PREDICTIVE ESRB ASSESSMENT TOOL

Requirements Specification

Abstract

Given the consistently increasing complexity of interactive entertainment and the sheer volume of titles released yearly, ensuring that video games are efficiently and appropriately rated is a critical endeavor for both developers and publishers. The process traditionally involves manual assessment by industry experts, which can be time-consuming and resource intensive. By harnessing the power of advanced machine learning algorithms, the Predictive ESRB Assessment Tool streamlines the rating prediction process, providing accurate and expedited preliminary ratings for video games. This more contemporary solution not only reduces the time required for developers and publishers to receive crucial ratings but also empowers them to make informed release plans with confidence.

Glossary

Entertainment Software Rating Board (ESRB)

A non-profit organization that assigns age and content ratings to video games and apps, indicating their suitability for different audiences.

Content Descriptors

Specific elements or themes within a video game, such as violence, language, or sexual content that are used by the ESRB to provide additional information about the content of a game.

Classifier

A machine learning algorithm or model that is trained to classify or categorize data into specific groups or categories.

Cross-Validation

A technique used in machine learning to assess the performance of a model by splitting the data into multiple subsets for training and testing.

User Requirements

Functional: Feedback Submission.

"As a user, I want to be able to provide feedback to the system administrators to report issues, offer suggestions, or comment on my experience."

Users should have the ability to submit feedback through a designated form within the application. Providing a feedback mechanism allows users to communicate their experiences, report issues, and suggest improvements, contributing to the system's overall quality and user satisfaction.

Acceptance Criteria

1. The application should include a feedback submission form accessible from the user interface.
2. The feedback form should capture the following information:
    a. User's name
    b. User's email address
    c. Feedback category (e.g., bug report, suggestions, general comments)
    d. Detailed feedback or description
    e. Optional file attachment

Verification

To ensure that a user has successfully submitted their form, a confirmation receipt should be generated.

Non-Functional: Transparency & Compliance with User Expectation

"As a user, I want to have access to clear and understandable explanations regarding the factors that influence the ratings of video games in the system. Additionally, I want the system to flag any content of interest or that falls under high-interest categories and notify me about it. This will help me make informed decisions about the games I engage with."

Transparency should be a key pillar stone in any application and our system will provide clear and understandable explanations as a rubric for what factors may influence ratings. Our goal is to serve information that users will consider useful in understanding rating prediction. The user will also be notified of any content that may be flagged of interest or falls under a high interest category.

Acceptance Criteria

1. Clear access to explanations
2. Flagging of Content
3. User notification of flagged content
4. User Response
5. Verification of rubric and links for verification

Verification

The rubric will match exactly what is documented by the official ratings guide of ESRB. The User will be provided with two links. One link that is reflective of the rubric in our application and another link to the ESRB search, where a game in question can be freely searched to see if the categories of the selected game match with the previously categorized games in the same category.

Functional: Recommending Games with Similar and Dissimilar Content.

"As a user, I want the system to recommend games to me based on content similarity. I want to discover games that share similar themes or characteristics with a selected game, as well as games that offer entirely different content for a varied gaming experience."

Users will have the ability to receive recommendations for games that have similar content to a selected game and games that do not share similar content. This feature enhances user engagement by suggesting games that align with the user's preferences or offer new experiences.

Acceptance Criteria

1. User is presented with a list of games that share similar/dissimilar content/themes in a user-friendly manner, including game titles and their associated categories.
2. Our system will highlight what features were most important in making the decision.

Verification

The user can go to the feedback form and submit a discrepancy, detailing the specifics of their issue which can be later reviewed by the team internally. This relevancy of each game can also be verified on ESRB's official website, which will be linked, where an end user can filter their game by relevant categories to see if the recommended games are also in that category.

System Requirements

Functional: Classify an Arbitrary Game

The system shall accurately classify and provide an ESRB rating prediction for any game, including those not present in the system's data set.

Function	Classification algorithm
Description	Accurately (>=90%) classifies game given specific in game criteria.
Inputs	The classification algorithm accepts data set – each data point represents a game to be classified.
Source	Dataset as a .csv
Outputs	Classification results for each game structured in a Random Forest
Destination	Main Control Loop

Precondition	The dataset is properly labeled choosing the most relevant attributes.
Action	The dataset is loaded into the program with necessary attributes. Split the dataset into training and testing sets. Then create a Random Forest classifier to create trees to hold the results. The classifier is used to train the data. We then use the trained model to calculate the accuracy and make predictions. Finally, we generate a report that contains detailed metrics for each class in the dataset.
Postcondition	Each data point is assigned to a category to allow for the validation of the results.
Side effects	N/A

Acceptance Criteria

1. The system should be able to classify a game with an accuracy greater than 90%. The spec will further clarify how this accuracy is to be achieved, further in the document.
2. The system should be able to classify a game within a reasonable amount of time < 5 seconds.
3. The system should be able to make this prediction with high confidence by comparing it to titles already in the system that have similar themes.

Verification

To fulfill the acceptance criteria, the system must keep track of relevant fields and their associated categories, highlighting overlap. This can be checked by keeping track of the arbitrary game's features and finding if most of these fields are a subset of a similar game within the system to suggest high confidence. Confidence can be measured by checking the arbitrary game against games with similar themes having an overlap of over 75%. If the overlap is above 75% then it is considered a success. *

**Please note that accuracy of prediction does not equal the percentage of overlap. Accuracy can be higher than the amount of overlap given that features in a similar category can lead to the same conclusion, without the need for both to be in the game's content descriptors.*

Functional: Error Handling with Logging

To keep our prediction model running optimally and working as expected, the program will check for bugs using conditional statements and error handling constructs.

Function	Detecting and alerting users to errors
Description	Using error codes and conditional statements, users will be alerted to the program malfunctioning and a log will be created detailing what needs to be addressed by the development team.
Inputs	Dataset and user input
Source	Dataset .csv file and user input
Outputs	Text files detailing improper behavior/bugs
Destination	Main Control Loop
Precondition	Identity and categorize different errors
Action	If any of the error conditions are triggered during run time, the user will be notified, and the program will output a log containing the errors and existing bugs discovered (and any other pertinent details e.g., severity level, timestamps, etc.).
Postcondition	Error and performance logs are collected for review
Side effects	N/A

Acceptance Criteria

1. Errors will be handled in a manner appropriate for their severity level.
2. Users will be given warnings if unwanted behavior occurs.
3. Any issues are collected in logs, the most critical bugs are given priority.

Verification

After the logs are collected and the bug severity levels are established, the team will verify the existence of the bug(s) and apply the appropriate modifications to the program. Once the errors are addressed, document any changes that are implemented to maintain transparency.

Functional: User Authentication/Roles & Permissions

Users will create an account to prevent unchecked access to our system's critical information and functionality. Accounts will be stored in a database and retrieved during the sign-in process.

Function	Allow user to save and access results, give users roles and permissions to limit access to critical elements of the system.
Description	The system shall provide user authentication using a username and password. There will be an option to reset password if the user forgets. Each user will have a role that determines what they are able to access/modify. The user can create, change, and delete their account
Inputs	User account information from account creation
Source	Usernames and passwords from text file passed through a User Class.
Outputs	Text file that has usernames and passwords.
Destination	Main Control Loop
Precondition	Users have created an account
Action	After the user creates an account, their credentials and permission level are stored in a data file. If the user forgets their credentials, they shall have the option to retrieve them. When the user logs in the text file is passed into an Array list of Class User. The user input will be compared to the existing database of users for verification. If user inputs incorrect information 5 times they be locked out of their account and required to change their password.
Postcondition	Users can log in, change password, and be able to access what their permission level allows. User credentials are correctly stored in database.
Side effects	N/A

Acceptance Criteria

1. Users will be able to create, alter, and delete accounts.
    a. Add entry to database.
    b. Change entry in database.
    c. Remove entry from database.
2. Each user is asked to confirm their access level so appropriate permissions are applied to their account.
3. All user information is properly stored in the database and used to verify users.

Verification

All users can make an account and experience the program in a way that is apt for their permission level. User logins will be logged to confirm there are no accounts accessing anything beyond what their permission level allows.*

Functional: Handle Missing Data Fields

The system must be able to handle situations where certain data fields are incomplete or missing, ensuring that it can accurately classify games and provide meaningful recommendations, even with limited information.

Acceptance Criteria

1. The system will keep a record of what categories are most important in making a decision, determined by how much information gain is acquired from each field during a split.
2. The table must provide the information in a digestible manner to avoid confusion when trying to draw valid conclusions and can be referenced later if needed.
3. The number of missing data fields must be enough to maintain our target accuracy of above 90%.

Verification

Given that the data fields for determining a game will be selectable via a dropdown list that is pre-determined by our team, the system will run test cases keeping track of the minimum number of fields required to maintain above our threshold of 90% accuracy via an object table.

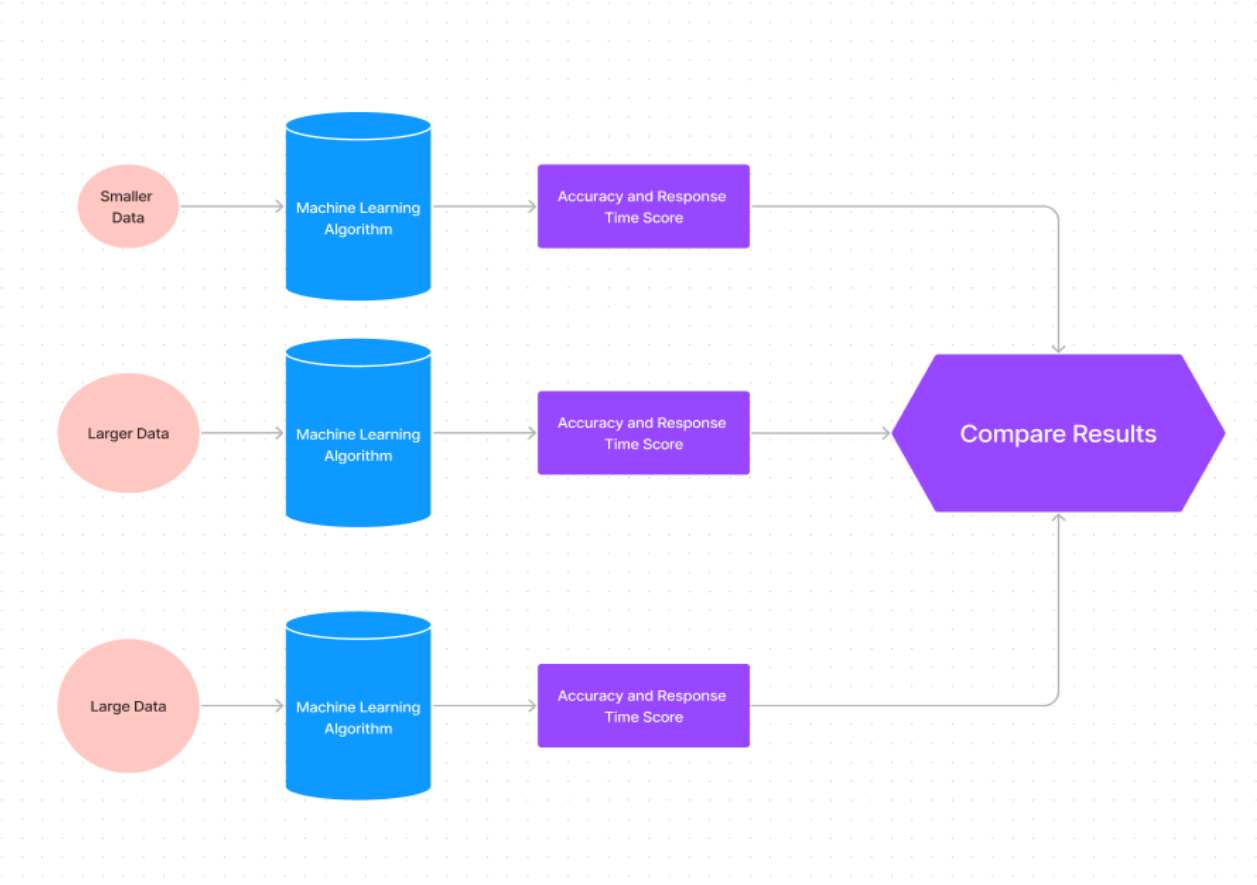
Non-Functional: Scalability

The system's scalability ensures that it can handle an increasing volume of game data and user interactions while maintaining optimal performance and responsiveness, thereby accommodating future growth and user demands. Scalability is essential for meeting the growing needs of our user base and game database. It ensures that our system remains efficient and reliable as we expand our services.

Acceptance Criteria

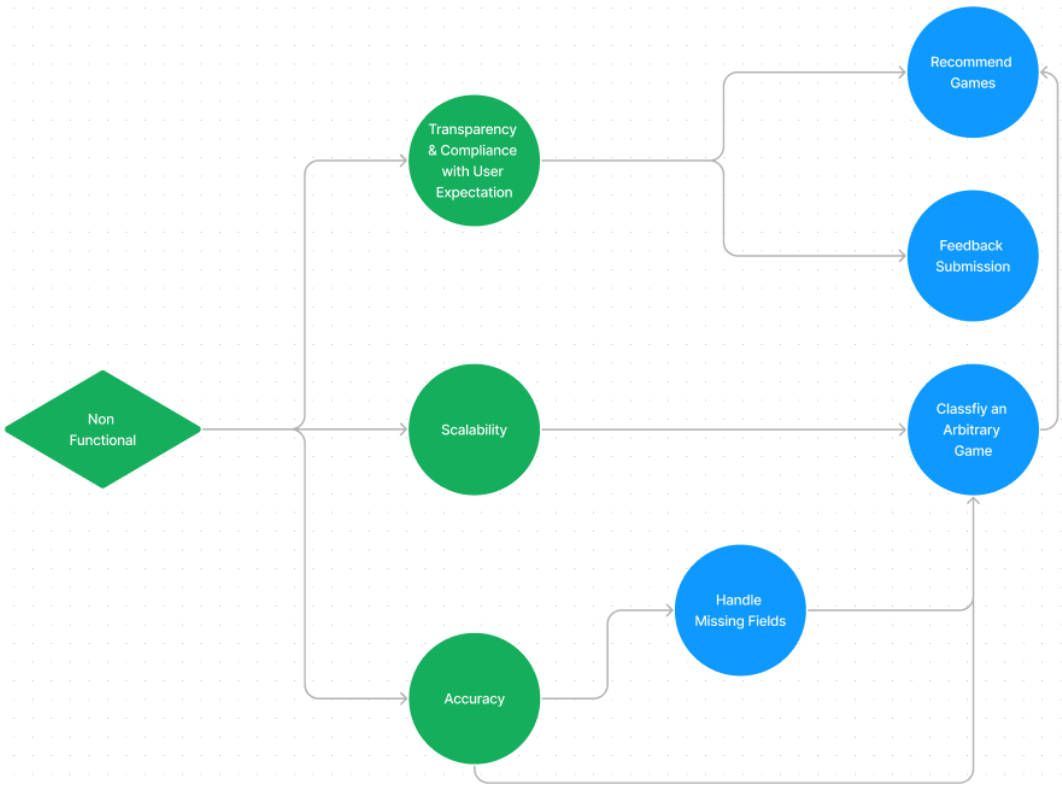
1. The system should well under subsequent requests
2. The system's response time of no more than 5 seconds.
3. As the data set grows the system should continue to maintain fast response time.

Verification



We will begin testing the system with a small load and gradually increase over time keeping track of the results. If there are any significant increases in response time, we will take note and output this to a form that will allow the team, to analyze and then optimize for retest. Identifying key metrics such as CPU response time and memory utilization will allow our team to identify any bottlenecks.

Functional & Non-Functional Mapping



Development Team 6

Ali Elayni

Richard Recar

Armand Soto