Standardizing Stepmania Song Difficulties Using Regression

Business Understanding

- **Domain Introduction**
  Stepmania is an open-source software designed to support many styles of rhythm game play, but one of its most widely-used formats is for 4-panel “dance games” (well-known “dance game” examples are Dance Dance Revolution and In The Groove, but these are specifically-licensed games by Konami and Roxor). The software was released in 2001 and has since been updated through version 5.3 - the concept behind this software is to allow players to play a “DDR-like” game with virtually boundless customizations, including custom charts (arrow patterns for songs).

- **Business Problem**
  The problem that I aim to solve is using exploratory data analysis and statistical learning to help establish a standardization for song chart difficulty ratings based on various features of the chart, for example BPM, duration, and quantity of technical elements.

- **Motivation**
  While there are many respectable step-artists with expertise in their domains, there has not been an official and standardized rating system for song difficulties - it is not uncommon to see players in conversation over whether a song should be “re-rated” due to various features of its chart. Creating a method of standardization will help examine which features are the most important in determining difficulty and help step-artists remove the intuitive/guesswork element from rating their own charts.

  The community of rhythm game players is surprisingly large and the amount of dedication that this community has poured into creating an open-source rendition of rhythm games is incredible. This software is free and is used for tournament play and for writing charts and is a staple to any outside-of-arcade dance game player, and is sometimes employed in small, local arcades as well.

- **Domain Understanding**
  There are no similar projects in the realm of data analysis, although there have been projects that aimed to analyze the stepcharts themselves to essentially perform a tally of each individual technical element.
  Due to Stepmania being open-source, many people have ventured into writing charts for songs (the 4-arrow patterns that players step on) and a culture has evolved for

different styles and subdivisions of this game - some people focus on “stamina,” which is high-volume and high-density charts designed to test a player’s ability to push their physical limits. Another focus is “tech” or highly-technical play, designed to test a player’s ability to unpack and execute difficult patterns with elements such as BPM changes, “mines,” crossovers, footswitches, etc.
These features will be used as the predictive elements of my model.

Data Understanding

- **Features to Examine**
    - Song duration
    - BPM
    - Total number of steps
    - Number of changes in BPM, if any
    - Number of jumps, hands, and quads (aka step on 2, 3, or 4 arrows at a time)
    - Song density (number of notes/steps per second of the song)
    - Peak NPS (the highest density in a song)
    - Number of song measures containing a “stream” - continued steps with no break valued at a 16th note or higher
    - Number of crossovers, footswitches, jacks, sideswitches, and brackets (each of these is a technical element requiring a particular style of stepping)
    - Number of “mines” (arrows that are meant to be purposefully avoided due to them “exploding” if stepped on and decreasing the player’s score)
    - Classification of whether the chart is a “stamina” chart or not - this is one of the most important distinctions because difficulty ratings for stamina songs look very different from other songs. Stamina songs typically start north of 2.5 minutes long and some even exceed several hours! To address this kind of outlying information, it may be worth splitting the data into “stamina” data and “non-stamina” data.

- **Data Acquisition**
  itgpacks.com is a community-maintained spreadsheet containing the majority of song “packs” released in the last five years. There are step-artists who are reputable for consistent and seemingly-accurate ratings due to high domain knowledge, and I plan to use packs that they have released.
  The data will be extracted from each individual file using a parser written in Python that will collect the above information and store it in a .csv file. This parser is currently a

work in progress as a collaborative effort between myself and a software engineer and fellow dance game player and will extract the features in the exact format that is intended to be used in a DataFrame.
This specific problem has never been worked on before, although there have been chart analysis algorithms written to handle the counting of technical elements in each chart (brackets, footswitches, etc).

Data Preparation

- **Storage**
  The data will be stored in a .csv file with largely continuous values and a select few categorical variables. I anticipate the distribution of songs by their difficulties to be fairly normal (i.e., most of the difficulties will be in the "medium" range of 9-12, while easier and harder charts will be much less frequent.

- **Preprocessing and Exploratory Data Analysis**
  Due to having direct control of how each chart is parsed, cleaning of data will likely be minimal. There will likely be some encoding of categorical features as well as ensuring all data types are consistent.
  The base goal is to collect at a minimum 1,500 individual rows of data, which would be an estimated 650 - 750 songs, as most songs have at least two charts of varying difficulties.
  I plan to visualize individual components of songs in relation to their difficulties via scatter plot as well as a distribution of difficulty ratings using histograms, as well as introduce the concept of different chart styles.

Modeling

My initial inclination was to use linear regression modeling due to the potential for the difficulty rating to be a continuous variable. With that said, song ratings fall most often between 5 and 22 so I did also consider using multi-class classification.
For multi-class classification, I plan to use logistic regression or a neural net with a sigmoid activation function, with "song difficulty" as the target variable and logistic regression used as the baseline model.

Evaluation

- **Metrics**
  Due to the lack of a distinct "cost," the metric of focus will be accuracy and f1 score in the event that this remains a classification project.

- **Minimum Viable Project**
    - The MVP involves a Jupyter notebook detailing the parsing of chart data by bringing in the parser to show some individual examples, data exploration, a final tuned model, and feature descriptions detailing which chart features seemed like the highest contributors to a chart's difficulty.
      It will also include an option for the user to input their own chart data (either by using the parser or their own generated information) to create a prediction of difficulty.

    - A smaller project would be to store all of the data in a dataframe, perform EDA, establish a baseline model, tune the model, and present the results of which features of the chart are most significant in predicting its difficulty.

- **Stretch Goals**
    - Ideally, I would like to host this information and model online and allow users to interact with it. The goal is to use Flask for this portion of the project. I also would like to incorporate a blog post detailing the data acquisition and modeling steps and describing the meaningful implications of this project.

Deployment

- Reporting and Deployment
  The final results will be in a github repository and jupyter notebook. My goal for deploying it would be by using Flask to create a web application. The ideal functionality of this project is to allow interested users to input a chart file to be parsed, or to input individual statistics into the model in order to predict a difficulty rating.

Tools/Methodologies

- **Tools**
    - Numpy
    - Pandas
    - Sci-Kit Learn
    - Statsmodels
    - Seaborn
    - TensorFlow/Keras

- **Algorithms and Analysis**
  I am interested in testing the differences between using a multi-class logistic regression model from SKLearn and Tensorflow's deep neural net model. Both data storage and analysis will be stored on my local machine.

References:

- Example of stamina play:
    - https://www.youtube.com/watch?v=HMfgbACzU90

- Example of technical play:
    - https://www.youtube.com/watch?v=Ryk0LrlsKaI

- Simply Love theme for Stepmania (the "industry standard" for running in conjunction with Stepmania software):
    - https://github.com/Simply-Love/Simply-Love-SM5

- Github repo detailing a chart analysis algorithm used as a basis for the parser:
    - https://github.com/Simply-Love/Simply-Love-SM5/pull/271