ECS171 Project Proposal - Group 3

Project Description

For this project, we would like to work on a System Implementation project since it enables us to take something abstract and convert it into a real, functioning system. By bringing together data scraping, machine learning algorithms, and deployment strategies into an end-to-end pipeline, we'll get practical experience in both data science and software engineering while providing a real-world application to forecast game popularity. For our project presentation, we will prepare visualizations of our results with project process and model explanations.
We will implement and compare 4 specific algorithms on our dataset: multiple linear regression(MLR), random forest, Long Short-Term Memory(LSTM), and Temporal Fusion Transformar(TFT). For multiple linear regression, we will use the variable such as game rank or avg viewer as the target value to predict game popularity, and select independent variables to learn relationships and evaluate performance. We will also implement a random forest classifier to categorize games into popularity groups ('Top 10', 'Top 50', 'Top 100', 'Top 200'), and use a random forest regressor within each category to predict the ranking. For LSTM, which is a type of recurrent neural network (RNN) that is designed to handle sequential data and capture long-term dependencies, we plan to use it for time-series forecasting to predict the popularity of games (measured by hours watched) for the next 12 months (i.e. capture information in time series, and predict future popularity trends with past patterns). For TFT, it is a combination of LSTM and Transformers, which handles long-range dependencies and captures temporal relationships.

Resources

For the software libraries, we will use scikit-learn for regressions in tradition ML, statsmodels for time series forecasting models like ARIMA, pytorch for training more complex deep learning models such as LSTMs for time series forecasting, numpy and pandas for data manipulation and preprocessing, and matplotlib and seaborn for visualization of historical trends and model predictions, with Tableau enhancing dashboard presentation. For other resources, GitHub will be the main platform for version control and collaborative development. Additionally, Twitch API may be utilized for real-time data collection.

For the dataset, we plan to use the Twitch streaming data from 2016 to 2023 from kaggle, ranking the top 200 games each month over a span of 12 months and 5 years. [https://www.kaggle.com/datasets/rankirsh/evolution-of-top-games-on-twitch] It includes key metrics such as watch time, stream time, peak viewers, number of streamers, and other relevant statistics.
Since we decided to implement LSTM and TFT, we'll need a GPU for training. Assuming we train on our existing RTX4060, LSTM with 50 epochs takes approximately 10 mins, and TFT with 100 epochs takes around 45 mins.

Value and Improvement Plan

Game popularity prediction is valuable across the gaming industry. Game developers can optimize release timing to maximize engagement, streaming platforms can refine recommendations to boost user retention, and streamers can anticipate trends to grow their audiences. By improving prediction accuracy, this project

helps stakeholders make data-driven decisions, enhancing both user experience and market success. To ensure our model fits into a diverse market, we aim to test its generalization across different datasets. To evaluate the accuracy of the model, we plan to test our model on different genres and time scales to examine its ability to capture universal trends, using performance metrics such as RMSE, MAE, and R^2 scores. Through the process, we can fix the potential overfitting issue and report on the possible defects of training predictive models with a single source.

Risk and Solution

This project itself contains some risks, such as the availability and quality of data, computational resource limitations, and expanding the project scope too far. The dataset may have some missing values or imbalanced cases, which we will handle by processing the data and adding more public datasets. Computational demands may be more than enough to exceed available resources, so we will turn to the GPU provided in Google Colab or UCD CSIF resources to optimize data processing. As a result, to prevent scope overload we will concentrate on LSTM-based forecasting and keep other models as secondary comparisons to have done inside the five-week timeframe. To incorporate the ability to interpret, we will use Matplotlib and Seaborn for data visualization. Finally, we run the risk of overfitting, especially with high dimensional data, and as such we will implement cross-validation, and dropout layer tuning to improve generalization.

Below is a workflow chart that represents our plan: