GRAPHICALAI: USING GENERATIVE PROGRAMMING TO IMPROVE USER EXPERIENCE ON DEVELOPING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING MODELS

Andrew Shen[1] and Yu Sun[2]

[1]Beckman High School
[2]Cal Poly Pomona, Computer Science Department

ABSTRACT

GraphicalAI is an application served as an example to bridge the gap between domain experts who do not know how to create Artificial Intelligence (AI) and/or Machine Learning (ML), while also having a separate, simpler user interface (UI) to utilize the versatility of AI/ML in their domain. It is a proof of concept to show how a graphical application can aid domain experts to not only create AI models with ease, but also show the potential to be flexible in terms of greater choice and combinations of mathematical functions and mechanism, with balance, to provide a uniform application for the diverse domains. The recent addition of generative programming (GP) with the end-user interface creates an easier interface for the domain experts to utilize the AI models created and trained from the software.

KEYWORDS

Computer Science Education, Software, Graphical User Interface, Artificial Intelligence, Machine Learning, Generative Programming

1. INTRODUCTION

AI and ML has been a trending topic [2] in most parts of the industries in the recent decades, but more importantly, the domain experts (e.g., accountants, doctors, security experts, etc.) are realizing the capabilities of artificial intelligence to efficiently automate and aid tasks in their daily works. It has come to a point where if the business operation does not use AI in any aspects of them, it will likely give them a disadvantage to their competitors (e.g., if the competitor had not utilized AI in business marketing [3]). AI and ML has already been with us for a long time now, but we can lower the barrier of entry to creating AI models and utilizing ML to train models. This phenomenal sub-field of computer science has grown every more intrusive to almost every profession and tools used in current day's operation.

For a company, or any amount of people, to start utilizing the power of AI with ML, there are generally three potential methods to get started. First, we can start teaching ourselves with programming if the domain experts have the time to learn it. But for most domain experts, this could be problematic because not only could the complexity of programming conflicts with their field of study, but their work, projects, etc. could also add additional time pressure from learning programming. It is shown that collaboration between a domain experts and data scientists are needed when developing projects [9], which can also be under a corporation. Second, we can find a developer studying in the field of AI and/or ML to aid in the incorporation of AI into one's work. But regardless of whether the domain expert are paying the developer (which itself is its own hassle) or not, there will always be the chance for communication errors and human mistakes. Third, we can use external AI/ML services such as Microsoft Azure [4], Google Cloud [5], Amazon Web Services [6], etc. to help us, graphical or not. Now this is an interesting point since the paper is also talking about how its specific software could better serve this solution compared to other services.

GraphicalAI [7], the fourth solution, will help us better incorporate AI through ML into one's project. And not GraphicalAI specifically, but the inherent core idea it has been built on that is specific and apart: usability and flexibility. (Speed and scalability are also one of the core ideas, but we mainly focus on the other two.) Many services and applications in the third method only present themselves on speed and scalability but are lacking the flexibility without compromising the usability [15].

The current iteration of the GraphicalAI will be using generative programming [1] to better achieve its intended result of usability while maintaining its flexibility. GP is used for the application where the domain experts design their model, which the model passes its configuration to an end-user application. This end-user application will generate another application specifically to provide a cleaner and easier to use interface for that model specifically, with all the unnecessary UI component for that model removed.

To better show GraphicalAI's functionality and purpose, we will first show a motivating example to describe why GraphicalAI was created in the first place. Then, we further describe of what GraphicalAI does and how it alleviates and attempts to solve the problem. Next, we will show what GraphicalAI is all about by showing an example of step-by-step procedure on how to build a model and deploy it all inside GraphicalAI. Finally, we will show our user study to reveal how surveyors liked GraphicalAI, how GraphicalAI is different from potential other similar looking applications, and what is the next step for us.

2. MOTIVATING EXAMPLE

An example below will be shown to emphasize the point the unneeded complexity in designing an AI and ML model. In traditional programming, domain-experts will have to learn all the following programming concepts how they work (e.g., variables, modules, iterations, etc.), which for most domain experts will likely feel it is a not worth their time. The following example uses the Wine Quality dataset from Kaggle [14].

```
import pandas as pd
import tensorflow as tf

df_raw = pd.read_csv(".\\Testing-XVII\\resources\\winequalityN.csv")
df_raw.dropna(axis=0, inplace=True)  # removes any row in the dataset that contains missing values
df = pd.concat([df_raw.drop("type", axis=1), pd.get_dummies(df_raw["type"])], axis=1)  # convert a nominal data
into numerical dummy variables

ydt = df["quality"].to_numpy()
xdt = df.drop("quality", axis=1).to_numpy()

coef = tf.Variable([0.0 for _ in range(len(xdt[0]))])
bias = tf.Variable(0.0)

def linreg(x): return tf.math.reduce_sum(x*coef, axis=1) + bias  # our main AI model

def objv_mse(ypred, y): return sum((ypred - y) ** 2) / len(ypred)  # the loss function used to train the AI model

learning_rate = 0.000051  # manipulates the strength of the gradients affecting on the variables
for i in range(15):
    with tf.GradientTape(persistent=True) as g1:
        loss = objv_mse(linreg(xdt), ydt)

    [dl_dw, dl_db] = g1.gradient(loss, [coef, bias])
    coef.assign_sub(learning_rate*dl_dw)
    bias.assign_sub(learning_rate*dl_db)

print(linreg(xdt))  # our output

```

Figure 1. Wine Quality Trainer code example

The code shown above portrays the complexity and the difficulty for domain experts with minimal exposure to programming to create their own models. Considering this is only Python [8], where performance is subpar while provide easy syntax, shows us the resources needed to develop a more performant model, especially in production environment or large datasets. In contrast, the GraphicalAI uses UI to simplify the creation and utilization process of AI models, but also the ability to use more performant libraries, drivers, etc. to improve the speed without diminishing the software's usability.

3. SOLUTION: GENERATIVE PROGRAMMING

We want something that addresses the problems in each three methods stated in the first section. For the first method, if many domain experts want to utilize the potential of AI and ML in their projects, it is required that we provide an easy means for domain experts to achieve their goal without programming, which is currently the most common method to create AI and ML models. A common way for non-programmers, such as domain-experts, to utilize a tool that is not commonly found outside of programming is to create a separate domain-specific language (DSL) for the specific purpose to easily use that tool through a different mean [19]. In this case, a graphical domain-specific language will benefit domain-experts a lot to provide a common ground between them and the programming-specific tool AI and ML. Regarding the second method, as long domain-

experts are working largely by themselves on developing the model, it mostly addresses the problem in the second method. Lastly, for the third method, we want to create a graphical application (where our graphical DSL will reside in) to be flexible yet usable for many domain-experts, differentiating other GUI-based AI and ML model development applications.

GraphicalAI is a GUI-based application we envision with the goal of helping domain-experts create AI models and use ML with flexible choices and ease of use. The original iteration only developed the main application [20], whereas the second iteration had almost all the features of the first iteration, but with additional features utilizing Flutter to help ease the domain-expert the deployment process and model utilization.

The second iteration of GraphicalAI uses generative programming to further make the utilization of AI models palatable. It all starts off with separating UIs for model creation and model utilization, as both actions will never be used simultaneously. The separation does not necessarily mean a separate person will be working on it (or else it just defeats the purpose), but merely to make the utilization of a specific model more manageable with the less cluttering of the UI once the domain expert has finish modeling their models. To clarify, it is like when a person is working on a project, there is this one crucial complicated part the person wants to abstract out. So the person themself working on that complicated part can use the help of a tool to do it for them, instead of using alternative solutions.

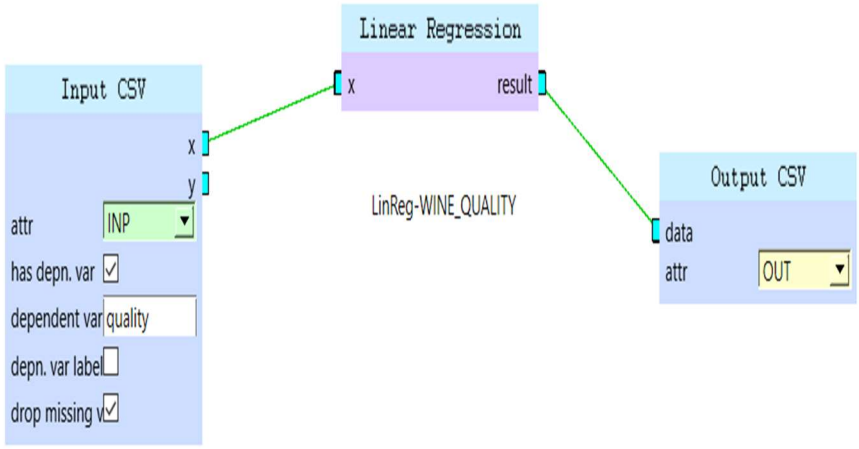


Figure 2. Model Visualization

The above shows an example of what a visualization of the model layout looks like for the Wine Quality training. The example shown is for the simplest case for demonstration purpose, but as the model grows more complex, the node-based design helps untangle the complexity.

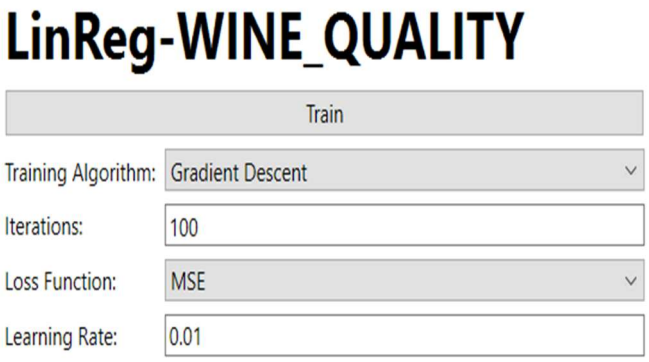


Figure 3. Model Training Options

The intuitiveness and the conciseness of configuring the training process contrasts the previous example shown via programming. One notable abstraction is the lack of presence of gradient descent (i.e. Tensorflow's GradientTape [11]) in GraphicalAI. Knowing that the application can handle the complicated, routine tasks, the domain experts can focus on building the model itself, while also able to benefit from using fast, performant libraries without the worries of development complexity.

Creating models and utilizing models are all important and easily use by domain experts. Model creation is where the actual building the model happens, it defines what inputs and outputs are needed, and how the model gets interacted with the outside world. The model utilization is when the user uses the model for the sole purpose to utilize and connect it with their work/projects.

Generative programming helps mitigates the unnecessary complexity for deploying a model through removing the unnecessary UI components that is not needed for that model, whereas in other case, the other types of models may need a different sets of UI components. For example, if a model only needs one input, we can simply show only one input inside the end-user application; if a model needs outputs, we can show the output location inside the end-user application. This allows much flexibility in creating models while also maintain a comparatively concise and clean UI for end-user to simply utilize the model, achieving the goal of flexibility while ease of use.

Figure 4. End User Application

For model creation application to transfer the parameters to the model utilization application, we need a system to mediate the transfer effectively. Since the model creation app and the model utilization app uses different languages, we found using a web

server (i.e. Python Flask [11]) helps transfer the parameters to direct the model utilization application which UI component to generate.

The end user application will able to use generative programming to only build necessary UI components for the developed AI model. And in GraphicalAI's case, the number attributes needed to build as required by the model.

4. GRAPHICALAI IN ACTION

To show what GraphicalAI is all about, we will show an example of how to train an Iris dataset using linear regression. It will go through all the steps required from developing a model with nodes to deploying a model.

To start, we must create a project before we can start creating models. Assuming we have no prior projects to load from, we will begin with creating a new project. In the figure below, we can see in the middle column (the other two columns have been cropped) we can name our project and choose where to save our project.

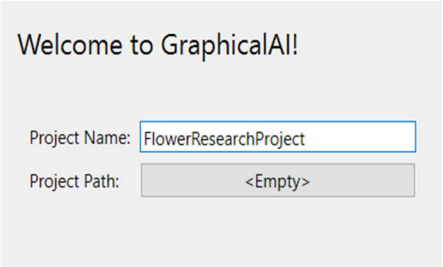


Figure 5. Project Creation Menu

Once we created a new project, we will be immediately greeted with an empty "unnamed" model page.

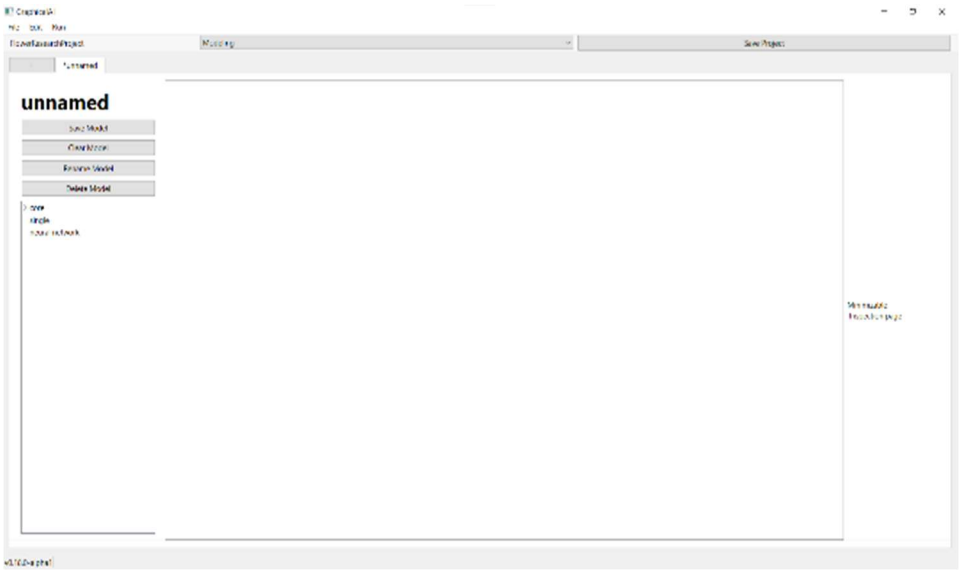


Figure 6. Empty Model Page

Now we will populate the empty model with nodes for training the Iris dataset [13] using a simple linear regression, as an example.

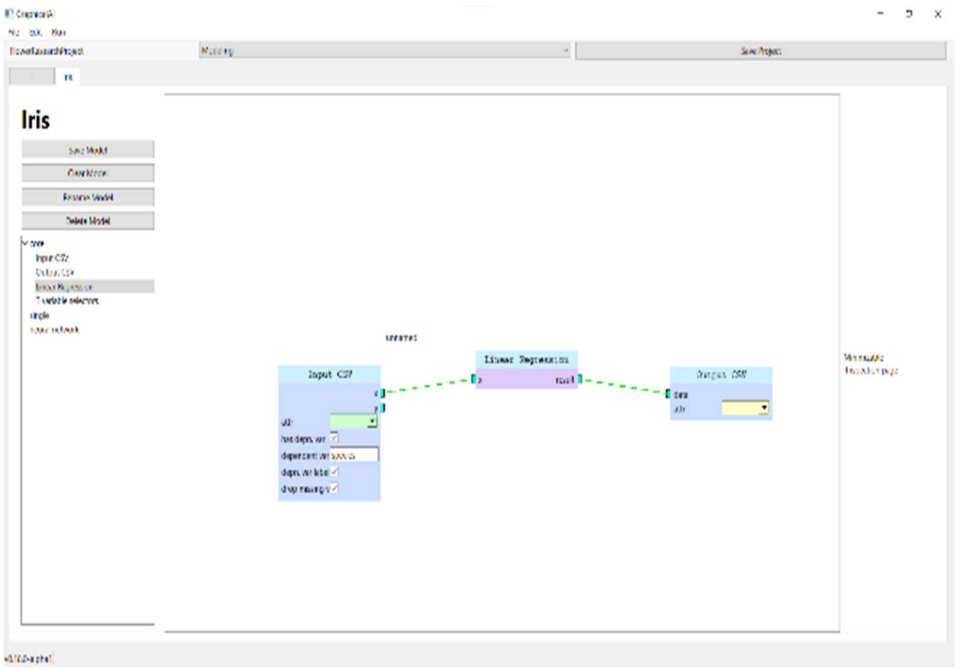


Figure 7. Populated "Iris" Model Page

Notice the nodes in sizeable blue rectangles. These are the basic units in a model separated by their functions to provide a flexible development of an AI model. And observe the green lines—connections—connecting the small squares protruding off the sides of the nodes—connectors. This overall help the domain-experts and other users easily conceptualize the function of their AI model. Onwards to the training section as shown below.

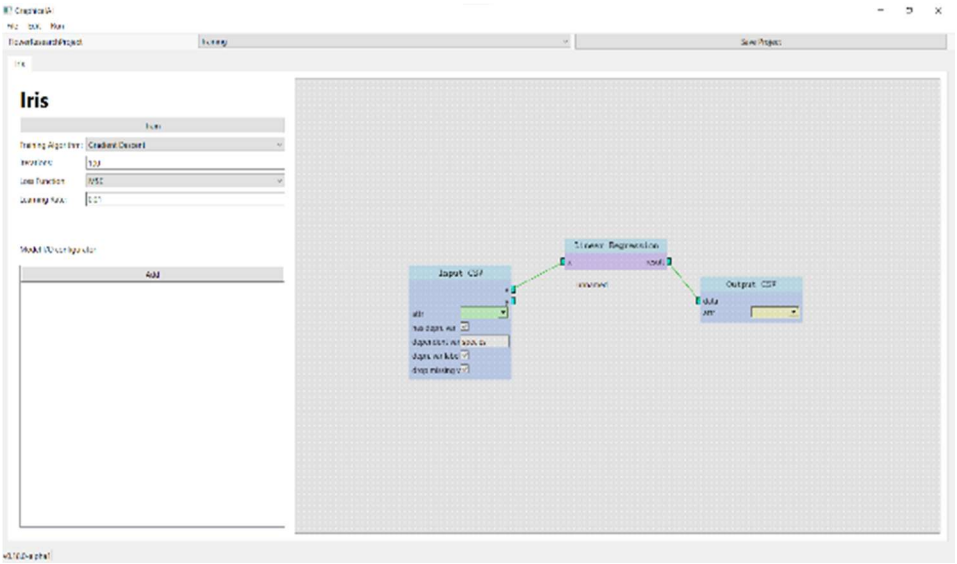


Figure 8. "Iris" Model Training Page

This is the prototype interface used for training the model. Notice the attributes in the bottom left region, these are dedicated regions to handle the location and the type of

inputs/outputs for the model, away from the model code itself. Adding a new attribute by clicking on the “Add” button will yield a modal window, asking for the attribute name and the type (e.g., Input or Output).

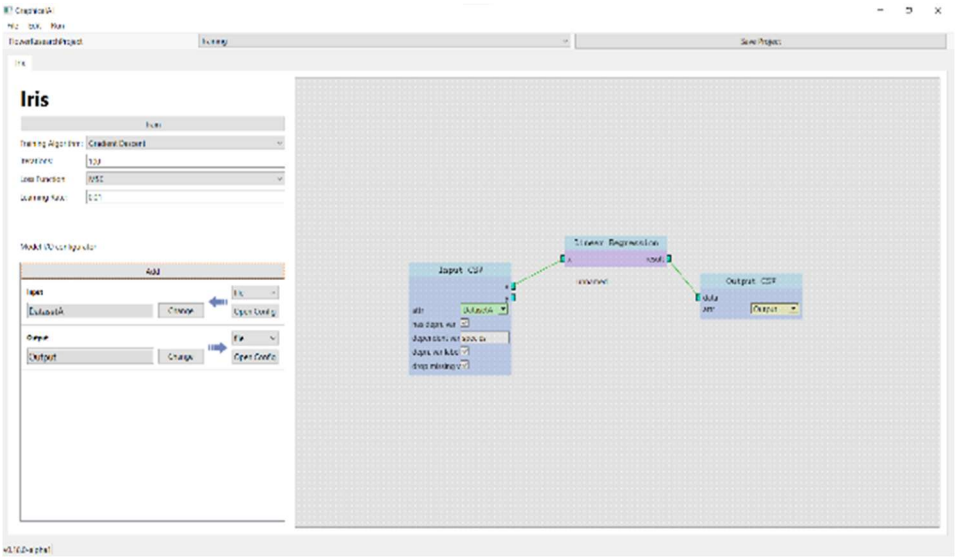


Figure 9. “Iris” Model Training Page with Input/Output Attributes

An example showing all the necessary Inputs and Output attributes. Since all of our attributes are file type, we must specify the location of our source and destination file for our inputs and outputs, respectively. Pressing the “Open Config” will open a modal to configure that one specific attribute. In this case, we are only changing which files the model will be reading from and writing to. Before going on to predict the model, training the model is required to record and train the model weights.

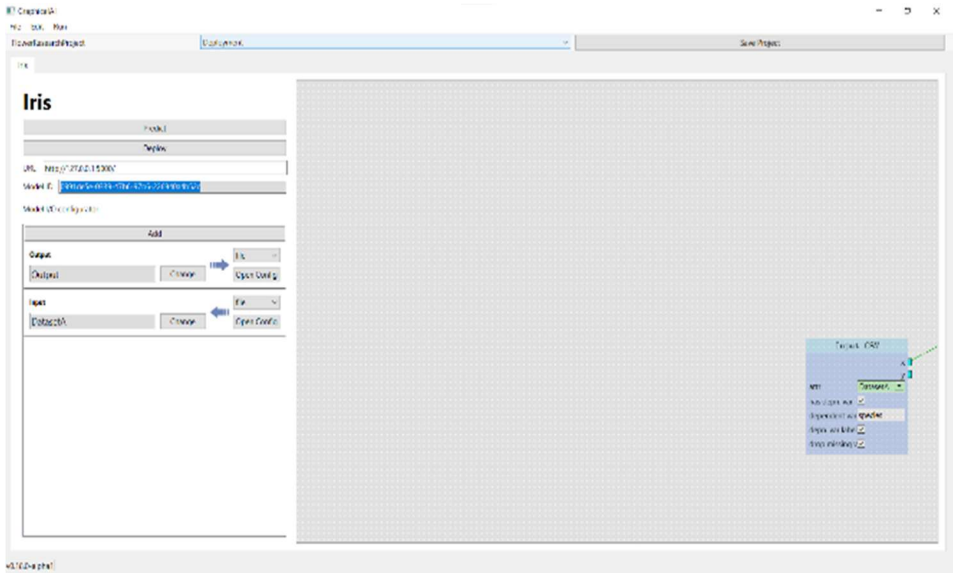


Figure 10. “Iris” Model Deployment Page

This is where we predict the model. The prediction page also includes a list of same attributes, except with different configuration to allow model training and model

predicting to use different data. The left-side menu also shows the input where we can enter the server location, and then it will return and show us the model key for this model once we have pressed the "Deploy" button. Deploying the model will copy and transfer all its model data to the local server (for demonstration purposes, it is running on the localhost), where the end user application, the Flutter [12] application, can read the model data and show the required attributes to the interface.

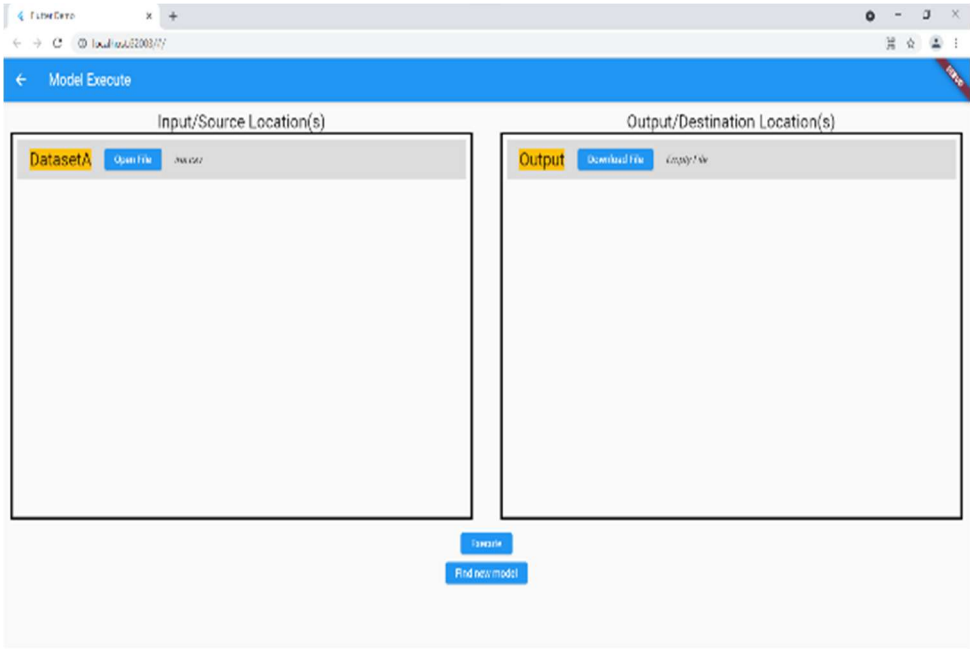


Figure 11. End User Application Interface

Before entering this specific page as shown from the figure above, we must go through a preceding page to enter the model key to get the page with the specific model we have trained for. With this separate but very similar functionality, in terms of *utilizing* the model, this reduces the complexity compared to the program used to also create the model. This is a way for domain experts to easily distribute their model and hide the unnecessary visual clutter of the models when its not needed.

This specific page as shown above requires the context of what model it is working with, so before arriving the specific page as shown above, the user must go through a separate page (not shown) to enter the model key and it will then retrieve the specified model. This page still offers similar functionality compared to the previous page used to create the model, in terms of what the user can *use* the model for, thus reducing the complexity in the deployment process. This is in a way for users to distribute their model easier and remove the visual clutter of the models used to only *create* the model.

5. USER STUDY

[Please finish this section with just a small amount of informal user study. Please refer to the previous paper for this section.]

The user study largely consists of younger users mainly ranging from ten years old to thirty years old. The people who surveyed the form does include substantial interests/major/profession in computer science (CS) or related, though people are not in

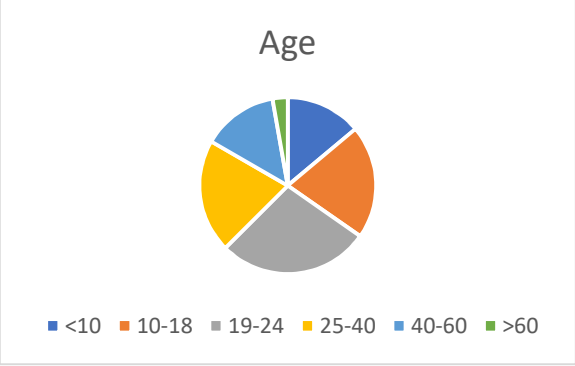
CS are largely to emphasize to show GraphicalAI serves its purpose. Among the people who do not show CS in any of their aspects has an average of 1-2 out of 5 rating of their knowledge on programming. Separately, among those who are in CS rate themselves largely a 5, meaning they know a good knowledge on programming. And among them, their most comfortable programming languages are apparently C++ and Java while Python comes a near second, and notably occasional other niche programming languages such as Julia…

Now going towards their experience on using GraphicalAI, many non-programmers like the experience of the GraphicalAI while some programmers, especially those who are in AI and ML, feel the application is still limiting. And that's fair knowing AI and ML is most flexible when programmed through a programming language. Most non-programmers feel it was easy to use and setting-up models, especially after their first try. Many non-programmers like the aspect of nodes and connectors connecting to form a flexible model.

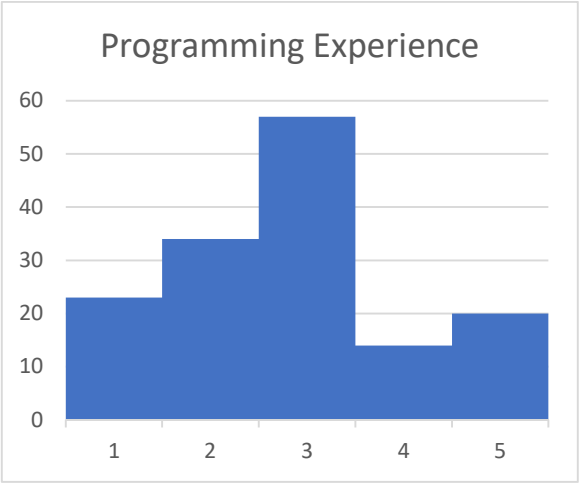
So connecting GraphicalAI or GUI AI/ML application to programming, many non-programmers feel they like GraphicalAI or GUI AI/ML a lot, compared to what they believe what programming AI/ML will be while programmers feel its limiting, even for other GUI AI/ML applications.

In conclusion, we can say GraphicalAI has served its purpose by provided a usable interface while providing flexible options of nodes to build models.

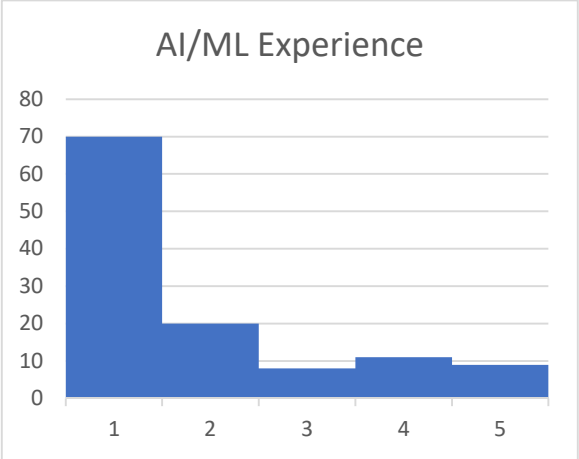
This is the survey we conducted. Now to start off the analysis, we will be going over the introductory questions to learn more about our surveyed people's background and experience with computer science in general.

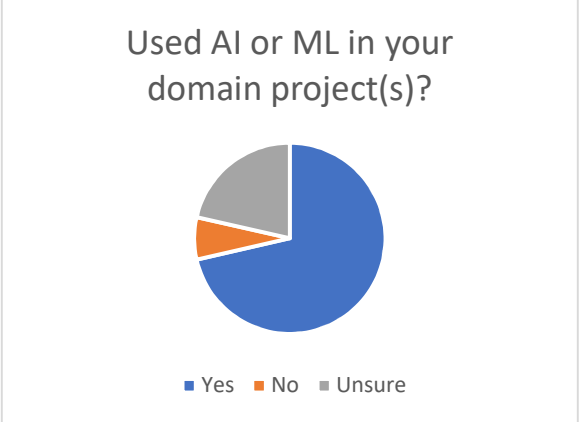


Among the people who we surveyed, many have various major, professions, interests, and hobbies (e.g., botany, finance, pharmacist, etc.).



Many people who has at some programming experience has their most favorite programming language as Java and Python, with a close runner-up from R.





Many people used AI/ML in their domain project(s), or at least those who know it, as the people who were unsure about it was considerable. This also shows the current state of knowledge of AI and ML among the general public.

The next part of our survey is the application experience, where we analyze how was the user's experience specifically towards GraphicalAI.

Table 1. Questions asked in the survey

Section	Question
Introduction	What is your age? [Number]
	What specific major, profession, interests, or hobbies you have? [Text]
	How much do you know about programming? [1-5]
	What programming language are you most comfortable, if any? [Text]
	How much do you know about Artificial Intelligence and Machine Learning? [1-5]
	Have you used AI/ML in your project? [Yes/No/Don't Know]
Application Experience	What is your overall experience in using GraphicalAI?
	Was it easy to use?
	Was setting-up GraphicalAI difficult?
	Which part of the GraphicalAI do you like most? [Creating the model, deploying and utilizing the model, neither, either]
Generalization	Compared to programming, do you think GraphicalAI served its purpose?
	Do you see further potential in GraphicalAI?
	As a domain expert with limited knowledge on programming, would you say was it easy to use GraphicalAI? [Yes/No/Maybe]
	If you have used similar GUI-based AI/ML application, do you think GraphicalAI was better? [Yes/No/Maybe]

6. Related Works

An article from the *Fast Company* shows us the worlds "first" Graphical AI interface application [16], even borrowing the similar name from the application. Even though this application, called Cortex, has a very similar goal of allowing developers to develop AI models without the complexity of manually programming the models, a major difference is their application is focused on integrating it with company workflow. While the article had not mentioned the flexibility of choices their application provides, it seems clearly that they are not focused on flexibility to integrate the AI models with wide case of scenarios, which we do, so the domain experts can easily modify and change parts to fit their specific, unique needs.

The company Trendskout has provided multiple applications, as showcased on their website [17], for users to build specific AI models for specific purposes also using nodes

and connections. Though the goal of this company is to integrate AI and ML with business aspect of other companies but does little to emphasize the goal for domain-experts and their projects especially for an individual and a team. Additionally, the applications provided by them are split into different sets of specific use cases, whereas GraphicalAI or its goal is to provide such applications but with wider use-cases for unique scenarios and needs from other developers. In the end, this company only focuses on the business aspect and provides limited flexibility in terms of what the user may specifically want add and change their models from the ground-up.

The graphical application, PerceptiLabs [18], comes even closer in terms what GraphicalAI does. They have the model page, training page, evaluation page, etc. similar to our pages—modeling, training, and deploying—in corresponding the stages needed to fully build a model. Though to gain the flexibility (i.e., customization) of each of the nodes in their application, the user has to know how to code. And that immediately differentiates our goal, which is to only use the components of GUI to let the users customize their individual nodes of any model. Wrapping up, all three of these applications has their own unique advantages, but all of them does not have the goal of providing the flexibility to let the users use all kinds of nodes to build their model without hindering the usability (e.g., inclusion of programming).

7. Conclusion and Future Work

So from the user study, the results were rather positive that the users, such as domain experts, like the GraphicalAI and its idea. Even though there are many similar applications competing, we are also looking to further differentiate ourselves by focusing on GraphicalAI to be more educational and youth friendly, as the demand for it grows in many industries in the future [21]. Also in the future, we are thinking of further adding more nodes to provide more flexible options, as the current number of nodes is very limiting. We will also add visualization for the training page to better see the date.

8. References

[1] Schlee, M., & Vanderdonckt, J. (2004). Generative Programming of graphical user interfaces. Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '04. https://doi.org/10.1145/989863.989936

[2] Furman, J., & Seamans, R. (2019). AI and the Economy. *Innovation Policy and the Economy*, *19*, 161–191. https://doi.org/10.1086/699936

[3] Campbell, C., Sands, S., Ferraro, C., Tsao, H. Y. J., & Mavrommatis, A. (2020). From data to action: How marketers can leverage AI. *Business Horizons*, *63*(2), 227–243. https://doi.org/10.1016/j.bushor.2019.12.002

[4] *Cloud Computing Services*. (n.d.). Microsoft Azure. Retrieved April 9, 2022, from https://azure.microsoft.com/en-us/

[5] Google Cloud. (n.d.). *Cloud Computing Services* |. Retrieved April 9, 2022, from https://cloud.google.com/

[6] *Cloud Computing Services - Amazon Web Services (AWS)*. (n.d.). Amazon Web Services, Inc. Retrieved April 9, 2022, from https://aws.amazon.com/

[7] Shen, A. (n.d.). *GraphicalAI*. GitHub. Retrieved April 9, 2022, from https://github.com/AndrewHC36/GraphicalAI

[8] *Python*. (n.d.). Python.Org. Retrieved April 9, 2022, from https://www.python.org/

[9] Bangert, P. (2021). The Necessity for Collaboration Between Data Scientists and Domain Experts. *SPE Symposium: Artificial Intelligence - Towards a Resilient and Efficient Energy Industry*. https://doi.org/10.2118/208634-ms

[10] *tf.GradientTape | TensorFlow Core v2.8.0*. (2022, March 23). TensorFlow. Retrieved April 9, 2022, from https://www.tensorflow.org/api_docs/python/tf/GradientTape

[11] *flask: The Python micro framework for building web applications.* (n.d.). GitHub. Retrieved April 9, 2022, from https://github.com/pallets/flask

[12] *flutter: Flutter makes it easy and fast to build beautiful apps for mobile and beyond*. (n.d.). GitHub. Retrieved April 9, 2022, from https://github.com/flutter/flutter

[13] a. Fisher, R. (1936). Iris Data Set. *UCI Machine Learning Repository*. https://archive.ics.uci.edu/ml/datasets/iris

[14] Parmar, R. (2018, July 9). *Wine Quality*. Kaggle. Retrieved April 9, 2022, from https://www.kaggle.com/rajyellow46/wine-quality

[15] Tsai, W. T., Sun, X., & Balasooriya, J. (2010). Service-Oriented Cloud Computing Architecture. *2010 Seventh International Conference on Information Technology: New Generations*. https://doi.org/10.1109/itng.2010.214

[16] Schwab, K. (2018, July 10). *This Is The World's First Graphical AI Interface*. Fast Company. Retrieved April 9, 2022, from https://www.fastcompany.com/90157777/this-is-the-worlds-first-graphical-ai-interface

[17] Trendskout. (2022, March 15). *Trendskout, AI & Machine Learning Software*. Retrieved April 9, 2022, from https://trendskout.com/en/

[18] PerceptiLabs, Inc. (n.d.). *PerceptiLabs*. PerceptiLabs. Retrieved April 9, 2022, from https://www.perceptilabs.com/

[19] Visser, E. (2008). WebDSL: A Case Study in Domain-Specific Language Engineering. *Lecture Notes in Computer Science*, 291–373. https://doi.org/10.1007/978-3-540-88643-3_7

[20] Shen, A., & Sun, Y. (2021). GraphicalAI: A User-Centric Approach to Develop Artificial Intelligence and Machine Learning Applications using a Visual and Graphical Language. *2021 4th International Conference on Data Storage and Data Engineering*. https://doi.org/10.1145/3456146.3456155

[21] Suárez-Varela, J., Ferriol-Galmés, M., López, A., Almasan, P., Bernárdez, G., Pujol-Perich, D., Rusek, K., Bonniot, L., Neumann, C., Schnitzler, F., Taïani, F., Happ, M., Maier, C., Du, J. L., Herlich, M., Dorfinger, P., Hainke, N. V., Venz, S., Wegener, J., . . . Cabellos-Aparicio, A. (2021). The graph neural networking challenge. *ACM SIGCOMM Computer Communication Review*, *51*(3), 9–16. https://doi.org/10.1145/3477482.3477485

Andrew Shen

Short Biography

Yu Sun

Short Biography