Final Project Reflection Paper
Erdun E, Xinyu Wang, Fengkai Liu
Nov. 19th, 2024
Dr. Alan Jamieson
CS 5800:Algorithm

Final Project Reflection Paper

Introduction(Hailee)

K-nearest neighbor algorithm (kNN) is a machine learning algorithm that trains a program to compare the distance of one data point with a set of data points and make predictions based on such comparison. The kNN algorithm works based on the assumption that data points within the same category, with similar qualities, or within the same class can be found near one another. In other words, choose a kNN algorithm under two conditions: first, there are existing datasets that categorize data points within a certain distance range; second, the problem that is waiting to be solved is to categorize or classify a new data point based on the existing dataset. This characteristic of kNN further illustrates that it is a supervised learning algorithm. In supervised learning algorithms, a labeled dataset is fed into an algorithm, and learned by the algorithm to predict the outcome of a new data point accurately. Supervised learning algorithms are best for pattern recognition. In this project, it's the ideal algorithm to learn from the iris dataset and predict the variety of the new data points.

This final project implements the kNN algorithm using the Iris dataset to train the program and predict user-input data points.

Background(Hailee)

In 1951, Evelyn Fix and J. L. Hodges, Jr. from UC Berkeley, published a paper analyzing nonparametric discrimination on consistency properties. In this paper, they developed an algorithm to solve the discrimination problem, which is to choose a positive integer k that is large but small compared to the sample size; specify a matrix, such as Euclidean distance; find k values from the pooled samples which are nearest to the targeted distributed space z.

In the paper “Nearest neighbor pattern classification” published in 1967, Thomas Cover and Peter E. Hart came to the conclusion that “half of the available information in an infinite collection of classified samples is contained in the nearest neighbor”.

An improvement of the kNN algorithm was introduced by James M. Keller, Michael R. Gray, and James A. Givens, Jr. in 1985. They refined the algorithm and reduced the error rate by using a Fuzzy K-NN Classifier. The Fuzzy K-NN Algorithm assigns membership values to the sample sets which increase the weight of the correct class when making decisions.

Illustrative Example(Hailee)

An example of a k-Nearest Neighbors (kNN) application is an event recommendation system. When users search for events nearby, the system can learn their preferences by analyzing their favorite choices. Using these preferences, it identifies similar event categories and recommends the most relevant events, providing personalized suggestions tailored to the user's interests.

Another example could be the wine quality classification system. Wine quality prediction could be affected by 11 factors such as acidity, residual sugar, pH, and alcohol content. To test the quality of a wine sample, we can use an existing wine quality dataset to train a kNN model and based on the accuracy of the classification fine-tune the model until it reaches an optimum k-value that yields the best accuracy (Lavasani, 2024).

Technical Part

Data Visualization(Kai)

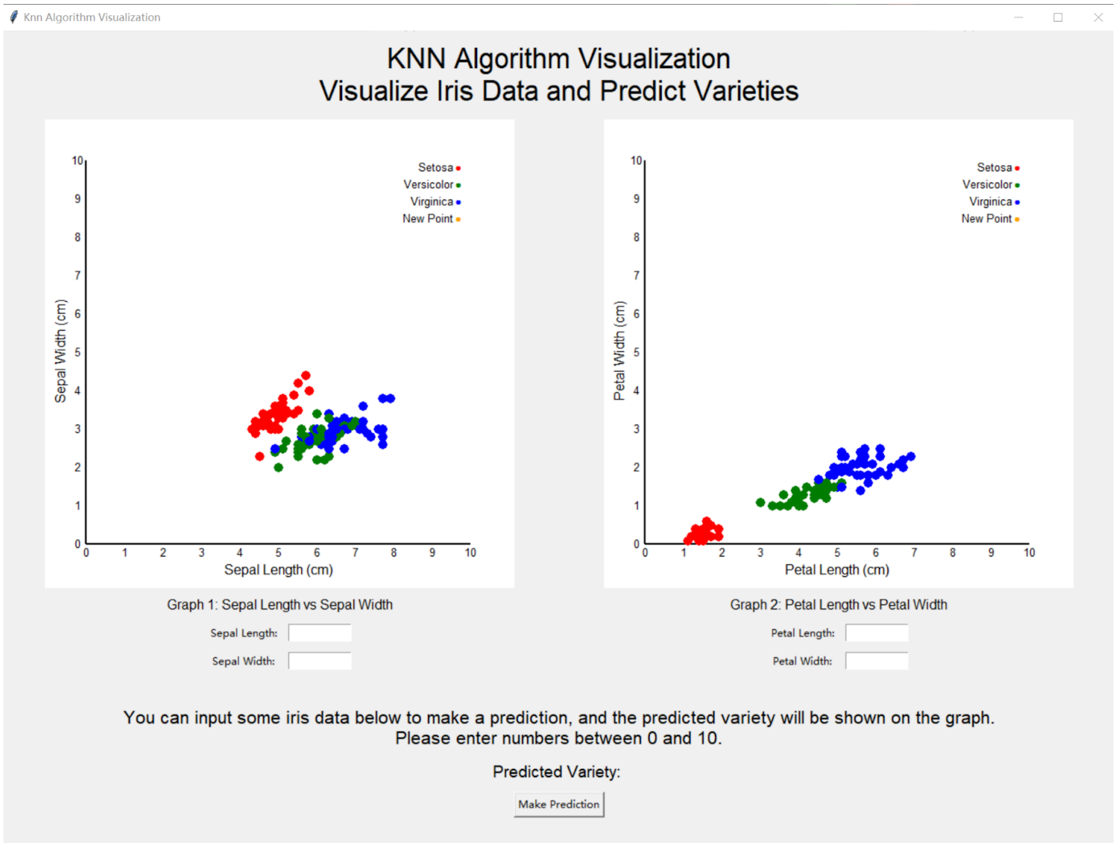
The DataVisualizationModule is designed to visualize the kNN algorithm. It shows the iris dataset on the graph to present how the kNN algorithm works. It can also help users predict species of the iris when they only know the sepal length, sepal width, petal length, and petal width of one flower.

In this module, We chose Python because it can draw the interface easily with its third-party library. (Kopczyński, n.d.)We use Tkinter, a third-party library of Python, to draw the interface to visualize the iris dataset and data inputted by the user. This third-party library can draw the graph clearly and easily. (*Tkinter — Python Interface to Tcl/Tk — Python 3.13.0 Documentation*, n.d.)

The interface is displayed below in Fig. 1. It includes:

- Interface description.

- Graphs. One shows the relationship between the iris' sepal length and sepal width, and the other displays the relationship between petal length and petal width. We use points to represent data:
  - Red points represent Setosa data.
  - Green points represent Versicolor data.
  - Blue points represent Virginica data.
  - Orange points represent newly input data.
- Input field. Users can enter sepal length, sepal width, petal length, and petal width. Every input should be numeric and between 0 and 10.
- Prediction result description. After kNN made the prediction, here will show the result.
- Make Prediction button. After inputting all four data, users can click this button and use the kNN algorithm to make a prediction. The predicted species will be displayed, and the graph will use orange to represent the newly inputted data. After entering a new point, the previous point changes color to indicate its species, allowing the user to easily compare it with other points.



KNN Algorithm Visualization
Visualize Iris Data and Predict Varieties

Graph 1: Sepal Length vs Sepal Width

Sepal Length:

Sepal Width:

Graph 2: Petal Length vs Petal Width

Petal Length:

Petal Width:

You can input some iris data below to make a prediction, and the predicted variety will be shown on the graph.
Please enter numbers between 0 and 10.

Predicted Variety:

Fig.1 Project interface

In DataVisualizationModule we implemented 5 methods: __init__, initial_graph, draw_axes, draw_point, input_predict. Each of these methods has a distinct responsibility:

- __init__: Responsible for drawing the interface which includes graphs, descriptions, input field, and button with Tkinter.
- initial_graph: Receive training data from DataProcessorModule to draw graphs of training data.
- draw_axes: Draw axes and descriptions on the graph.
- draw_point: Draw points on the graph.
- input_prediction: Receive data inputted and get species predicted by the Knn Algorithm.

The DataVisualizationModule played a foundational role in visualizing the whole project. It helps users conveniently understand the kNN algorithm.

Knn Algorithm(Hailee)

The KnnAlgorithmModule is designed to calculate the distance between data points, find the K-nearest neighbors, and categorize a given data point into a variety based on the learning results from its neighbors.

The easily readable syntax and the efficiency in coding (fewer lines of code) make Python the ideal language for programming this machine-learning algorithm. The built-in library math provides mathematical functions such as sqrt(), which is used to calculate the square root of a number (Python Software Foundation, 2024). Another built-in library collection implements container data types such as Counter, which is used for counting hashable objects. The Counter function contains the most_common method which can extract the predicted result from analyzing the neighbor data points (Python Software Foundation, 2024).

There are 4 functions in KnnAlgorithmModule: calculate_distance, k_nearest_neighbor, categorization, and calculate_accuracy.

- calculate_distance: Calculate the distance between data points based on the sepal length, sepal width, petal length, and petal width attributes. We chose Euclidean distance to measure the distance between data points because of its simplicity and its ability to effectively capture the magnitude of differences between points (Lavasani, 2024).
- k_nearest_neighbor: Find the k nearest neighbors of a given data point.
- categorization: Categorize the variety of a data point.
- calculate_accuracy: Calculate the accuracy of this k-nearest neighbor based prediction, which is used to fine-tune the algorithm. (When setting k as an integer larger than 3, we reduced the prediction accuracy significantly as k increases. When setting k as an integer smaller than 3, we diminished the purpose of using this algorithm for such prediction. With k=3, we achieved satisfactory accuracy, demonstrating the effectiveness of this approach)

This KnnAlgorithmModule is the main logical part of this project. It provides the logic/algorithm for data training, testing, and visual application.

Data Processing(Erdun)

The DataProcessorModule is one of the critical components of our project, it's designed specifically to handle raw data from the iris dataset (Shin, n.d.).

We chose Python for its simplicity, flexibility, and extensive library support (Python, n.d.). Its syntax is clean and easy to understand, making it ideal for rapid development and collaboration. Python is also rich with tools that cater to data processing and machine learning, which perfectly suits the needs of our projects.

The library's CSV (Python Docs, n.d.) and random (Python Docs, n.d.) modules are used in DataProcessorModule. CSV is a built-in library that provides efficient methods for reading and parsing CSV files, and no external dependencies are needed. We used csv.DictReader loads the dataset into a dictionary format seamlessly, enabling the data to be directly utilized programmatically without additional transformations. And random is a built-in library as well, random was employed to split the dataset into training and testing sets. Meanwhile, the use of random.shuffle() (Jain, 2022) ensured the data was evenly randomized, promoting fairness during model training and testing.

In DataProcessorModule we implemented 3 methods load_dataset, data_processing and train_test_split, each of these methods has a distinct responsibility:

- load_dataset: Responsible for loading the raw data from the Iris dataset in CSV format and preparing it in a usable structure.
- data_processing: Processes the loaded data, extracting key attributes and converting them into a consistent format suitable for analysis and algorithm implementation.
- train_test_split: Splits the processed dataset into training and testing subsets based on a defined ratio(80% training, and 20% testing (Roshan, 2022)), ensuring the model has separate data for learning and evaluation.

The DataProcessorModule played a foundational role in preparing the dataset for the subsequent stages of the project. It facilitated seamless integration with other modules, ensuring smooth progress throughout.

How To Run(Kai)

1. Make sure the Python version is later than Python 3.8.
    - Open Terminal (Linux/macOS) or Command Prompt (Windows).

- Enter the following command and press Enter:
  ```
  python --version
  ```
- If the Python version is earlier than 3.8 please go to https://www.python.org/downloads/ and install Python.

2. Clone the project repository to your local machine:
   - git clone https://github.com/ErdunE/CS5800-Final-Project.git
3. Make sure Tkinter is installed
   - Run the following code to check if Tkinter is available:
     ```
     try:
         import tkinter
         print("Tkinter version:", tkinter.TkVersion)
     except ImportError:
         print("Tkinter is not installed.")
     ```
   - If Tkinter is available, the terminal will output Tkinter is not installed.
4. If Tkinter is not installed. Run the following command:
   - python -m tkinter

5. Run the Main Program and launch the interface
   - Run the main script to launch the application:
     ```
     python main.py
     ```

Input Formatting(Kai)

Users can input iris data, including sepal length, sepal width, petal length, and petal width, through the interface, input formatting are :

- Numeric
- 0 < input < 10.

Reflection(Erdun)

Despite encountering a few challenges, our project featured several highlights that contributed to a smooth and enjoyable experience. After our efforts, we also outlined key optimization plans to enhance future projects, which are the following:

What Went Well

Firstly we began with a well-structured timeline and clearly defined milestones. This ensured that everyone understood their responsibilities from the outset. Secondly, the team discussions were productive, with each member contributing unique ideas and perspectives. Next, we had good peer reviews for each other's code, which ensured everyone was on the same page and improved overall code quality. To ensure smooth and efficient teamwork, we decided to collaborate on GitHub, which allows us to track progress, resolve conflicts, and review code effectively (*Best Practices for Organizations*, n.d.). Finally, the UI goes through iterations to allow users to enter and visualize new data points, which makes it intuitive and user-friendly.

What Didn't Go Well

Firstly, the limitations of the dataset were apparent, the scope of the project was overly narrow. It was designed exclusively for the Iris dataset, leaving it incapable of accommodating datasets with different structures or larger sizes. Secondly, the team's experience with collaborative development was limited. This led to inefficiencies, particularly when resolving merge conflicts. Finally, the testing coverage was insufficient. While the project was functional, our tests did not account for all edge cases, leaving potential room for unexpected bugs.

What Could Be Improved

To begin with, incorporating a variety of datasets could significantly improve the model's adaptability and enable the exploration of a wider range of use cases. Next, enhancing team collaboration is essential. We can reduce miscommunication and boost productivity by practicing GitHub workflows and IDE to standardize procedures. Another area for improvement is testing. Writing more comprehensive test cases, especially those that address edge cases, would make the project more robust and minimize errors. Lastly, increasing the generalizability of the project by modifying it to accept datasets with varying structures and minimal configuration would greatly expand its usability.

References

Best practices for organizations. (n.d.). GitHub Docs. Retrieved November 20, 2024, from https://docs.github.com/en/organizations/collaborating-with-groups-in-organizations/best-practices-for-organizations *csv — CSV File Reading and Writing — Python 3.13.0 documentation*. (n.d.). Python Docs. Retrieved November 20, 2024, from https://docs.python.org/3/library/csv.html

Fix, E., & Hodges, J. L. (1989). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. International Statistical Review / Revue Internationale de Statistique, 57(3), 238–247. https://doi.org/10.2307/1403797

Jain, S. (2022, December 19). *Shuffle a given array using Fisher–Yates shuffle Algorithm*. GeeksforGeeks. Retrieved November 20, 2024, from https://www.geeksforgeeks.org/shuffle-a-given-array-using-fisher-yates-shuffle-algorithm

J. M. Keller, M. R. Gray and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 4, pp. 580-585, July-Aug. 1985, doi: 10.1109/TSMC.1985.6313426.

Lavasani, A. (2024, February 6). Classic Machine Learning in Python: K-Nearest Neighbors (KNN): Proximity-Based Predictions. Retrieved November 21, 2024, from https://medium.com/@amirm.lavasani/classic-machine-learning-in-python-k-nearest-neighbors-knn-a06fbfaaf80a

Python Software Foundation. (2024, November 21). Math — Mathematical functions. Python.org. Retrieved November 21, 2024, from https://docs.python.org/3/library/math.html

random — Generate pseudo-random numbers — Python 3.13.0 documentation. (n.d.). Python Docs. Retrieved November 20, 2024, from https://docs.python.org/3/library/random.html

Shin, J. (n.d.). *iris.csv · GitHub*. GitHub Gist. Retrieved November 20, 2024, from https://gist.github.com/netj/8836201

T. Cover and P. Hart, "Nearest neighbor pattern classification," in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.

Executive Summary. (n.d.). What is Python? Python. Retrieved November 20, 2024, from https://www.python.org/doc/essays/blurb/

Python Docs. (n.d.). tkinter — Python interface to Tcl/Tk — Python 3.13.0 documentation. Retrieved November 22, 2024, from https://docs.python.org/3/library/tkinter.html

Kopczyński, E. (n.d.). Patterns - Implementing Graphs | Python.org. Python. Retrieved November 22, 2024, from https://www.python.org/doc/essays/graphs/