See It Learn: Machine Learning Visualizer

Ishmam Tashdeed
Computer Science and Engineering
Islamic University of Technology
Dhaka, Bangladesh
ishmamtashdeed@iut-dhaka.edu

Tauseef Tajwar
Computer Science and Engineering
Islamic University of Technology
Dhaka, Bangladesh
tauseeftajwar@iut-dhaka.edu

Muftiqur Rahman
Computer Science and Engineering
Islamic University of Technology
Dhaka, Bangladesh
muftiqurrahman@iut-dhaka.edu

Farhan Ishmam
Computer Science and Engineering
Islamic University of Technology
Dhaka, Bangladesh
farhanishmam@iut-dhaka.edu

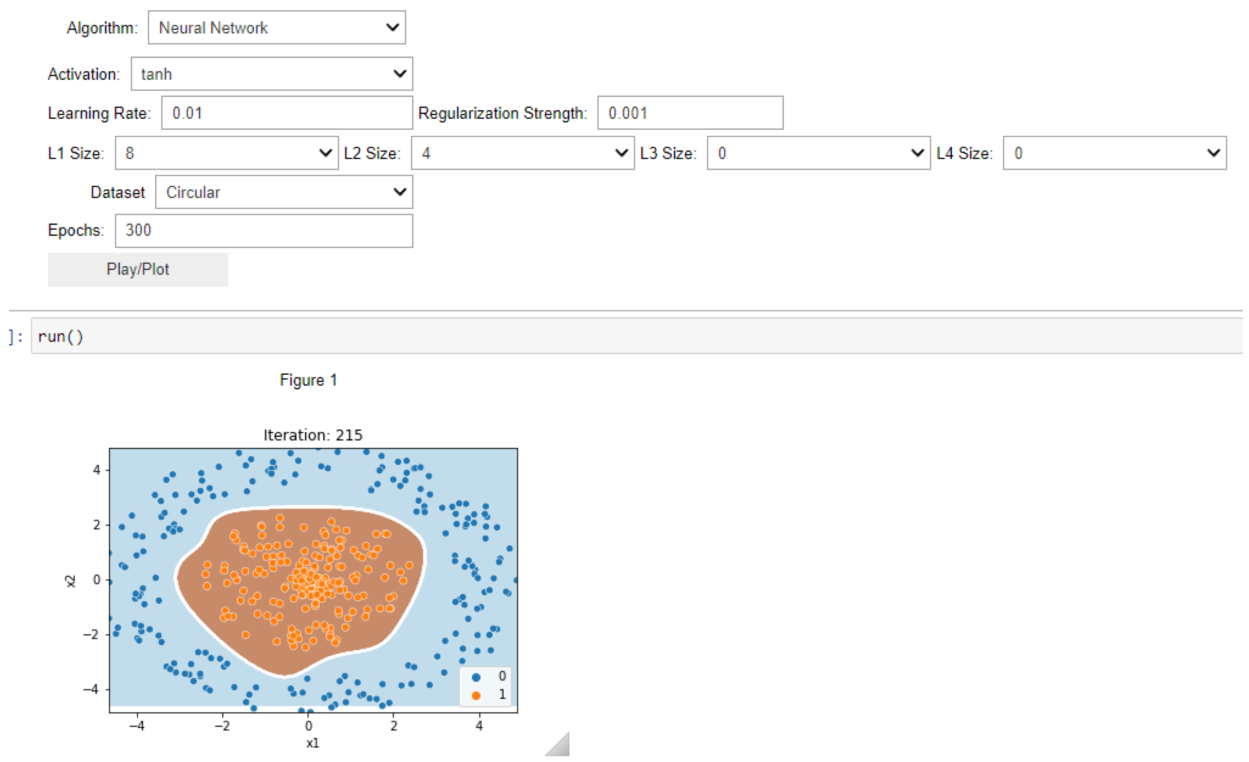
***Abstract*—Visualization is the key to comprehending complex algorithms and dynamic scenarios. Machine learning is an emerging and expanding field that can benefit from visualization as a tool for the learners. Supplemented by interactivity and accessibility, we plan to deliver a tool that would serve as a convenient resource for academic learners and machine learning enthusiasts. Our report shall provide a brief overview to the machine learning algorithm visualizer tool: See It Learn and explore all the details and nuances about its motivation, features, technologies, methodologies and implementations. We shall conclude our report with the analysis of our current standard of work followed by our view on it. Finally, a brief discussion about the future of this project will be discussed. We hope this report shall be a comprehensive guide to anyone trying to get better comprehension of this project or working on similar projects.**

I. Introduction

If we look at earliest means of human literature, then we see forms of visualization engraved on caves in Europe [1]. Archaeologists concluded that humans had necessitated opting for visual means to convey information above any other medium as a visual representation can easily be deciphered by any other human being. The advent of textual scripts also had a visual context i.e., the texts were more symbolic to their visual forms than an abstract meaning. A common example can be seen when deciphering Roman numerals, I, II, and III — as their visual forms corresponded to the number of 'I' s in the number. Visually intuitive texts lead to larger information retention by the brain and thereafter, leads to convenience in use of the texts by other humans. As information began to spread across communities and the influx of information surpassed the means to store, the humans realized that visual forms of information are somewhat more difficult to produce and preserve — eventually forcing them to opt for textual means of representation. Textual representations are more compact, can be universally comprehended, and with the advent of paper became an easily communicable medium. Subsequently, with the advent of the printing press, one of the most revolutionary inventions of humanity, the number of textual resources became astronomically high; books and newspapers became a part of our daily life; children were raised as if they were born to read. However, the great minds continued to opt to conceptualize using visual forms rather than inferring from large blocks of structured data. Even the texts they produced had more visual, graspable, and realistic analogues compared to explicit description of mathematical events and mathematical formulae.

Now, in the age of computers, producing and storing visualization has become less of a challenge and more of a choice. The effort required to create a picture book is similar to that of a storybook — visual forms do not necessarily come at the expense of additional time and effort. Photography has ensured realistic visualizations; animations have added dynamicity; touch-screens, keyboards, monitors have added interactivity. Augmented and virtual reality have added extra depth to the immersion and interactivity. Technologies revolutionized every field from medical diagnostics to media consumption — however, the means of delivering education has not shifted much and is still enrooted in the rudimentary techniques of delivering static lectures and processing blocks of texts written in books.

Education probably has the highest potential for improvement due to the revolutionization of visualization across all media. The process of learning fundamentally revolves around remembering factual data, identifying patterns among data, and capturing concepts preferably in visual forms [2]. Conceptualization is highly correlated to identifying patterns and thus, making the two tasks part of a single major task of identifying and connecting patterns. Let's look at an example of a student trying to understand the concept of gravity. He would firstly try to relate the concept of gravity to a previous knowledge source or past experience in his life; for instance, when he jumps, he falls down – can be associated with gravity. Secondly, he has to remember some facts e.g., the force is called "gravity" and it works downwards towards the ground. Given the facts and experience, the student now faces the most difficult task of connecting all the pieces to the textual formula of gravity. The formula roughly translates to - the higher the mass, the greater the force of attraction — a phenomenon that is common in real life and can be easily explained by "seeing".


Algorithm: Neural Network

Activation: tanh

Learning Rate: 0.01 Regularization Strength: 0.001

L1 Size: 8 L2 Size: 4 L3 Size: 0 L4 Size: 0

Dataset Circular

Epochs: 300

```
]: run()
```

Fig. 1. User Interface of See It Learn. The users can interact by selecting the learning algorithm, the dataset and hyperparameters

The student would then attempt to connect this formula to a real-life phenomenon, let's say, he imagines a big truck falling off the cliff which would subsequently fall faster than a paper sheet due to its huge mass. However, this seemingly simple yet naive mental visualization is incorrect as gravity affects both the paper and the truck equally, and the only real reason why the paper will realistically reach the ground after the truck is due to air resistance. This leads to another issue of ambiguity of textual representations which happens to be a common problem faced by beginners of a particular field.

The whole process of text-to-visual conversion in the head can be avoided if the learning method focuses on visual delivery of the concepts i.e., the student will be shown an animation of an object falling down where the mass of the object changes to denote the change of force. By identifying the changing pattern, the student can linearly relate mass with gravity. The aforementioned ambiguity can also be removed if the fall of a lighter and heavier object is compared and afterwards, concluded that they reach the ground at the same time, but the force they experience is different. By doing so, the student will have a better grasp of the concept of force itself along with the concept of gravity. The key point of using visualization in this scenario is the efficient identification of pattern change and removal of text-based ambiguity.

Another aspect of learning that we overlooked earlier is the amount of effort given by the student. The student has to provide attention to specific parts when he is reading large bodies of texts. Reading is a mentally intensive task and the brain optimizes this task with the help of attention. Attention makes the brain focus on certain parts of the text more which would consequently improve the processing of the texts by a significant margin but would also reduce the amount of data retrieved from the text source. Human attention is far from perfect; especially for the non-experts like amateur learners and we consider students as non-experts. A student would be in an endless gamble of where to focus and where not while reading large corpora of texts. The problem can be solved in many ways - either by training the student's reading skills, re-reading the texts over and over again, or by learning from "easier" sources. Reading skills are developed over time and an amateur reader cannot be transformed to an expert reader overnight. Re-reading texts seem to be the better alternative, and unsurprisingly, that's what the education system has enforced on the students for years. Rereading texts not only makes learning a duller experience but also reduces the efficiency of students to develop concepts by a significant margin [3]. The most intuitive and easiest source to human learning is by visual means. The brain, in fact, learns every concept visually but has to go through extra steps to transform the structured data first to produce the visualization. All these extra steps can be avoided if we can, simply, learn visually. The above text has no intention to reproach the act of reading; on the contrary, we believe reading to be the best source of supplementing one's learnt skills as texts serve as one of the easiest means to communicate. However, when other media are available in a field of repeated teaching like the education system, we ought to find better ways of delivering the content to the students. Hence, we recommend the primary learning method i.e., lectures delivered in the classrooms, to be built around visualizations and the supplementary knowledge sources can be textual resources.

Machine Learning is a rapidly growing field and in modern days, the number of people learning machine learning is well above millions. The concepts of machine learning are enrooted in statistics - a field notorious for showing large amounts of structured data. Both machine learning and statistics traditionally rely on static images as a form of visualization with little to no interactivity. For instance - statistics books have a single picture of a certain distribution with a certain parameter and would then explain in text how the distribution would change if the parameter changes. All the staticity and text-based explanations can be avoided by providing a simple system that can just animate the scenario and change based on the user's given parameters. We would believe that the added animation would make the visualization easier for the learners and the interactivity would make it easier for the learner to provide more attention for the learners [4].

Our provided tool is just a piece to our goal of renovating the field of education with the best means of visualization. The provided tool will come as a Jupyter Notebook that we uploaded in a Github repository [5] along with necessary instructions to run this tool. The tool would provide visualizations to 9 different learning algorithms commonly used in machine learning on 9 different datasets. There will be opportunities to change the parameters and the dataset as a form of interactivity, along with stopping the animation at any given frame. It should be noted that we plan to deliver the best form of visualization and some of the non-iterative algorithms cannot be animated; thus, being shown as static images. The static visuals still provide interactivity as seen with the animated visuals. The whole code is modular and can be extended to support more algorithms, datasets, and learning parameters.

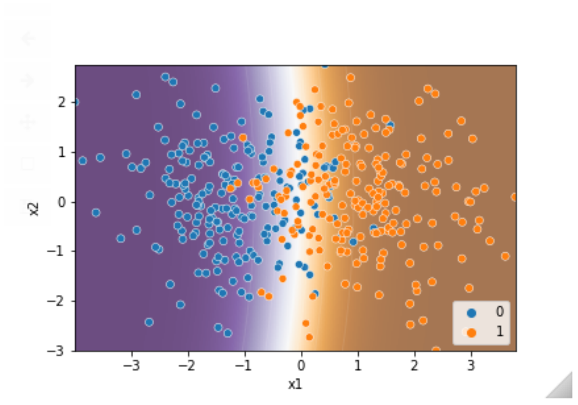


Fig. 3. Plot of Naive Bayes visualized by See It Learn. As Naive Bayes is non-iterative, only a single plot is produced. The intensity of the hues denote the probabilistic value associated to that particular data class.

II. Methodology

We envisioned a tool to animate machine learning algorithms and the first thing that came to our mind is to use Python as our preferred programming language as the reason should be quite obvious given Python's dominance in the field of machine learning [6], and its huge repertoire of third-party libraries to implement and visualize machine learning models. Secondly, for our development platform, we opted to use Jupyter Notebook — a web-based and interactive computational environment for running code snippets along with plots, markdown, and rich media [7]. We believe that Jupyter is the perfect platform to run our Python code snippets and animating them. For the implementation of the learning algorithms, we have a huge repertoire of predefined functions available in third-party libraries like scikit-learn, or can easily construct our own models using keras from the tensorflow library. For the user interface, we opted to use ipywidgets for its simplicity to use in the notebooks and ease of expandability

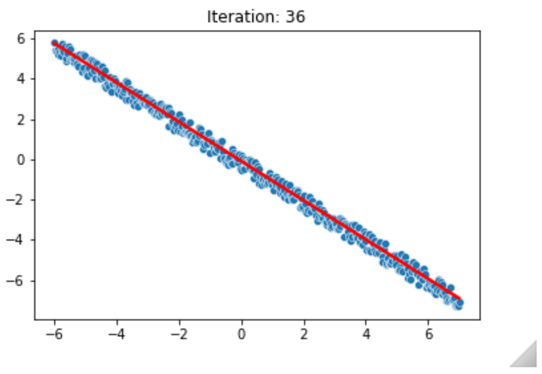


Fig. 2. Snapshot of animation of Linear Regression visualized by See It Learn. The blue points denote the data points and the red line denotes the best fit line. The number of iterations is also shown at the top of the plot.

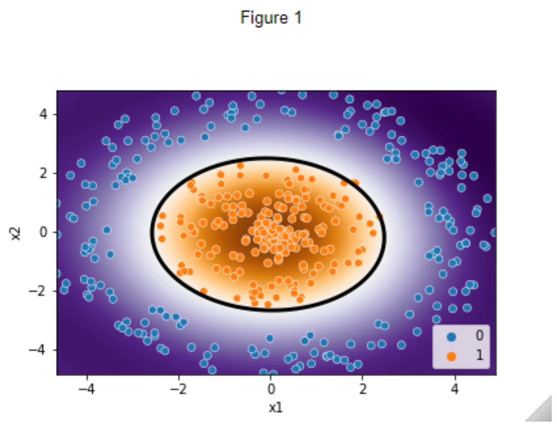


Fig. 4. Plot of non-linear SVM where the intensity of the hues denote the probabilistic value associated to that particular data class.

to web-based applications using deployment tools like voila.

III. Core Features

The core of our project revolves around providing the user the best form of visualization which is either animation for iterative algorithms or static images for non-iterative algorithms. The following algorithms are visualized in our program:

TABLE I
Visualized Machine Learning Algorithms

Algorithm Name	**Type**	**Visualization**	**Unique Hyper-parameter**
Linear Regression	Supervised, Iterative	Animation	Regularization Type, Strength
Logistic Regression	Supervised, Iterative	Animation	Regularization Type, Strength
Neural Network	Supervised, Iterative	Animation	Layer Size, Activation
Linear SVM	Supervised, Iterative	Animation	Regularization Type, Strength
Non-Linear SVM	Supervised, Iterative	Animation	Kernel, Polynomial Degree
K-Means	Unsupervised, Iterative	Animation	No. of Clusters
Naive Bayes	Supervised, Non-iterative	Static Image	X
Decision Tree	Supervised, Non-iterative	Static Image, Tree Graph	Impurity Criterion
PCA	Unsupervised, Non-iterative	Static Image	X

IV. Experimental Setup

All the dependencies for the experiment are listed in the requirements.txt file along with the specified versions. Some notable ones will be discussed in this section.

- `notebook`: Requirement for *Jupyter Notebook* which is a web-based application for executing cell-based code. It features kernels which are processes that run interactive code and return output to the user. [7]
- `numpy`: This is the fundamental Python package for scientific computation providing multidimensional array objects and tasks such as broadcasting for efficient computation. [8]
- `scikit-learn`: Scikit-learn is a library for Python which features various classification, regression and clustering algorithms alongside SVMs, random forests, gradient boosting, k-means clustering etc. [9]
- `matplotlib & seaborn`: These two are comprehensive libraries for creating static or animated and interactive visualizations in Python. [10], [11] These were used for the visualizations of different machine learning algorithms.
- `ipywidgets`: This is a package for making interactive HTML widgets for the *Jupyter Notebook* file. This gives the user an immersive experience as they can control different parameters of the visualizations. [12]

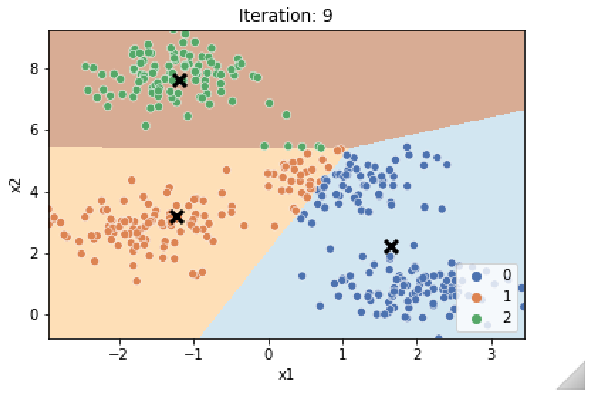


Fig. 5. Snapshot of the animation of K-means clustering where the crosses denote the cluster centers.

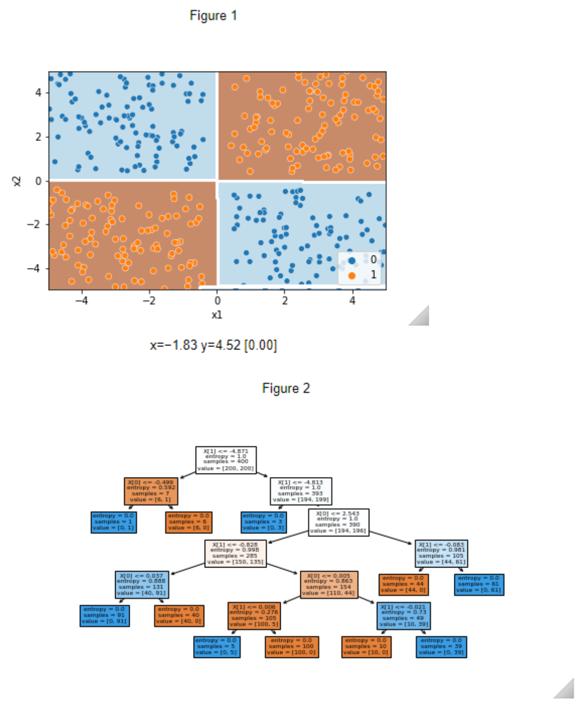


Fig. 6. The plots of a Classification Decision Tree. Since, it is a non-iterative algotithm, only a static image of the decision boundary and the annotated decision tree's image is produced.

V. Result analysis

As this was an implementation-based project, we do not have a basis to compare our results with pre-existing methods. So, in this section, we will discuss the problems that we have faced during implementation of our method. One of the most difficult hurdles for us was to synchronously update the Jupyter front-end and back-end. This was evident when changing a value in the user-interface using the interactive widgets would not update that value in the Jupyter kernel. The update could be observed in the execution of any other cell following the change. So the solution to this problem was to programmatically execute a cell using a *Javascript* function. Another one of the problems were the animations

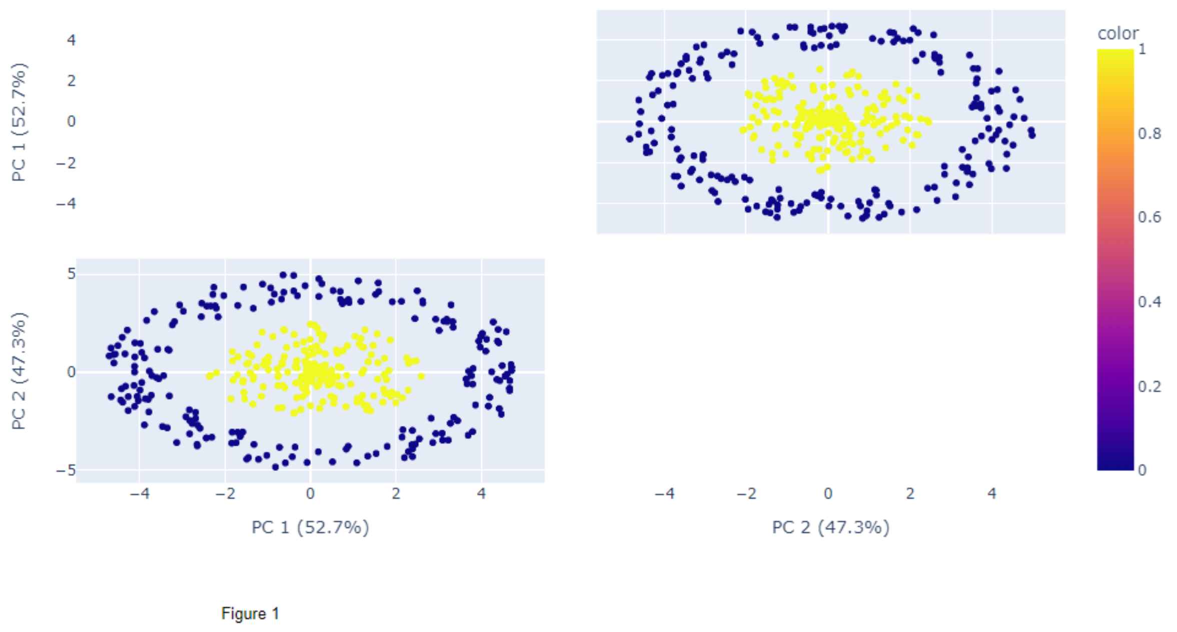


Fig. 7. The plot of Principal Component Analysis. Since, it is a non-iterative algotithm, only a static image is produced.

for the visualizations. To overcome this, `FuncAnimation` function was used along with a back-end method that connects the widget (play/plot button) with acquiring the current plot and updating it in intervals.

VI. Conclusion and Future Work

The tool performed satisfactorily well in visualizing the aforementioned algorithms and showed great results in accessibility, ease of use, and interactivity. However, we do believe that the tool can be further improved by working in some particular fields — notably, improving the accessibility of the tool by deploying it as a web-based application and adding more commonly used algorithms to its repertoire. We excluded some popular learning algorithms like, K - Medoids Clustering, K - Nearest Neighbors, Regression Trees etc — all which can be easily included in the future due to the modular design of our system. Furthermore, some of the state-of-the-art models use learning algorithms that aren't quite popular right now but might revolutionize the field of machine learning in the near future. We would, hence, like to add an extra operational mode for developers to easily add new learning algorithms interactively by using an interactive development medium instead of programming it. Such a development platform can be either integrated to our current system or be distributed as a standalone system depending on its use cases and complexity. Finally, we plan to improve the flexibility in visualizing datasets by enabling users to add their own datasets and animating the learning algorithm on those datasets. Our tool met all the requirements for a complete visualization tool added with interactivity. Any learner who is willing to delve in the field of machine learning or any mentor willing to deliver comprehensive lectures on machine learning can find this resource quite useful as a primary or supplementary material. Our tool was designed in such a way that it can be incrementally developed, used as a backend system to a larger or more robust user-interface, or be modularly added to a different visualization system. We hope a good number of people will find this resource helpful and will continue to contribute to the source code that we shared in our GitHub repository [5].

References

[1] G. M. Morriss-Kay, "The evolution of human artistic creativity," Feb 2010.

[2] E. KELE and S. EPN, "Brain and learning," *Journal of Turkish science education*, vol. 3, no. 2, p. 31, 2006.

[3] M. Weimer, "Is rereading the material a good study strategy?," 2014.

[4] N. Geri, A. Winer, and B. Zaks, "Challenging the six-minute myth of online video lectures: Can interactivity expand the attention span of learners?," May 2017.

[5] T. Tajwar, I. Tashdeed, F. Ishmam, and M. Rahman, "Ml algorithm visualizer github repository," 2022.

[6] G. L. Team, "Why python so popular for ai and machine learning?," 2021.

[7] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (F. Loizides and B. Schmidt, eds.), pp. 87 – 90, IOS Press, 2016.

[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, p. 357–362, 2020.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[10] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[11] M. Waskom, O. Botvinnik, D. O’Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, and A. Qalieh, “mwaskom/seaborn: v0.8.1 (september 2017),” Sept. 2017.

[12] J. widgets community, “ipywidgets, a github repository.” Retrieved from https://github.com/jupyter-widgets/ipywidgets, 2015.