The Soccer Performance Recommendation System

**Overview**

My project, the soccer performance recommendation system, utilizes artificial intelligence to recommend performance improvements for soccer players and how to achieve those improvements. This system came about due to the rising prevalence of tracking systems in soccer specifically, which provides coaches with extensive data on player performance during games and practices. The system allows users to analyze this data by exporting game metrics for players and inputting them into the system, following the format of the provided example game data file.

There are two main stages to the system. First, the system groups players using clustering based on a user-provided dataset. Then, a random forest classifier, trained on labeled game data from the 2023 UNA soccer season, predicts the recommended class (category) of soccer drills for each cluster. Ultimately, the system displays the players in each cluster, the recommended drill category, and a description of the category recommended. This system offers coaches and players tailored drill recommendations to improve performance and will hopefully allow tracking data to be more beneficial to soccer teams.

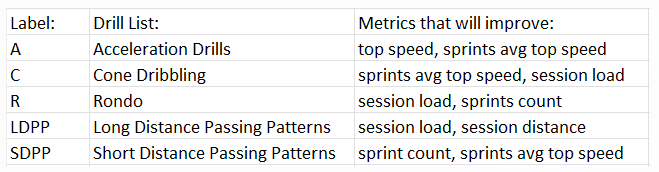
**Methodology**

Data Collection

To begin this project, I gathered performance data from TITAN (one of the many platforms that provide the trackers and interface to track player performance). Using the administrator access provided by the coaching staff, I exported data from the current 2024 UNA soccer season. This export created a massive Excel spreadsheet that became the foundation for the test data in this project.

From this dataset, I decided to focus exclusively on data from games since games are where players aim to demonstrate improvements in their performance. I selected the following metrics to include in the dataset; session distance (mi), session active time (min), session load, session top speed (mi/hr), sprints count, and sprints average top speed (mi/hr).

In addition to the 2024 data, I collected data from TITAN of the previous players of the 2023 season and the drill they utilized to increase their performance to create a labeled data set. To simplify the classification I predefined five drill categories (displayed below with their labels) before questioning previous players. After categorizing the drills, I exported the 2023 data, filtering it to match the selected metrics and keeping only data from games. Finally, I added a new column to the dataset to label all entries for each player with their assigned drill category (all entries for a player given the same label).



Clustering

The next step involved developing a program that utilizes the Kmeans++ algorithm to cluster players into groupings based on their performance metrics. Since every player had multiple entries in the dataset corresponding to different games, the program first assigned a cluster to each game entry.

Then a unique player list was created, gathering all cluster assignments for each unique player. The program then identified the dominant cluster for each player, which was the cluster most frequently assigned across their game entries. Lastly, the program calculated the average metrics for all players within each cluster. These cluster averages will eventually be used later to predict the recommended drill category.

Classifying

With the player groupings established, the next phase focused on building a program to train and test a classifier using the labeled dataset of previous players. Since the dataset labels were in string format, they first needed to be encoded into numerical values for the classification algorithms to work.

Initially, I implemented a decision tree classifier and evaluated its performance using metrics such as precision, recall, f1-score, and accuracy on both the training and test datasets. After observing low performance, I decided to switch to a random forest classifier in an attempt to improve performance. However, accuracy remained low. To address this, I expanded the labeled dataset to include both practice and game data to hopefully enhance accuracy with a larger dataset. Unfortunately, this approach led to a decrease in performance for both the decision tree and random forest classifiers. This result may have been due to significant differences between practice and game data, particularly in metrics like session load and distance.

Ultimately, I reverted to using only game data and decided to use the classifier that had the best performance, which was the random forest classifier. With a maximum depth of 20 and a test size of 25%, the classifier achieved an accuracy of 44%. While not the best-case scenario, this choice was the best due to the constraints of the dataset.

Integration of Clustering and Classification

The next phase involved integrating the clustering and classifying programs into one program. The program was designed to cluster players based on their performance metrics and then predict the recommended drill category for each cluster using the average metrics of all players within the cluster. To ensure the program’s functionality and reliability, I conducted thorough testing using the data set of current UNA players. This allowed me to validate the system’s ability to effectively group players and generate accurate drill recommendations for each cluster.

GUI Development and Testing

The final step of the project was to enhance the program by incorporating a graphical user interface (GUI) built with Python’s Tkinter library. The basic format of the GUI included a drag-and-drop box for uploading the dataset to be analyzed, a button to start the program and process the uploaded data, and a detailed output section showing the clusters, the predicted drill category for each cluster, and its benefits and example exercises.

To ensure the proper operation of the program, the GUI was programmed to verify that the necessary files (the labeled dataset of previous players and the dataset of current players) were in the same directory as the program. If these files were missing, the GUI displayed an error message informing the user, while allowing the program to proceed once the files were correctly located. Additionally, the GUI validated that the user-provided dataset was a CSV file. Any non-CSV file attempts triggered an error message.

To confirm the program’s compatibility with different datasets in the correct format, I generated a fake dataset of 20 players. This dataset contained random values within the maximum and minimum ranges of each metric from the current UNA player dataset. I used this fake dataset to thoroughly test the program to ensure it performed as expected.

**What I Did and Didn’t Do**

I did not write this code from scratch.

For the clustering phase of the project:

* A program taken from Dr. Terwilliger's AI class (10/3) provided the code to load input data and to train and create a KMeans object
* Reference #7 demonstrated how to use the kmeans labels attribute, which I used to store each player’s cluster
* Reference #8 provided an example of how to use the NumPy unique function, which I used to find all the unique players from the player list
* Reference #9 showed how to iterate over a list using enumerate, which I used to iterate through all the player entries while accessing the index of the player's list
* Reference #5 provided an example of using the index method which I used to find the index of a specific player from the unique players list
* Reference #10 demonstrated how to convert a NumPy array to a list, which I used to convert the NumPy array of unique players into a list
* Reference #12 provided the code to create a function to find the most frequent element in a list, which I used to find the most frequent cluster assigned to each player
* Reference #13 showed how to use the NumPy vstack function to stack multiple arrays into one array, which I used to create one stack of data for all the players in each cluster
* Reference #14 provided an example of using the NumPy mean function to find the average of all columns of a NumPy array, which I used to calculate the average of all data in each cluster

For the classifying phase of the project:

* A program taken from Dr. Terwilliger's AI class (9/17) provided the code for creating the label encoder object, fitting it with labels, encoding a set of labels, and decoding a set of labels
* A program taken from Dr. Terwilliger's AI class (10/1) provided the code for loading the input data, ensembling the learning classifier, evaluating the classifier performance, and computing the confidence for the cluster averages

For the GUI development phase of the project:

* Reference #15 provided the code to import the needed modules/libraries and to use a list box to create the drag-and-drop functionality, which I used in my project
* Reference #16 provided the code to create the Tk object named window, set the title, create buttons, create labels, utilize the grid method to place the widgets on the screen, create event handlers, configure widgets, and use the mainloop method
* Reference #18 demonstrated how to change the font of widgets, which I used throughout the program
* Reference #28 showed how to horizontally center widgets, which I used to configure the grid to center widgets
* Reference #30 provided the tkinter color names, which I used to create a list of colors I used throughout the program
* Reference #26 provided the code to check if a file exists, which I used to confirm the file dropped into the GUI is in the current directory as the program
* Reference #20 demonstrated how to use the cget method to get widget text or background color, which I used to get the list box color background (to see if it was green to confirm if the file was verified) and button text (to see if it was set to complete and had been run before)
* Reference #19 provided the code to change the size of widget text, which I used throughout the program
* Reference #23 defined the common method size and get, which I used to find the current index of the last item in the list box and to see if it was the first time a file was dropped into the list box
* Reference #22 showed how to use the strip method to remove leading and trailing characters, which I used to remove the curly brackets from the file name
* Reference #21 demonstrated how to use the endswith method, which I used to verify the file dropped into the GUI was a CSV file
* Reference #32 provided the code to get the file name from the file path, which I used to get the file dropped into the GUI’s name
* Reference #29 showed how to create and insert text into the text widget, which I used to display the drill category description
* Reference #27 provided the code to make a text widget read-only, which I used to prevent the user from entering text into the drill category description
* Reference #31 demonstrated how to change the parameters of a label widget using the config function, which I used throughout the program

**Final Results**

In reference to the goals set in my term project proposal, the system successfully groups players on a team into clusters, utilizes past TITAN data from the 2023 season to predict suggested exercises for each cluster, and displays the results in a GUI created with Python’s Tkinter library. While I achieved all the goals set in my project proposal, one significant aspect of the project that was not up to standard was the accuracy of the classifier responsible for predicting the drill categories for each cluster. Despite my best efforts, this remained a major obstacle in the final implementation of my project. If I were to continue developing this project, improving the classifier’s accuracy would be my top priority. On top of this, I would also attempt to transition my project to a web-based interface to make it more convenient for users to access it. Through this project, I gained experience creating GUIs with the Tkinter library, handling real-world datasets, and designing functional systems that incorporate AI.

**References**

References for classifying and clustering:

1. Base code for using random forest classifier taken from Dr. Terwilliger's AI class (10/1)
2. Base code for clustering taken from Dr. Terwilliger's AI class (10/3)
3. Base code for using the label encoder taken from Dr. Terwilliger's AI class (9/17)
4. https://www.geeksforgeeks.org/numpygenfromtxt/
5. https://www.freecodecamp.org/news/python-find-in-list-how-to-find-the-index-of-an-item-or-element-in-a-list/
6. https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.LabelEncoder.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans
8. https://numpy.org/doc/stable/reference/generated/numpy.unique.html
9. https://www.geeksforgeeks.org/iterate-over-a-list-in-python/
10. https://www.geeksforgeeks.org/how-to-convert-numpy-array-to-list/
11. https://pythontextbook.com/chapter-5/
12. https://www.geeksforgeeks.org/python-find-most-frequent-element-in-a-list/
13. https://www.geeksforgeeks.org/numpy-vstack-in-python/?ref=next\_article
14. https://www.codecademy.com/learn/ida-3-introduction-to-numpy/modules/ida-3-2-numpy-syntax/cheatsheet

References for tkinter GUI**:**

1. https://stackoverflow.com/questions/14267900/drag-and-drop-explorer-files-to-tkinter-entry-widget
2. https://pythontextbook.com/chapter-13/
3. https://www.geeksforgeeks.org/python-gui-tkinter/
4. https://www.geeksforgeeks.org/tkinter-fonts/
5. https://www.tutorialspoint.com/how-to-change-the-size-of-text-on-a-label-in-tkinter
6. https://www.geeksforgeeks.org/how-to-get-the-tkinter-label-text/
7. https://www.tutorialspoint.com/python/string\_endswith.htm
8. https://www.geeksforgeeks.org/python-string-strip/
9. https://www.geeksforgeeks.org/python-tkinter-listbox-widget/
10. https://www.tutorialspoint.com/how-to-fully-change-the-color-of-a-tkinter-listbox
11. https://www.geeksforgeeks.org/python-grid-method-in-tkinter/
12. https://pythonistaplanet.com/python-file-io-exercises/
13. https://www.tutorialspoint.com/how-to-make-the-tkinter-text-widget-read-only
14. https://www.geeksforgeeks.org/horizontally-center-a-widget-using-tkinter/
15. https://www.geeksforgeeks.org/python-tkinter-text-widget/
16. https://cs111.wellesley.edu/archive/cs111\_fall14/public\_html/labs/lab12/tkintercolor.html
17. https://www.geeksforgeeks.org/change-the-color-of-a-tkinter-label-programmatically/
18. https://www.geeksforgeeks.org/python-program-to-get-the-file-name-from-the-file-path/