AI Modeling for Auburn University Baseball Team

Developer Documentation

Authors:

*Isaiah Gallardo*

*Chase Garner*

*Sarah Holland*

Comp 4710 Senior Design Team #10

Department of Computer Science and Software Engineering Samuel Ginn College of Engineering, Auburn University

Table of Contents

1. **Architecture 3**
2. **Structure 4**
3. **Interfaces 5**
4. **Assumptions and Dependencies 7**

# Architecture

Our heatmaps were created in IntelliJ. We utilized Python in order to create everything and to integrate with the database team, we connect to the database and write queries in SQL. The database team writes our images to display on the website using Javascript, so we tried to ensure our code for displaying heatmaps was language agnostic and did not rely on any Python-specific libraries.

The heatmaps can be used by anyone on Auburn’s campus, but the two main groups we considered were coaches of the baseball team and players on the baseball team. These two groups can use the heatmaps in similar ways, but draw different conclusions from them. The most important element for our group was ensuring the heatmaps were easy to understand, so we tried not to complicate them with unnecessary elements.

\

# Structure

Our application is split into two parts, .py files with utility functions for the different features we are implementing and then the updateheatmaps.py file for integrating our project into the database. So far for our utility modules we have developed heatmap.py and reportutils.py. All code for our project can be accessed on our GitHub repository.

The first module, heatmap.py, implements the heatmaps and related functions. The heat maps are constructed with the different combinations of batters, pitchers, and pitch types. The heatmap function takes the data along with the ranges of both axes and the desired resolution of the heatmap. From this, it generates a 2D array representing the heatmap. This heat map array can be visualized using libraries like pyplot or any other graphing library.

The second module, reportutils.py, deals with genrating data and the .tex file to be able to generate a pdf report for all the pitchers. The gen\_report\_data() function first generates and saves all the data for the report- heatmaps, pitcher statistics, etc. The report\_to\_latex() function generates the .tex file that uses the data generated from the gen\_report\_data() function. This .tex file can then be compiled to a pdf with any latex compiler.

Lastly, the updateheatmaps.py file handles connecting to the database and updating the heatmaps stored there. It first queries the pitch data for each pitcher on each team, then it calculates the heatmaps, and finally inserts them into the database for later use. These heatmaps are then displayed by the database team.

# Interface

For our interface, we rely on a connection to the database. They then handle displaying our heatmaps on the website. In terms of files we use in order to connect to the database and perform other activities in our project, we have listed them below. We also included detailed documentation of the functions below and within our codebase.

* heatmap.py - Class to create heatmaps for each different type of pitch for each pitcher
  + heatmap() - Outputs an array corresponding to the inputted resolution and size that represents the heatmap.
  + normalize() - Normalizes a heatmap so everything is on the same scale
  + div\_heatmap() - Divides the values within two heatmaps
  + save\_to\_image() - Saves a given heatmap as an image that we can display and use to convene with the client
* updateheatmaps.py - Connects to the database, queries the necessary tables, and runs our heatmap functions on these tables. Then it writes these tables back to the database. The database team then handles transforming these arrays into svg images and displaying them on the UI.
* main.py - Runs the code in the reportutils.py file and generates a latex file with a report of heatmaps for each pitcher. Helped us to show the client our progress before it was uploaded to the database.
* reportutils.py - Class of utility functions that help us create a report to give to the client. This was created in case we were not able to integrate with the database in time; it still allows us to deliver something to the client.
  + gen\_report\_text() - Generates the text statistics included for each pitcher.
  + gen\_report\_images() - Generates the images of all the heatmaps for a certain pitcher-pitch type combination.
  + gen\_report\_data() - Combines all of the photos and text statistics for all pitchers and complies into a report.
  + report\_to\_latex() - Function to actually create the report and format it in LaTex. Provides a LaTex file which we can then transform into a PDF.
* db\_info.txt - A text file specifying connection parameters for the database. This includes the username, password, and other important information. This file is critical to ensuring successful connection and updating our heatmaps.

# Assumptions and Dependencies

* Data Getta should be browser-agnostic, and renders on all browsers.
* Mobile device compatibility is out of scope at this time.
* The connection to the database and updating of heatmaps requires there be a file in the same directory as our code called “db\_info.txt” which specifies connection parameters to the database.
* We use the following versions and packages. This is important for consistency and ensuring the correct methods are being used. Our heatmaps depend on the proper functioning of these libraries.
  + psycopg2-binary — 2.9.9
  + numpy — 1.26.0
  + matplotlib — 3.6.3
  + pandas — 2.1.1