Artificial Intelligence Project: Future Game Review Predictor

1. Introduction

The "Future Game Review Predictor" project describes the console-based C# program that explains core machine learning techniques. The program's primary goal is to classify game reviews as "Positive" or "Negative" using a numerical review score. The project is accomplished by starting from scratch with three different kinds of machine learning models: a Simplified Linear Regression, a Naïve Bayes Classifier, and a Multi-Layer Perceptron (MLP). The program takes user inputs, trains models, and utilizes them to anticipate the sort of review for an upcoming game. The project report emphasizes about the architecture of the system, data structure applied, the implementations of the model, workflow and logic, limitations and future work, and conclusion.

2. System Architecture

The system's design is a straightforward, linear pipeline that processes data via many critical stages:

1) Data Input

The application invites users to manually input game names, review ratings (ranging from 0.0 to 10.0) and review types (either "Positive" or "Negative"). The obtained data is used as the dataset for training and testing the models.

2) Data Preprocessing

The raw input data is transformed into a structured format that is appropriate for the models. For the numerical models (Regression and MLP), the data is transformed into DataPoint objects with a numerical feature (the review score) and a numerical label (1 for positive, 0 for negative). The Naïve Bayes model categorizes numerical scores as "low", "medium", or "high".

3) Model Training and Evaluation

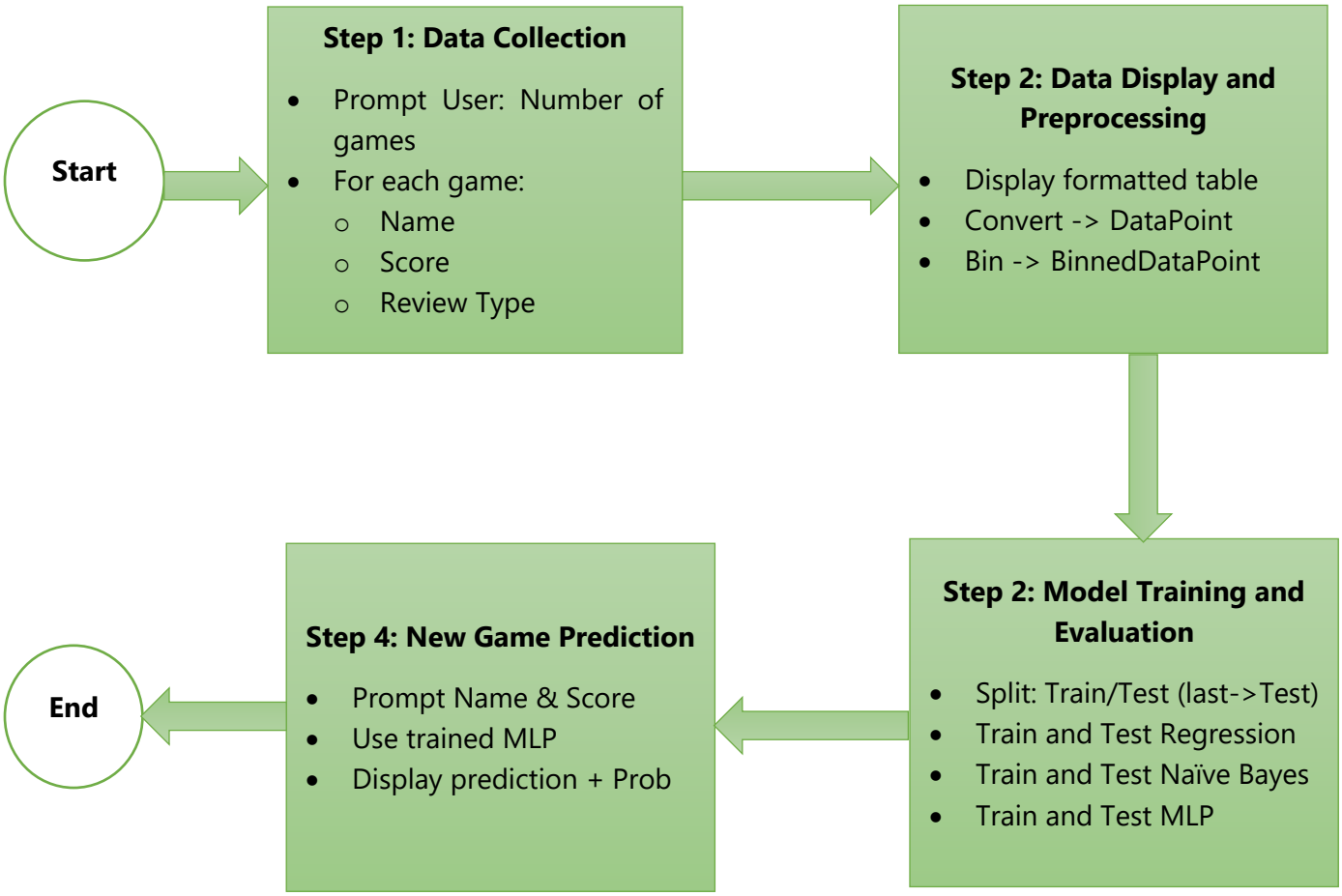
The preprocessed data is divided into training and test sets. Each of three machine learning models is trained using the training data. After training, each model's performance is compared to the test data to determine its correctness.

4) Prediction

The user may enter a new game's name and score. The algorithms anticipate the review type for this new data point using their previously learnt parameters.

5) Application Flowchart

The following flowchart visualizes the system's workflow:



The system initiates with the start stage and moves on to step 1: Data collection, where the user is requested to enter the number of games. A loop is then run to collect information about each game, such as the name, score, and review type.

Step 2: Data Display and Preprocessing presents the collected data in a structured table, which is then translated into a numerical format known as DataPoint and further classified as BinnedDataPoint for usage in certain models.

Step 3: Model Training and Evaluation separates the dataset into training and test sets, with the latter being reserved for testing. The software then trains and tests three models: regression, Naïve Bayes, and a multi-layer perceptron.

In step 4: New Game Prediction, the user enters facts about a new game, which the trained MLP model uses to forecast the review type and likelihood. The procedure culminates with the End stage.

3. Data Structure

The data structure applied in the system uses three main classes to organize the data:

- **GameReview:** A simple class that stores the raw, user-entered data for each game, containing the **Name**, **ReviewScore**, and **ReviewType**.
- **DataPoint:** This class is used for the numerical models (Regression and MLP). It holds a single numerical feature (**Feature,** representing the score) and a binary numerical label (**Label**, where 1 is "Positive" and 0 is "Negative")
- **BinnedDataPoint:** This class is specific to the Naïve Bayes Classifier. It stores a categorical feature (**BinnedScore**) and a numerical label (**Label**). The scores are binned into "Low" (for < 5.0), "Medium (for > 8.0) and "High" (for >= 8.0).

4. Model Implementations

4.1 Simplified Linear Regression

This model is a minimal linear regression implementation designed for binary classification tasks. It uses the linear equation **y = mx + b** to learn the connection between an input score (**x**) and a projected output (**y**) where m is the weight and b is the bias. It is trained by gradient descent, an iterative optimization process. It constantly changes the weight and bias to reduce the difference between the anticipated and real labels. The weight update rules are as follows:

- **Weight Update:** $\boldsymbol{m_{new} = m_{old} + \propto (y_{true} - y_{pred})\, x}$
- **Bias Update:** $\boldsymbol{b_{new} = b_{old} + \propto (y_{true} - y_{pred})}$

Where $\propto$ is the learning rate, $\boldsymbol{y_{true}}$ is the label and $\boldsymbol{y_{pred}}$ is the model prediction. The system has a learning rate of 0.001 over 20,000 epochs.

The model's raw prediction is a continuous number. To convert this to a binary classification, a threshold is used. If the forecast exceeds or equals 0.5, the model predicts a positive review (1); otherwise, it predicts a negative review (0).

4.2 Naïve Bayes Classifier

The Naïve Bayes Classifier is a probabilistic model that relies on Bayes' Theorem. It works particularly well with text and categorical data, which is why numerical scores are first translated into bins. The model determines the likelihood of a review being positive or negative based on a specified binned score. This is founded on the following formula:

$$P(Class|Score) = \frac{P(Score|Class).P(Class)}{P(Score)}$$

The training method comprises computing the prior probabilities P(Positive) and P(Negative) for each class, as well as the conditional probabilities P(Score bin | Positive) and P(Score bin | Negative). To anticipate a new game, the model computes the likelihood that it belongs to each class and selects the class with the highest probability. The method also contains Laplace smoothing to handle scenarios where a feature bin is missing from the training data, which prevents zero probability.

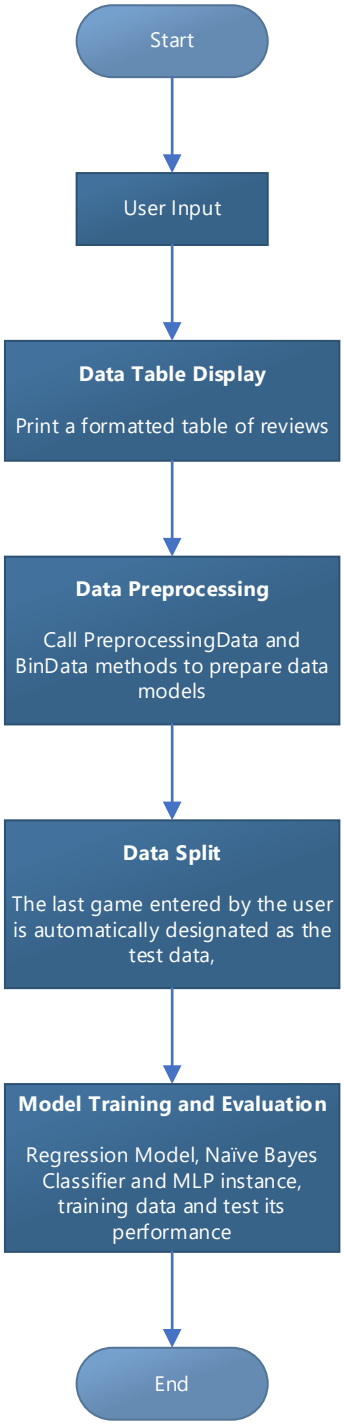
4.3 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron is a basic neural network with a single hidden layer. It can learn complicated non-linear correlations in data. The network is composed of an input layer (the game review score), a hidden layer with a size of 5, and a single output neuron. The sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ serves as the activation function for both hidden and output layers. This function reduces the result to a range of 0 to 1 making it appropriate for probability-based predictions.

Then, the backpropagation is used to train the model, which determines the error gradient in relation to the network's weights and biases. These gradients are then utilized to adjust the weights and biases, reducing the prediction error. The software trains the model for 20,000 epochs with a learning rate of 0.01. The network's ultimate output ranges between 0 and 1. A binary prediction is made using a 0.5 threshold, same like in the regression model.

5. Workflow and Logic

The Main method operates the entire system flow:



The Main method coordinates the whole system process by controlling each step sequentially. It starts with user input, inviting the user to provide information for a set of number of games, such as their name, score, and review type. Then, the system displays a data table with all submitted game reviews structured for clarity. Following that, data preparation is performed by invoking the PreprocessData and BinData methods, which prepare the information in numerical and categorical formats suitable for the models.

Then it performs a data split, with the latest upcoming game submitted by the user automatically labeled as test data and the remaining games as training data. Following that, the model training and evaluation phase begins, with the creation of a Regression model, Naïve Bayes Classifier and MLP instance, each trained on the training set and assessed on the test data, with accuracy shown.

Eventually, the system goes on to new game prediction, allowing the user to enter the new game's name and score before utilizing the trained models, mainly the MLP, to forecast the review type an likelihood, and displaying the findings. The workflow comes to a conclusion when the program is completed.

6. Limitations and Future Work

One significant disadvantage of the existing approach is its reliance on a manually supplied numerical score for prediction. The original user request was to predict a score or review type based just on the game name, which is not possible with existing algorithms. To predict a numerical score from a name, it needs a vast collection containing game names, genres, creators and historical review ratings and advanced machine learning techniques, such as Neural Language Processing (NLP) that are used to assess the game's name and other textual elements.

Future upgrades of the systems are that creating a more comprehensive dataset with new parameters like as genre, developer, and platform, implementing a regression model to predict a continuous score using these additional characteristics, and using more powerful NLP models to do sentiment analysis on text-based reviews rather than scores.

7. Conclusion

The future game review predictor effectively illustrates the implementation and deployment of three different machine learning categorization models. It is a good foundational example of how to create, train and assess machine learning models from the ground up, demonstrating the many techniques and capabilities in a real setting.