THE OHIO STATE UNIVERSITY

Goodreads Book Recommendations from Fully Connected Neural Networks

Project Category: Neural Networks and Text Analysis
Physics 5680, Autumn 2024

Author: Colin Voorhis

November 19, 2024

Abstract

This report investigates the application of Fully Connected Neural Networks (FCNs) and Bidirectional Encoder Representations from Transformers (BERT) in creating user directed book recommendation algorithms. Goodreads text reviews, book descriptions, and numerical ratings are used to train an FCN which predicts the likelihood that a reader would enjoy a given book based on their opinions of another book. Text based inputs are used to capture nuanced user preferences that are typically lost in other book recommendation techniques, and to generate recommendations that do not rely on book popularity as a metric for recommendation. Once trained, the recommendation algorithm can intake new reviews from users to predict how much they may enjoy any book with a text description available that summarizes its contents. The network is found to have a >70% effectiveness in predicting user book preferences across a collection of roughly 7000 book reviews of 4000 distinct books, and the techniques discussed can be generalized to media beyond literature.

1 Introduction

Goodreads is a web database that allows users to categorize, rate, and review books. It contains a large collection of data about English literature including numerical ratings and written reviews from thousands of users, and text metadata describing the contents of over two million books. The size and depth of this dataset presents many opportunities for machine learning applications, especially in the field of recommendation algorithms.

Recommendation algorithms are a widely studied and hugely influential topic in computer science. Highly optimized recommendation algorithms are the foundation of Google's search functionality, Amazon's digital storefront, and most content platform's user engagement metrics. Goodreads itself even has a recommendation section that "learns about your personal tastes from your ratings, then generates recommendations unique to you."[1]. While these algorithms are generally effective, they are typically optimized towards the interests of the corporations creating them rather than the users interacting with them, require an immense amount of computing power and data to produce satisfactory results, and allow for minimal user input beyond what interaction data they provide to the corporation. With this report I aim to showcase a book recommendation algorithm which allows users to voluntarily direct their own recommendations with text

prompts, rather than just their interaction data, while minimizing the amount of computing power and external data required to implement the algorithm.

The proposed algorithm uses an FCN to predict if a given reviewer will enjoy a given book based on their review and rating of another book, and the descriptions of both books. The FCN takes in three BERT vectors as input; the first is a vectorized version of a text review for a book, the second is a vectorized description of that book, and the third is a vectorized description for a second book. The FCN has a single node output which attempts to predict if the reviewer would highly rate the second book or not, based on their opinions of the first book. Once trained, the algorithm can be used to predict book approval or disapproval given new reviews and book descriptions.

2 Related Work

As previously discussed, recommendation algorithms are a well studied topic in the field, so many large and small scale projects have attempted to apply these algorithms to the medium of literature. Goodreads itself attempts to "better predict which books people will want to read next" via your reading list and book ratings, and provides more general recommended lists of currently popular books as voted by the entire website[2]. Goodreads' approach follows the methodology of user based filtering[3], mostly relying on identifying users with similar groups of read books, and recommending books to an individual that other similar users have read and enjoyed. This approach is frequency criticized for being as overly simplistic and impersonal[4], as its structure highly favors books that are popular in general, rather than specifically resonant with a single user's interests. This approach also requires a large amount of an individual's data and a massive collection of other users to work in the first place, and is therefore impractical to use in a scaled-down approach.

Many smaller scale projects have attempted to use the Goodreads dataset to classify and recommend books as well. The most common method across these projects is to define a dataset of books linked by similarity, analyze what books a user has read or selected, and recommend popular books that are similar to their favorites[5]. This approach is known as item-based collaborative filtering[3], and these projects often use the metric of cosine similarity to determine book similarity, vectorizing relevant text surrounding a book and ranking two books as similar if their vectorized information has a high degree of cosine similarity[6]. This approach is more scalable and often reaches a passable level of recommendation quality, but cosine similarity is a broad and imprecise metric when it comes generating personalized recommendations. As with user similarity, it also struggles with recommending less popular books, as books are weighted by popularity in their selection.

The most effective recommended systems available tend to follow a hybrid recommendation strategy, drawing on both of the two previous methods by scoring and weighting user/item based metrics before tallying a list of books that score well in both categories[3]. This approach has promise, but is still limited by the nuance of the data it relies on. An algorithm can only be so personal if it only uses the title and tags associated with a user's preferred books, as many of these hybrid models do[5]; a user's genuine preferences are not well captured in a list of book titles and genres they approve of.

The approach proposed in this project is specifically structured to attempt to resolve the two most common issues with all the above strategies: their lack of genuine personalization and their reliance on book popularity to generate suggestions. User text reviews are highly nuanced and personal in comparison to user genre preferences or book popularity rankings, and incorporating these text reviews as input allows the algorithm the possibility of capturing highly unique user preferences. Relying on text as input also sidesteps the issue of book popularity as a metric, as every work in the Goodreads database has a text description associated with it. As long as a user is able to write a brief review about a book they enjoy in the Goodreads database, that review can be used to make informed predictions about any other book in the Goodreads database, as they all have descriptions which can be vectorized and fed into the prediction algorithm input.

3 Dataset

The data to train the FCN is drawn from the Goodreads Book Graph Datasets, which collected all publicly available user data from Goodreads in 2017[7][8]. The full dataset contains information relating to 2,360,655 books and 876,145 users. In these datasets the term "work" refers to a specific literary product and all its editions and releases, whereas "book" refers to a specific release, edition, or version of a given work.

Three of the Goodreads datasets are used in this project. The first is goodreads_books, which provides a unique numerical book ID and text description for every book in the Goodreads system. The second is goodreads_book_works, which contains a list of every work in the Goodreads system, as well as the best book ID to reference for a particular work in cases where a work has multiple associated books. The final dataset is goodreads_reviews_spoiler_raw which includes all text reviews in the Goodreads system alongside a book ID for the book being reviewed, a user ID that uniquely identifies the reviewer, and the numerical rating from 1 to 5 that the user gave the book.

The dataset used to train and test the FCN is generated using the above three datasets. First, a subset of book IDs is collected from goodreads_book_works such that only works with more than 50,000 reviews are included. This metric serves both to limit the size of the dataset and to ensure that selected reviewers are likely to have reviewed both the book in question and other books in the dataset. These book IDs can then be used in goodreads_reviews_spoiler_raw to collect every text review and in goodreads_books to collect a text description for each of these books. The text reviews are then searched for pairs of books where a single user has rated and reviewed both books, and where one of the text reviews is at least 200 characters long. When a valid book pairing is found, the text description of each book is saved alongside the review text for one of the books, and the numerical rating of the other. This data collection method was able to find 6,942 valid book/user pairings across a set of 4,564 unique works within the allotted computing time. See Figure 1 for a detailed breakdown of these data processing methods.

Once this dataset is collected, it must be sanitized to be compatible with the architecture of the FCN. The description and review texts are vectorized using BERT. BERT is a pre-trained text encoder which is designed to transform long strands of text characters into 768 dimensional "embeddings" for each set of text where each dimension is a single floating point decimal value between zero and one[9]. The ratings for each book are also translated into a single binary value. 4-5 stars is interpreted here as a positive review, denoted by a 1, and 0-3 stars is a negative review, denoted by a 0. A given row from the final FCN dataset therefore contains 2304 decimal values from three BERT vectors as input and a single binary value which classifies book enjoyment as an output label. 80% of this dataset is randomly selected for training the FCN, and 20% is used to test it. 20% of the training dataset is further reserved for model validation.

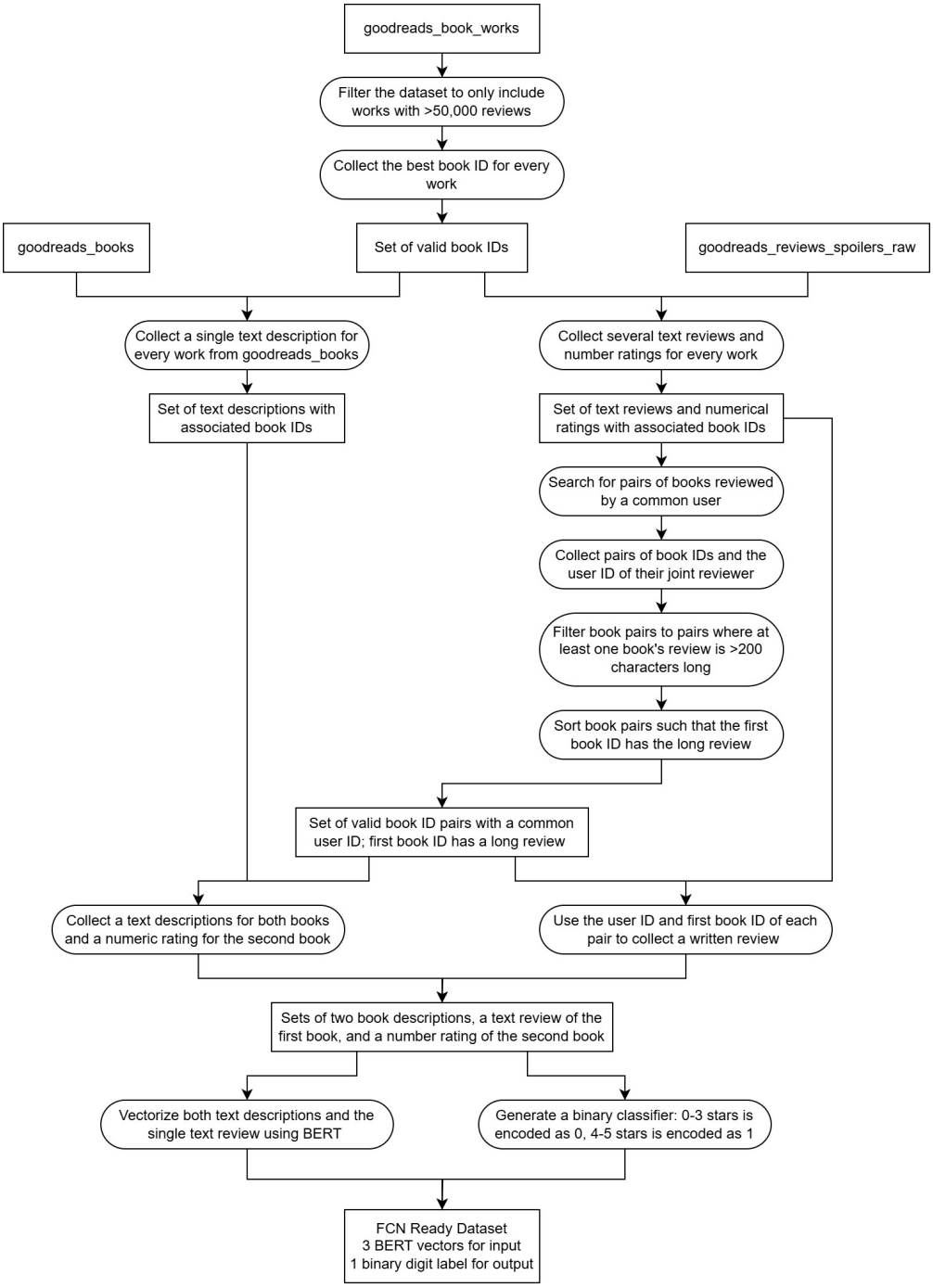

Figure 1: The process by which the Goodreads datasets are transformed into a final FCN compatible dataset. Rectangles represent datasets. The top three are derived from the Mengting Wan Goodreads datasets, and are labeled with their respective names. The rest are generated as part of the processing methods. Rounded rectangles represent data processing steps. An arrow drawn from a database to a processing step indicates that database is used in that processing step. An arrow drawn from a processing step to a database indicates that the processing step ends by creating the dataset it points to.

4 Methods

A Fully Connected Neural Network (FCN) is trained here to predict user ratings. A neural network is a collection of interconnected nodes, represented by numerical values, that send signals to each other according to certain mathematical functions. In an FCN, nodes are arranged into groups called layers, and each node in a given layer is connected to all the nodes in the next consecutive layer, hence the name "fully connected". The value contained in a given node is calculated by taking a weighted average of all the node values in the previous layer, then running the sum through a specific "activation" function. The first layer is called an input layer, and is supplied data from an external source. The last layer is called an output layer, and is designed to signal or categorize specific features of the supplied data. The intermediate layers are called hidden layers, and the parameters of these nodes are iteratively tweaked using machine learning to try and generate accurate output layer values given certain input layer values. All the layers used in this FCN except the output layer also contain one bias node, an extra node set that is always set to a constant value which allows the activation function an extra degree of flexibility[10].

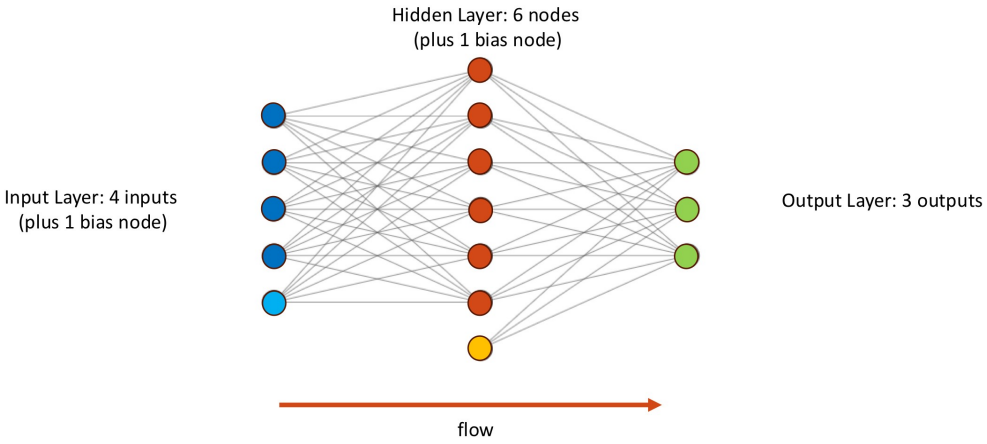

Figure 2: A simple FCN with four input nodes, three output nodes, and a single six node hidden layer, with bias nodes in every appropriate layers.[11]

An FCN is appropriate for this specific classification task primarily because it is able to extract nuanced information from very high dimension numerical inputs, then use that information to output very low dimension outputs. The input data of this network is a set of high dimensionality vectors which should contain information about the content of books and the opinions of book reviewers, and this information must be condensed into a single binary metric that represents approval or disapproval of a given book.

The hidden layers of this network use Rectified Linear Unit (ReLu) as an activation function, which is defined as follows[12]:

$$f(x) = (max)(0, x)$$

In practice, this function simply returns 0 for non-positive values, and returns the input value if it is positive. This restricts the hidden layer node values to be only positive.

The output layer uses the sigmoid function for activation. The sigmoid function is defined as follows[13]:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

In practice, this function takes any real number and restricts its output to the range (0,1). Very large positive and negative inputs generate outputs close to 1 and 0 respectively, and an input of 0 generates an output of 0.5. For this classification problem, the sigmoid output number can be interpreted as a probability that represents the network's confidence that the input reviews and descriptions represent a positive review.

The FCN used here takes in three BERT vectors as input, meaning the input layer of this FCN contains 1536 nodes that each receive an decimal between zero and one inclusive. Similarly, the FCN outputs a single decimal value to a single output node. The number of hidden layers, and the number of nodes per hidden layer, are both hyper parameters, metrics related to the structure of the network that can be optimized. In this case, a brute-force approach can be used to select the best set of hyper parameters. A range of 2-8 hidden layers and 200-500 nodes per hidden layer were tested; a network is generated with every possible combination of nodes divisible by 50 (200, 250, etc.) and hidden layer count and sequentially tested to determine which hyper parameters are most effective for this particular problem.

5 Results/Discussion

The FCN's effectiveness is evaluated using the Area Under the Curve (AUC) metric. AUC is defined as the area under the curve of a False Positive Rate (FPR) vs True Positive Rate (TPR) graph, which are defined as follows:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

Where:

- True Positive (TP) is the number of correctly classified positive instances.
- True Negative (TN) is the number of correctly classified negative instances.
- False Positive (FP) is the number of negative instances incorrectly classified as positive.
- False Negative (FN) is the number of positive instances incorrectly classified as negative.

In order to prevent overfitting, the FCN training process also incorporates binary cross entropy as a loss metric for early stopping. Binary cross entropy is a metric that quantifies the difference between the true distribution of labels and the network's distribution of predicted labels. The formula for binary cross-entropy is as follows[14]:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right]$$

Where:

- L is the scaler loss value, in the range [0,1].
- N is the number of samples in the dataset.
- y_i is the true label of the i-th sample (either 0 or 1).
- p_i is the model's predicted probability that the i-th sample that the label is 1, in the range [0,1].

The network uses the validation sample to calculate a loss scaler for every epoch. If loss does not improve within five epochs of training, the model stops training and saves the network weights from the previous most effective epoch.

As previously discussed, many candidate networks were generated with a varying number of hidden layers and nodes per hidden layer. These networks were all evaluated using AUC as the success metric, i.e. the network with the highest AUC is the most successful classifier. Each network is trained for 30 epochs using a batch size of 128, and with the above early stopping functionality, most networks stopped training between 12 and 18 epochs. The best performing network had five hidden layers with 400 nodes per hidden layer, which generated a test AUC of 0.7472, test loss of 0.5895, and a test accuracy of 0.7210. A plot of validation AUC versus epochs of this model is shown below in Figure 3, and the test AUC of all evaluated models is shown below in Table 1:

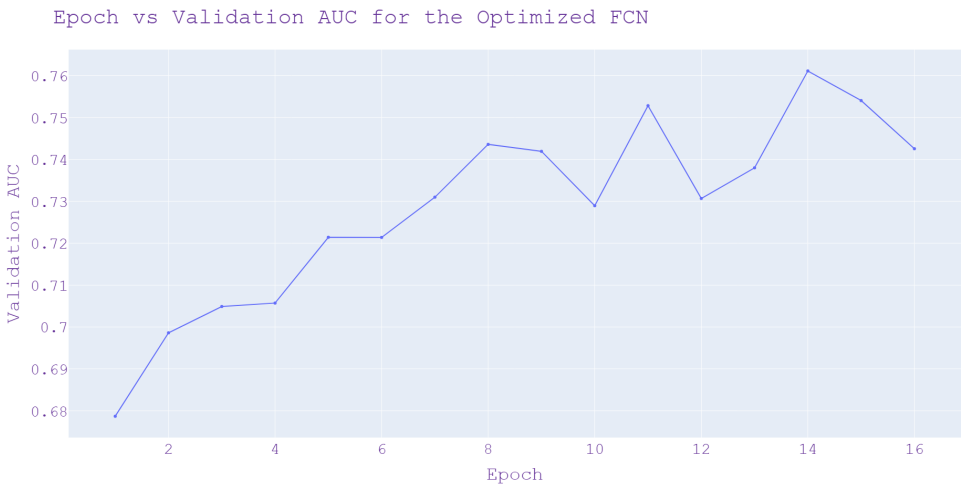

Figure 3: This Figure displays the validation AUC for every epoch of the most optimized FCN model, which contains five hidden layers and 400 nodes per hidden layer. The validation AUC peaks at 0.7611 during epoch 14.

Nodes per Layer	2 Layers	3 Layers	4 Layers	5 Layers	6 Layers	7 Layers	8 Layers
200	0.7330	0.7357	0.7337	0.7116	0.7176	0.7370	0.7242
250	0.7282	0.7126	0.7318	0.7314	0.7221	0.7296	0.7230
300	0.7318	0.7255	0.6926	0.7042	0.7388	0.7171	0.7302
350	0.7406	0.7300	0.7342	0.7398	0.7297	0.7124	0.7380
400	0.7266	0.7333	0.7247	0.7472	0.7277	0.7193	0.7362
450	0.7231	0.7344	0.7312	0.7279	0.7184	0.7236	0.7292
500	0.7303	0.7342	0.7362	0.7170	0.7262	0.7273	0.7384

Table 1: The test AUC for all evaluated combinations of numbers of hidden layers and nodes per hidden layer.

6 Conclusions/Future Work

This project presents a working model for FCN based, user directed content recommendation algorithms. The network is designed to personalize recommendations based on user inputs and to not rely on book popularity when assessing recommendations, instead basing recommendations on text based data including book descriptions and user reviews. The proposed algorithm is demonstrated to achieve a >70% success rate at predicting user preferences using a dataset of less than 7000 user reviews and book description pairs.

This technique could be further refined in several ways. A larger training dataset could be used to refine the performance of the algorithm, and k-fold validation could be used to fully utilize all available data for training. The network could also be remodeled to take in a fourth vector representing a random review of the book whose rating is being predicted; the algorithm could then use this second review as further context for the contents of the book. Another natural step for this algorithm would be to use a fully trained model to predict book ratings for users not contained in the dataset. In this case, a user would be required to supply a book ID and text review of one book, and the book ID of a second book they would like to compare their review to. A program could easily be constructed that allows users to write a short description of why they enjoyed one book, and ask the program if they would enjoy other books of their selection. Finally, this network could be generalized to other forms of media where large datasets of reviews, ratings and descriptions are available; TV shows, movies, and music are three mediums in particular that could benefit from this technique.

7 Contributions

I, Colin Voorhis, completed the entirely of this project on my own. Chat-GPT was used to generate a select few Python functions which were then modified for my own use; these code snippets are explicitly marked as AI based when included in the final algorithms.

References

[1] Goodreads, "Recommendations," Goodreads, 2022. [Online]. Available: https://www.goodreads.com/recommendations.

[2] Goodreads, "Announcing Goodreads Personalized Recommendations," Goodreads Blog, 2022. [Online]. Available: https://www.goodreads.com/blog/show/303-announcing-goodreads-personalized-recommendations.

[3] Moneka, M., "Book Recommendation System," Medium, 2022. [Online]. Available: https://medium.com/@mmoneka11/book-recommendation-system-75963c7d4144.

[4] Reddit, "Book Recommendations Thread," r/books, 2022. [Online]. Available: https://www.reddit.com/r/books/comments/z7vrh4.

[5] Ashima, "Building a Book Recommendation System," Medium, 2022. [Online]. Available: https://ashima96.medium.com/building-a-book-recommendation-system-a98c58a4f1bb.

[6] Trishala, S., "My Kind of Books," Medium, 2021. [Online]. Available: https://ts594.medium.com/my-kind-of-books-142053b3c0d6.

[7] Wan, M., "RecSys18 Paper," Mengting Wan Website, 2018. [Online]. Available: https://mengtingwan.github.io/paper/recsys18_mwan.pdf.

[8] Wan, M., "ACL19 Paper," Mengting Wan Website, 2019. [Online]. Available: https://mengtingwan.github.io/paper/acl19_mwan.pdf.

[9] Hugging Face, "BERT Model Documentation," Hugging Face Transformers, 2022. [Online]. Available: https://huggingface.co/docs/transformers/en/model_doc/bert.

[10] Stack Overflow, "What is the Role of the Bias in Neural Networks?" [Online]. Available: https://stackoverflow.com/questions/2480650/what-is-the-role-of-the-bias-in-neural-networks.

[11] Hughes, Richard. *Module 5 Key Points*. Physics 5680, 2024. Available: https://osu.instructure.com/courses/173367/files/69113193?module_item_id=13482704.

[12] Deepchecks, "Rectified Linear Unit (ReLU)," Deepchecks Glossary. [Online]. Available: https://www.deepchecks.com/glossary/rectified-linear-unit-relu/.

[13] ScienceDirect, "Sigmoid Function," ScienceDirect Topics. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/sigmoid-function.

[14] Arize AI, "Binary Cross Entropy (Log Loss)," Arize Blog, 2022. [Online]. Available: https://arize.com/blog-course/binary-cross-entropy-log-loss/.

[15] Plotly, "Plotly Python API Reference," Plotly Documentation. [Online]. Available: https://plotly.com/python-api-reference/generated/plotly.io.html.

[16] Plotly, "Plotly Express," Plotly Documentation, 2022. [Online]. Available: https://plotly.com/python/plotly-express/.

[17] Pandas, "Documentation Index," Pandas, 2022. [Online]. Available: https://pandas.pydata.org/docs/index.html.

[18] PyTorch, "Documentation Index," PyTorch, 2022. [Online]. Available: https://pytorch.org/docs/stable/index.html.

[19] NumPy, "Documentation Index," NumPy, 2022. [Online]. Available: https://numpy.org/doc/stable/.

[20] TensorFlow, "API Documentation," TensorFlow, 2022. [Online]. Available: https://www.tensorflow.org/api_docs.

[21] Keras, "API Documentation," Keras, 2022. [Online]. Available: https://keras.io/api/.

[22] scikit-learn, "API Documentation," scikit-learn, 2022. [Online]. Available: https://scikit-learn.org/stable/api/index.html.