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.

### ESRB raters/ ESRB rating assigners

Trained professionals who assess video games and apply the appropriate content descriptors and age-appropriate ratings based on the game’s content.

# User Requirements

## Functional: Classify Games

“As a publisher, I’d like to know in advance what sort of rating a studio’s game might receive so that I can effectively and responsibly market the title. “

The system should process the content descriptor information provided by publishers in their rating request submission form or through manual entry and predict what the ESRB classification would be for the given title using a machine learning process that is trained to draw connections between the submitted content descriptors and received ratings of hundreds of games.

## Functional: User Feedback Loop

“It’s important that my feedback be accepted during the normal operation of the program. If there’s an outstanding or poor prediction, I want the system to adjust accordingly. “

The system should provide the user with a simple way to rate its prediction and the included supplementary information about how the classification was made. By submitting a ‘thumbs up’ or ‘thumbs down’, the user is helping to make the system more accurate over time by reinforcing the content descriptors that played the biggest role in the classification.

## Functional: Training on User Provided Data

“As an ESRB reviewer, the catalogue of reviewed games that I must consider is constantly growing. I’d like to be able to keep the system up to date by including the most recent, relevant titles in its library of past reviews. “

The system should accept updated data on past reviews. A user should be able to submit a new set of data or add to existing data and retrain the program to make its predictions more accurate or relevant given the new information. It should be possible to train the system to deal with a narrow set of games such as a publisher’s portfolio, or more large, general collections like the storefront of popular online gaming platforms.

## Functional: Recommending Games with Similar and Dissimilar Content.

“I want to compare other games that have received the rating the system predicts. It should be possible to check if games with that rating typically contain the same content descriptors or see a counterexample that might influence a different rating.

The system should highlight titles that clarify and provide context for its predictions. Games with the same rating should be shown on screen with a list of the content descriptors they share. The games with the same rating that share the most or least number of content descriptors are of particular importance, as they could highlight the consistency of the classification system. Special attention should be given to content descriptors that correlate with games rated Mature or higher.

## Non-Functional: Transparency and 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. I want the system to flag any adult content so that it is immediately visible. This helps me make sure that the highest-priority content descriptors aren’t overlooked.”

The system should present the factors that influence prediction in a digestible format. 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.

# 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 | An ESRB rating |
| 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 methods to achieve this accuracy are clarified below.
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

Provide the testing set, which includes a variety of games not in the system’s existing set. Separate the testing set into two categories: games present in the system’s data set (used for validation) and games not in the system’s data set used for verification. Generate ESRB rating predictions for games not present in the data set with the actual ESRB ratings of those games provided by the official site.

## Functional: User Feedback Loop

|  |  |
| --- | --- |
| Function | User Feedback Loop |
| Description | The Recommendation Enhancement System allows users to provide feedback on recommendations (likes/dislikes) and utilizes this feedback to enhance and refine recommendations using Random Forest prediction. |
| Inputs | A Boolean value that is true if feedback was helpful, false otherwise |
| Source | User Interface (GUI) |
| Outputs | Improved Recommendations and feedback data to be stored for analysis and system improvement |
| Destination | A txt file with all features and their associated weights based on their relative information gain that is used only during re-training |
| Precondition | A prediction has been made by the system alongside a set of recommended similar and dissimilar titles |
| Action | User feedback is collected and stored. The collected feedback is preprocessed and prepared for integration into the random forest model. Features that were most important in determining information gain are assigned a higher weight for use in the next retraining. The updated model is used to generate enhanced recommendations. |
| Postcondition | The system provides enhanced recommendations that are more aligned with users’ likes and dislikes. |
| Side effects | The system may experience changes in recommendations as the model is updated, potentially affecting user engagement and satisfaction |

### Acceptance Criteria

1. The system shall store user feedback in a structured and coherent manner.
2. The system will employ user feedback to update the system, providing more accurate results.

### Verification

Team can inspect the text file where data is stored to ensure that changes are reflected when user gives feedback. To confirm that the system utilizes user feedback to enhance its accuracy, perform a testing scenario. Use the system to provide feedback for a set of recommendations. Then repeat the same input for features, but this time, modify the feedback and observe the system’s behavior to see how results differ. There should be a change in the quality or relevance of the results.

## Functional: Training on User Provided Data

|  |  |
| --- | --- |
| Function | Train on User Provided Data |
| Description | Build an ensemble of decision trees to make predictions based on user-related features or attributes |
| Inputs | The training data includes features, corresponding target variables. Hyperparameters such as the number of decision trees in the forest, max depth of tree, minimum number of samples for leaf etc. |
| Source | Dataset as a .csv |
| Outputs | Random forest model, an ensemble of decision trees that collectively for the model. |
| Destination | Main Control Loop |
| Precondition | Have a sufficiently large and labeled dataset of user-related features and corresponding target variables |
| Action | Pre-process the user data by making sure there are no missing values and normalizing numeric features to get it ready for training. Then randomly select sample data from the dataset with *replacement* Then randomly select a subset of features for that tree. Then create a decision tree using the selected data and features, recursively by minimizing impurity though calculating the Gini Index to determine information gain. |
| 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 model shall achieve an accuracy >=90% on user data.
2. The model should generalize well to unseen user data.

### Verification

Employ cross- validation on user-provided data to assess the model’s accuracy and compare the model’s performance against a pre-defined accuracy of above 90%. Model should maintain reasonable accuracy and consistency when applied to unseen user data, by testing on a “hold-out” data set not used during training.

## Functional: Recommending Games with Similar and Dissimilar Content.

|  |  |
| --- | --- |
| Function | Content Based Recommendation |
| Description | Recommend games that are both the most and least like the model’s game prediction. |
| Inputs | The model’s prediction. |
| Source | Dataset as a .csv |
| Outputs | Games that have the most features and the least features in common with the model’s prediction. |
| Destination | Main Control Loop |
| Precondition | Accurate predictions from the model. |
| Action | Once the model gives a prediction, the game (along with its feature set) is compared against the other games in the data set to give appropriate recommendations (similar and dissimilar games). |
| Postcondition | The similar and dissimilar recommendations are appropriate given the feature set of the model’s prediction. |
| Side effects | N/A |

### 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.

## Non-Functional: Error Handling with Logging / Reliability

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

### 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.

## Non-Functional: Handle Missing Data Fields

Before passing information into the prediction model, it is essential to handle missing data fields by identifying, addressing, and normalizing missing values to ensure the dataset used for training and testing is complete and representative. If the data set provided is missing a data field, we will take an “assume the worst approach” to take excess caution to ensure that inappropriate content is not presented to a younger demographic.

### Acceptance Criteria

1. The handling of missing data results in a data set where all data fields are complete.
2. The approach to all missing data is the same for all missing data (worst approach).
3. The number of missing data fields must allow above target accuracy of above 90%.

### Verification

Simulate missing data fields during testing and see if the data performs similarly with missing data fields, than without. Employ proper error handling to ensure that there are no missing fields.

## 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. Have a system response time of no more than 5 seconds.
3. As the data set grows the system should continue to maintain fast response time.

### A diagram of a diagram Description automatically generatedVerification

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.

### Development Team 6

*Ali Elayni  
Richard Recar  
Armand Soto*