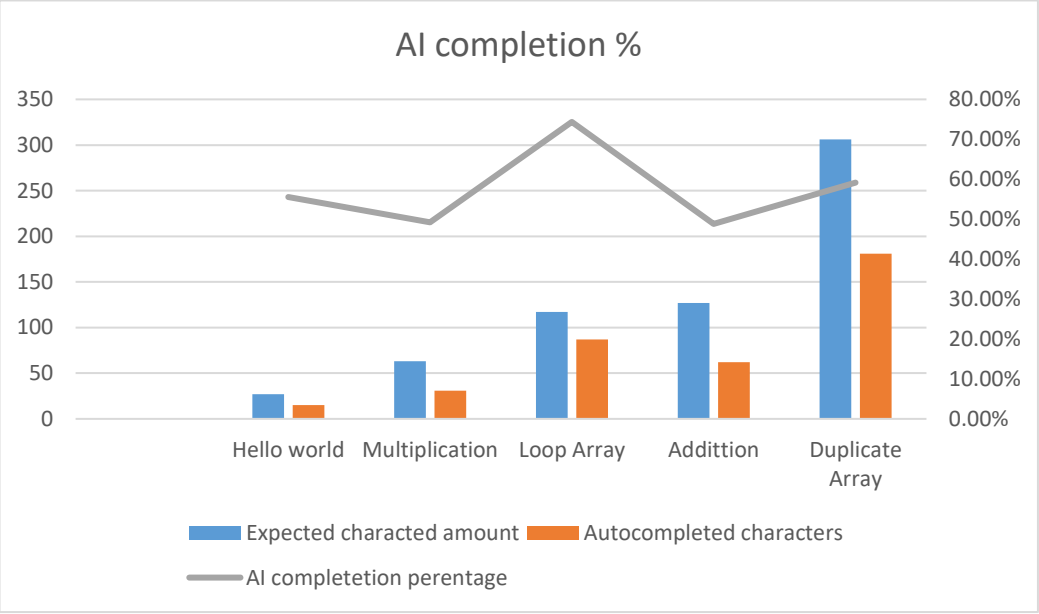
The addition experiment was very similar to the multiplication experiment, only difference being, more repetition. As a result, the completion rate was slightly higher than the multiplication experiment. One interesting point that stood out was that, when typing the calculation for the total, the AI suggested “var sum = number + secondNumber + thirdNumber;” already from just typing “var su”.

The most complex experiment Duplicate Array started to fit the pattern of higher receptiveness resulting in a higher completion rate. However, since this experiment had a higher complexity the completion rate was only 59.15% compared to the other array experiment. From this experiment we get almost the same suggestions like the “for loop”. When starting to write more code the programmer quickly notices how helpful the tool can be to save time. Instead of writing a whole list of numbers, the tool helps to autocomplete the entire list.

The average code completion percentage throughout these experiments was 57.42%, which is higher than the 30% that Tabnine claims. From this we can conclude that Tabnine is a great tool to help programmers write code faster, and an even greater tool for writing repetitive simple code. This also allows for beginner programmers to think more about the logic of their code instead of slowing their learning by being stuck on the syntax.

Figure 3 visualizes the AI completion percentage by each experiment ranging from the simples to the most complex.

Figure 3 AI completion percentage



As seen in figure 3 above, from the grey line, we cannot interpret a clear upward or downward trend in the completion percentage as the code complexity increases. This once again shows us how Tabnine works excellently at speeding up simple tasks, like naming variables and simple looping. Therefore, still leaving the logic of the code to be worked out by the programmer. Logic meaning the code as a whole, instead of specific words(syntax).

8 Conclusions

From the experiments and further research, we can conclude that extent of which AI will replace the job of the average programmer is negligible. However, AI powered code completion is a powerful tool that can vastly help programmers in speeding up their process in simple and repetitive tasks, such as iteration and naming, as well as avoiding errors. From this we can further conclude that beginner coders can learn more efficiently because they can focus more on what their code is doing instead all the small details.

Even if Artificial intelligence would improve drastically to the point where it was better at programming than the average software engineer, there would still be jobs for coding, just in different positions. Some example positions would be AI system moderators or AI developers. (Zohair, L. 2018.)

Lastly AI code completers are vulnerable to data poisoning attacks, there is risk that this might hinder the deployment of artificial intelligence in companies working on projects with high importance, such as government work or any other work that can have a big negative impact on people.

AI code completion tools also serve another purpose, which is error detection. Even though modern text editors and some languages do this job well, there is still a large room for improvement. Existing AI code completers have an error detection rate from 31% to 70% (Rahman, M & Watanobe, Y & Namakura, K. 2020) and a precision of roughly 90% (Bajaj, K & Pattabiraman, K & Mesbah, A. 2014.).

Lastly, from the results, we clearly see that code completion tools are the best at medium complexity code regarding beginner level coding. This is because, we can see the completion rate peaking at the middle of Table 2. Also, because at this level, the complexity is not too high whilst still having a high amount of variables and simple functions compared to the amount of characters in said code.

9 References

Albawi, S & Mohammed, & Al-Zawi, S. 2017. Understanding of a convolutional neural network URl: https://ieeexplore.ieee.org/abstract/document/8308186 Accessed: 9.14.2022.

Bajaj, K & Pattabiraman, K & Mesbah, A. 2014. Dompletion: DOM-aware JavaScript code completion. Vancouver. URL: https://dl.acm.org/doi/abs/10.1145/2642937.2642981 Accessed: 9.28.2022.

Copeland, B, 2022. artificial intelligence URL: https://www.britannica.com/technology/artificial-intelligence Accessed: 9.28.2022.

CSU Global. 2021. How Does AI Actually Work? URL: https://csuglobal.edu/blog/how-does-artificial-intelligence-actually-work#:~:text=AI%20systems%20work%20by%20combining,performance%20and%20develops%20additional%20expertise. Accessed: 9.28.2022.

De Cremer, D & Kasporov. G. AI Should Augment Human Intelligence, Not Replace It. URL: https://hbr.org/2021/03/ai-should-augment-human-intelligence-not-replace-it Accessed: 9.20.2022.

GitHub, Inc. 2021 Your AI pair programmer. https://copilot.github.com/ Accessed: 9.28.2022.

Cheng, J. 2018. Artificial Intelligence and auto-generation of code. Helsinki. URL: https://www.theseus.fi/bitstream/handle/10024/149252/report_v2.pdf;jsessionid=65AEF2E6852D8139EA19D9F6A95ADCC9?sequence=1 Accessed: 8.22.2022.

Herzing university. 2020. What Does a Programmer Do? URL: https://www.herzing.edu/description/computer-programmer Accessed: 9.18.2022.

IBM Cloud Education. 2020. Neural Networks. URL: https://www.ibm.com/cloud/learn/neural-networks Accessed: 9.28.2022.

IBM Cloud Education. 2020. Machine Learning. URL: https://www.ibm.com/cloud/learn/machine-learning Accessed: 9.28.2022.

Rahman, M & Watanobe, Y & Namakura, K. 2020. A Neural Network Based Intelligent Support Model for Program Code Completion. Fukushima. URL: https://www.hindawi.com/journals/sp/2020/7426461/ Accessed: 9.28.2022.

Stokdyk, D. 2021. What Do Programmers Do, Anyway? URL: https://www.snhu.edu/about-us/newsroom/stem/what-do-programmers-do Accessed: 9.10.2022.

Schuster, R & Song, C. 2021. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. New York. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/schuster Accessed: 9.28.2022.

Tabnine. 2018. AI Assistant for Development Teams URL: https://www.tabnine.com/ Accessed: 9.28.2022.

Microsoft. 2022. Visual Studio Code: https://code.visualstudio.com/ Accessed: 9.28.2022.

Zohair, L. 2018. The Future of Software Engineering by 2050s: Will AI Replace Software Engineers? Dubai. URL: https://journals.sfu.ca/ijitls/index.php/ijitls/article/viewFile/23/pdf Accessed: 9.20.2022.