GameSphere: A Smart Gaming Analytics & Recommendation Application

Devdatta Gandole, Nakshatra Sachinbhai Desai, Rajarajeswari Premanand, Tejashree Kishor Badve, Vineeth Rayadurgam
San Jose State University
Course: Data 226 - Big Data Technologies and Applications

***Abstract*—GameSphere is a web-based platform that utilizes data to offer tailored video game suggestions and visual analytics for users on Steam. It employs PySpark, HDFS, and machine learning techniques to handle and examine an 8GB database of user reviews, facilitating actionable insights through sentiment analysis and collaborative filtering. The solution features an interactive frontend built with Streamlit to visualize trends and recommendations.**

I. Introduction

The rapid expansion of user-generated content on gaming platforms offers both advantages and challenges for developers and researchers aiming to extract valuable insights at scale. Steam, one of the premier digital distribution platforms for PC games, receives millions of user reviews and gameplay logs on a daily basis. The analysis of this high-volume, fast-moving data requires a Big Data strategy capable of efficiently ingesting, processing, and modeling both streaming and batch data.

In this study, we introduce **GameSphere**, a web-based gaming analytics and recommendation system that utilizes a scalable pipeline incorporating Hadoop Distributed File System (HDFS) for storage and Apache Spark for distributed computing. The comprehensive workflow is illustrated in a detailed flow diagram (refer to FlowDetails.docx), which outlines the stages of raw data ingestion, preprocessing, exploratory data analysis (EDA), sentiment extraction, collaborative filtering, and result visualization. The repository's organization—detailed in FolderStructure.rtf—ensures a clear distinction between raw and processed data, analysis notebooks, source code, and deployment scripts.

Our exploratory analysis phase ("EDA.ipynb") reveals significant patterns in review length, distribution of ratings, and temporal trends, which informs subsequent modeling choices. In "sentiment analysis.ipynb", we derive sentiment-aware features using natural language processing techniques that transform free-text reviews into polarity scores, enhancing the recommendation engine. The fundamental recommendation logic, implemented through alternating least squares in "game recommendation.ipynb", is compiled as a deployable Python module ("GameSphere.py"), along with a Streamlit frontend. All dependencies and environment settings are documented in "requirements.txt" to ensure reproducibility.

This paper presents three primary contributions:

- A modular, fault-tolerant Big Data pipeline for analyzing large-scale gaming reviews.
- Integration of sentiment analysis with collaborative filtering to enhance recommendation relevance.
- An interactive dashboard that allows end users to explore both aggregate trends and personalized recommendations in real time.

II. Dataset and Initial Preprocessing

Dataset: Steam Reviews 2021 from Kaggle

- **Size:** Approximately 8 GB
- **Entries:** Around 17 million reviews

Raw Schema Includes:

- Game identifiers: `app_id`, `app_name`
- User identifiers: `author_steamid`, `author_num_games_owned`, `author_playtime_forever`
- Review content: `review`, `recommended`, `votes_helpful`, `timestamp_created`, `language`

Preprocessing Pipeline:

1) **Directory Setup in HDFS:**
Structured directories such as `/data/raw`, `/data/processed`, `/models`, and `/outputs` were created to organize different stages of the data pipeline. This provided scalable and fault-tolerant storage for processing at various levels.

2) **Schema Standardization:**
Dot-separated column names (e.g., `author.steamid`) were converted to underscore format (e.g., `author_steamid`) to support SQL queries and PySpark compatibility. Entries missing essential fields were dropped to ensure data consistency.

3) **Text Normalization:**
Review text was cleaned by converting to lowercase, stripping HTML, emojis, and symbols, and removing entries with only numeric or junk content.

4) **Data Type Conversion:**
Fields were cast to types compatible with PySpark ML:
   - Integers: `app_id`, `author_num_games_owned`

- Floats: `author_playtime_forever`, `author_playtime_at_review`
- Boolean: `recommended`
- Timestamps: Converted from Unix format using `from_unixtime()` in Spark SQL

5) **Language Filtering:**
To ensure compatibility with the sentiment analysis model, only reviews where `language = ''english''` were selected. The filtered dataset was saved to `/processed/steam_review_english.parquet`.

Technology Role:
PySpark facilitated highly parallelized transformation using DataFrames, enabling millions of rows to be processed efficiently in memory. Its SQL support and functional chaining allowed seamless integration of schema transformations, filters, and column renaming steps, reducing ETL complexity and time to delivery.

III. Tools and Technologies

To build GameSphere, a combination of big data tools, machine learning frameworks, and frontend visualization technologies were used. Each component played a specific role in the ETL pipeline, modeling, and dashboard interaction.

- **HDFS:** Distributed file system used to store raw input files, intermediate processed datasets, model outputs, and final recommendations. It ensures scalable and fault-tolerant storage across nodes.
- **PySpark:** Used for data preprocessing, text cleaning, feature extraction, and model training. PySpark enables parallel computation over large datasets and integrates well with HDFS and MLlib.
- **Pandas, Seaborn, Matplotlib:** These libraries were used for exploratory data analysis and generating static visualizations during the review behavior analysis stage.
- **HuggingFace Transformers (DistilBERT):** A lightweight transformer model used to perform binary sentiment classification (positive/negative) on user reviews.
- **Spark MLlib (ALS):** Spark's machine learning library was used to implement collaborative filtering via the Alternating Least Squares (ALS) algorithm for generating personalized game recommendations.
- **Streamlit:** Python-based open-source framework used to build the interactive dashboard for users to input Steam IDs, view recommendations, and explore game trends visually.

IV. System Architecture and Workflow

GameSphere's system architecture was designed to support modular development and scalable analytics through a structured file layout and systematic data flow pipeline.

A. Directory and Data Layout

The underlying Hadoop Distributed File System (HDFS) directory structure plays a foundational role in organizing project stages. Each subfolder serves a unique function in the ETL and modeling process:

- `/data/raw`: Contains the original Steam reviews CSV file ingested into HDFS. This is the unprocessed, source dataset that serves as input to the pipeline.
- `/data/processed`: Holds datasets cleaned and transformed using PySpark, including filtered English reviews and enriched sentiment scores. These Parquet files serve as training data for the recommendation system.
- `/data/mappings`: Stores auxiliary mappings (e.g., game ID to name, user ID aliasing) required for joining datasets during analysis and recommendation.
- `/models`: Persists trained machine learning models for both collaborative filtering and sentiment classification. These models can be reloaded for inference or retraining.
- `/outputs`: Final output directory that contains recommendation results and aggregated sentiment scores for use by the frontend dashboard.

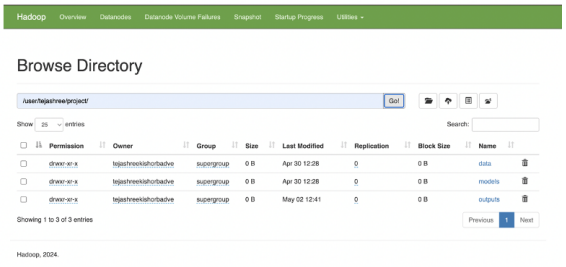

Fig. 1. Root directory structure showing data, models, and outputs folders

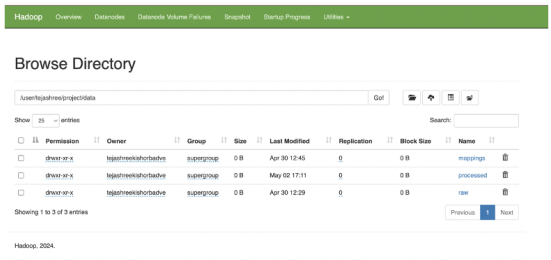

Fig. 2. Breakdown of data folder into raw, processed, and mappings

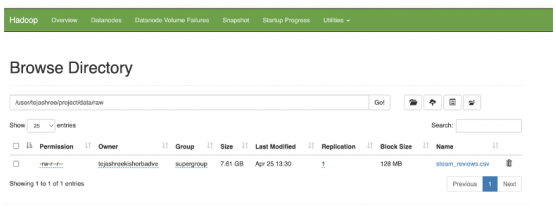

Fig. 3. Raw data folder showing the source Steam reviews CSV file

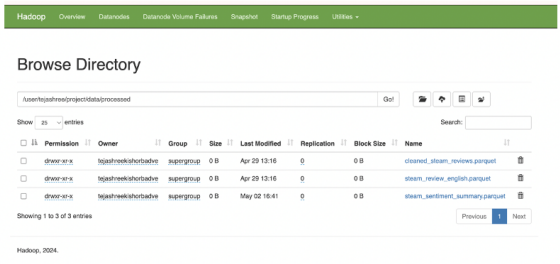

Fig. 4. Processed data folder with cleaned and filtered Parquet files

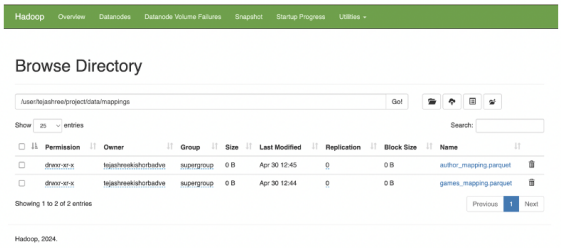

Fig. 5. Mappings folder with author and game ID reference files

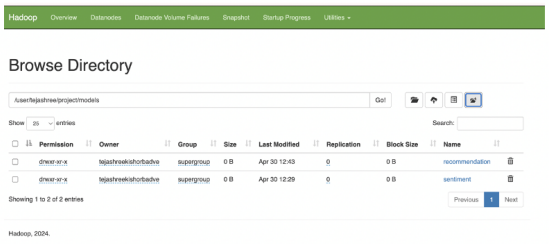

Fig. 6. Models directory storing trained ALS and sentiment models

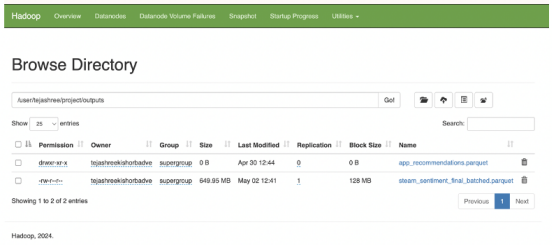

Fig. 7. Outputs directory containing recommendation results and sentiment batches

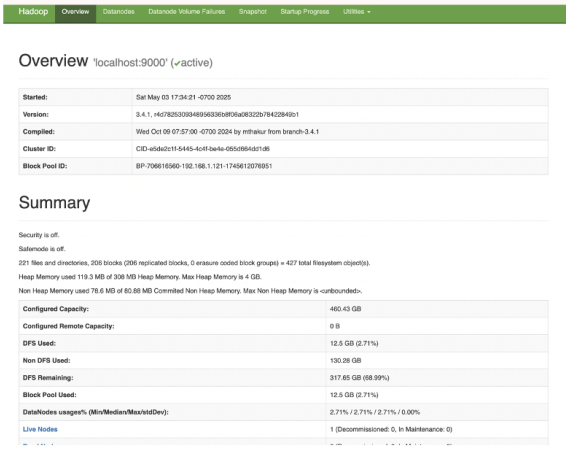

Fig. 8. Hadoop UI view showing DFS usage and cluster health metrics

B. Workflow Architecture and Data Flow

The GameSphere data pipeline follows a structured, multi-stage architecture that transforms raw reviews into actionable recommendations and visual insights. Each stage feeds into the next, forming a modular and fault-tolerant processing framework.

- **Data Ingestion:** The pipeline begins by importing the Steam Reviews CSV into HDFS under the `raw` directory. This centralized storage supports scalable processing.
- **Initial Cleaning:** Spark jobs remove duplicates, drop missing values, and standardize data types in preparation for modeling.
- **Advanced Cleaning:** PySpark handles deep normalization steps, including review text cleaning and converting nested fields into flat schema.
- **Feature Engineering:** Cleaned data is transformed into a user-game interaction matrix. Game and author IDs are mapped to unique indices for model compatibility.
- **Recommendation Modeling:** ALS-based collaborative filtering is trained to learn latent preferences. Generated predictions are saved as user-specific top-N lists.
- **Model and Output Persistence:** Trained ALS models and generated outputs (recommendations, sentiments) are written to the `models` and `outputs` folders.
- **Frontend Integration:** Streamlit loads outputs from HDFS and dynamically renders visualizations, allowing users to interactively explore game trends and personalized suggestions.

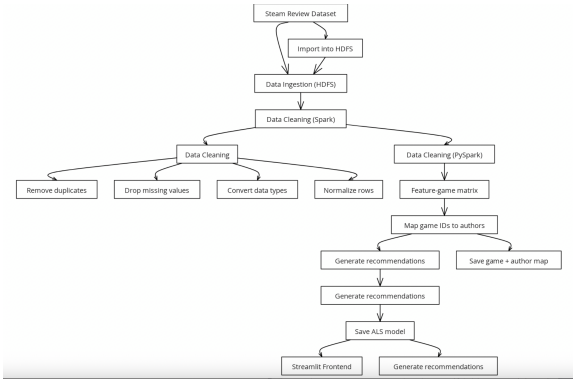

Fig. 9. Expanded pipeline showing detailed data cleaning, mapping, and ALS generation process

This architecture ensures efficient data reuse, reproducibility, and scalability. Each module performs a clearly defined role in the transition from raw data to intelligent, user-facing recommendations.

V. Methodology

The GameSphere initiative employed a systematic approach from conception to implementation:

A. Problem Definition

Identified the issue of deriving valuable insights from millions of user-generated reviews on Steam. Objectives were set to offer game suggestions, analyze sentiment, and visualize user interactions.

B. Data Collection

The Steam Reviews 2021 dataset, containing approximately 17 million entries, was obtained from Kaggle. The dataset was uploaded and organized within HDFS to allow for scalable and concurrent data access.

C. Data Cleaning and Preprocessing

The schema was standardized, and irrelevant or noisy data entries were removed. Text content in reviews was normalized and refined for NLP compatibility. Only English reviews were retained, and the cleaned dataset was saved in Parquet format for efficient downstream use.

D. Exploratory Data Analysis (EDA)

User behavior, review trends, and language distribution were visualized. Data biases such as skewed playtime and review frequency were identified and corrected through adjusted filters and aggregations.

E. Sentiment Modeling

The [**?**] model from HuggingFace was employed to classify reviews as positive or negative. The resulting sentiment scores were merged back into the original dataset to enrich downstream recommendation modeling.

F. Recommendation System

Collaborative filtering was implemented using the Alternating Least Squares (ALS) algorithm on the filtered review ratings. Hyperparameters such as rank and regularization were tuned, and model accuracy was evaluated using RMSE on a test split.

G. Dashboard Development

An interactive frontend was created using Streamlit. The interface accepts Steam IDs to generate personalized game recommendations and enables users to explore sentiment and popularity trends.

H. Testing and Results

The recommendations were cross-verified with popular game titles. Sentiment trends were analyzed in correlation with user playtime and review activity levels to ensure consistency and relevance.

I. Finalization

Models and outputs were stored in HDFS for accessibility. All backend processes were linked to the Streamlit dashboard to enable real-time insight delivery and a seamless user experience.

This organized methodology ensured that every stage—from raw data ingestion to insight visualization—was based on reproducible, scalable practices that leveraged contemporary big data and machine learning technologies.

VI. Exploratory Data Analysis (EDA)

The Exploratory Data Analysis (EDA) stage lays the groundwork for all future machine learning models and the development of dashboards. This phase includes statistical analysis, visual pattern identification, and data quality checks to uncover trends and anomalies.

A. Game Trends

The top 20 most reviewed games were identified to understand popularity patterns. A pie chart analysis revealed that titles like *Terraria*, *Tom Clancy's Rainbow Six Siege*, and *Garry's Mod* dominate user discussions.

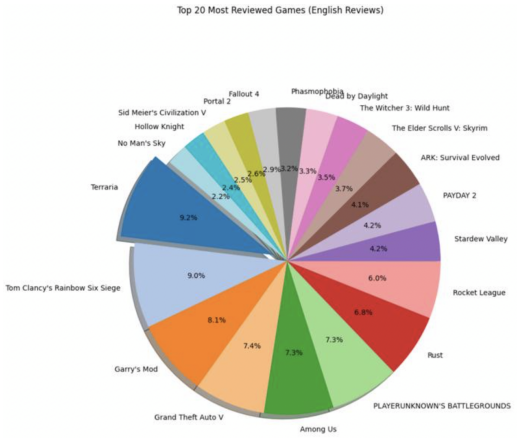

Fig. 10. Top 20 most reviewed games

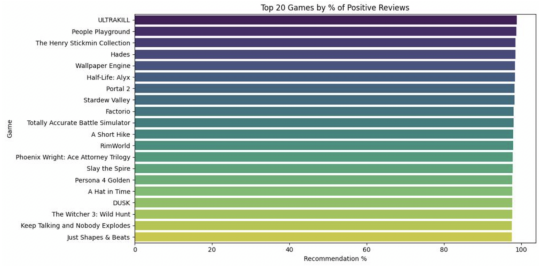

Fig. 11. Top 20 Games by % of Positive Reviews

Further, the recommendation percentages for each game were examined. Games such as *ULTRAKILL*, *People Playground*, and *The Henry Stickmin Collection* were among the highest-rated by sentiment. These insights guided the recommendation engine by establishing a reference set of high-performing titles.

B. Temporal Patterns

Monthly review patterns showed a spike in user reviews around global game sales and releases. This temporal trend also corresponded with peaks in sentiment activity, suggesting that promotional periods influence not only volume but also the tone of user feedback.

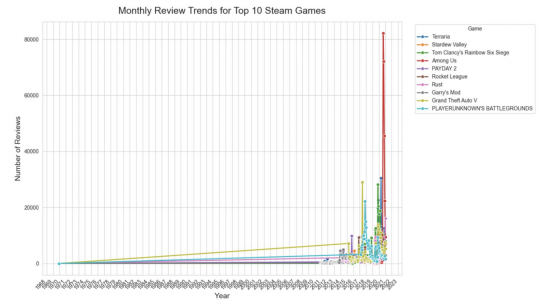

Fig. 12. Monthly Review Trends for Top 10 Steam Games

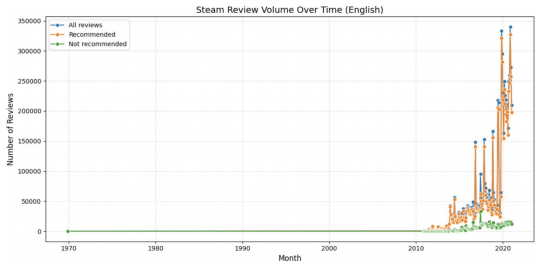

Fig. 13. Steam Review Volume over time

C. User Analysis

User behaviors were explored via histograms and correlation plots. The relationship between playtime and sentiment was clear: users who invested more time typically gave more favorable reviews.

A focused case study on *PLAYERUNKNOWN'S BATTLEGROUNDS* reinforced this trend—most users had limited playtime, and recommendations were nearly balanced. This helped inform filtering thresholds in modeling.

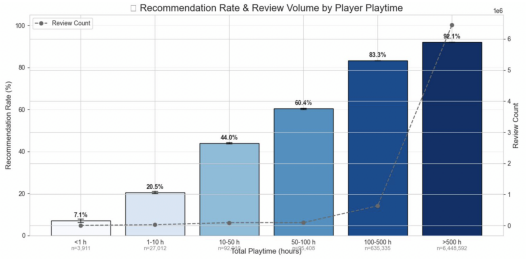

Fig. 14. Recommendation Rate & Review Volume by Player Playtime

D. Engagement Metrics

Helpful reviews and longer texts often came from users with strong sentiment (either positive or negative). The relationship between review count and recommendation ratio identified standout games like *Among Us*, *Rust*, and *Rainbow Six Siege* as highly engaged titles.

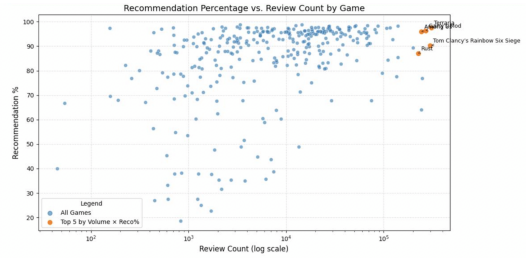

Fig. 15. Recommendation Percentage vs Review Count by game

E. Sentiment Word Patterns

Text analysis revealed that positive reviews frequently included words like "fun," "great," and "love," while negative reviews often mentioned "money," "cant," and "bad." These patterns validated the classifier's sentiment labels and were considered when designing interpretability tools for the dashboard.

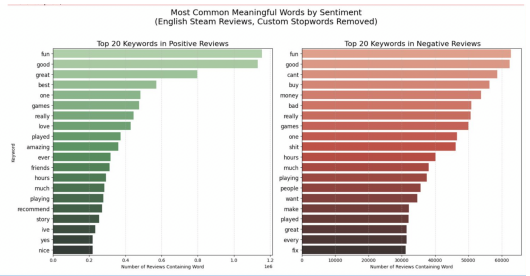

Fig. 16. Most Common Meaningful words by Sentiment

F. Hidden Gems

To uncover lesser-known yet highly rated games, reviews were filtered for titles with over 95% positive sentiment but under 10,000 total reviews. This highlighted indie successes

such as *ULTRAKILL*, *DUSK*, and *Townscaper*, providing a valuable lens on niche community favorites.

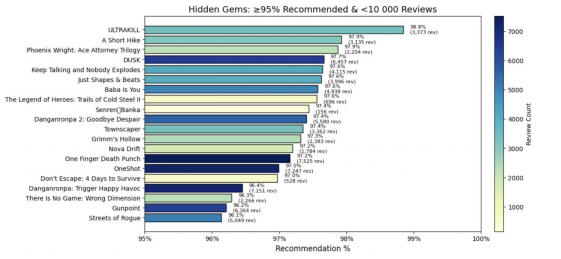

Fig. 17. Hidden Gems

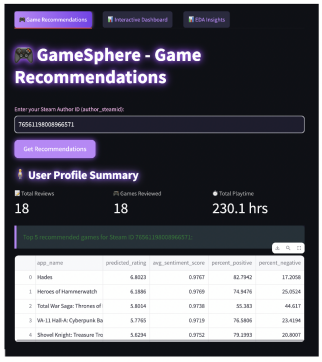

Fig. 18. Sentiment Analysis of Game Recommendations

VII. Sentiment Analysis Pipeline

The sentiment analysis stage enriches user-generated reviews with emotional context, enabling a more personalized and insightful recommendation experience. This phase classifies reviews as **POSITIVE** or **NEGATIVE** using a pretrained transformer model and calculates confidence scores that indicate the model's certainty.

Model Used

```
distilbert-base-uncased-finetuned-sst-2-english
```

1. Step-by-Step Process

1) **Loading Preprocessed Data:**
English-language reviews, pre-filtered and stored as Parquet files, are loaded using PySpark DataFrames for distributed processing.

2) **Review Classification:**
Each review is passed through the HuggingFace DistilBERT pipeline, which tokenizes the text and returns a sentiment label (POSITIVE or NEGATIVE) along with a confidence score. This stage is optimized for batch processing and supports GPU acceleration.

3) **Schema of the Output:**
The enriched DataFrame includes:
   - `review_id`
   - `app_id`
   - `app_name`
   - `sentiment_label`
   - `sentiment_score`

4) **Game-Level Aggregation:**
Using PySpark's `groupBy()` function, reviews are aggregated by `app_id` to calculate:
   - Percentage of positive reviews
   - Average sentiment score per game

5) **Merging and Storing Output:**
Aggregated sentiment scores are joined with the original cleaned dataset. The final output is written to HDFS:

```
/processed/cleaned_steam_reviews_merged.parquet
```

2. Contribution of Technology

- **HuggingFace DistilBERT:** A compact, fine-tuned transformer model optimized for sentiment classification. It balances high accuracy with low latency, suitable for large-scale textual analysis.
- **Spark DataFrames:** Enabled distributed and parallelized review processing. Aggregation and joins on massive datasets were handled efficiently using PySpark APIs.

This sentiment analysis pipeline bridges raw text with structured insights, improving the interpretability and personalization of game recommendations.

VIII. Game Recommendation System

The game recommendation system employs collaborative filtering with the Alternating Least Squares (ALS) technique. The comprehensive process builds on each previous step to yield high-quality, individualized recommendations at scale.

1. Data Preparation

- The `recommended` column is transformed into a numeric rating: 5 signifies a positive recommendation, while 1 indicates otherwise. This standardization is essential for the ALS algorithm's input.
- Games that have received fewer than 200 reviews are eliminated to minimize noise and guarantee that the model learns from data that holds statistical significance.
- Users with abnormally low or high playtimes are excluded to filter out anomalies and potential bot activity, resulting in a more representative user base.

2. Feature Engineering

- User and game identifiers (e.g., `author_steamid`, `app_id`) are converted into numerical indices using `StringIndexer`. These indices are necessary for the ALS algorithm's matrix operations.

3. Training ALS

- The ALS model is set up utilizing the indexed fields: `userCol=author_index`, `itemCol=app_index`, and `ratingCol=rating`.

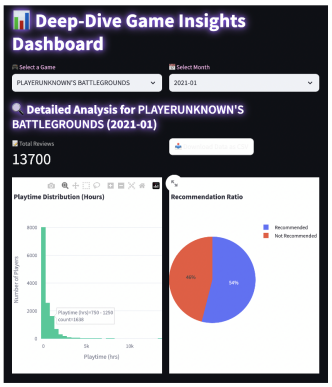

Fig. 20. Game Insights

- Hyperparameters such as `rank` (number of latent features), `regParam` (regularization), and `maxIter` (number of iterations) are fine-tuned to enhance prediction accuracy.
- The dataset is divided into training (80%) and testing (20%) portions. This division ensures that the model is evaluated using data it has not previously encountered.

4. Evaluation

- The model's performance is measured using Root Mean Squared Error (RMSE). A lower RMSE reflects superior predictive ability.
- This evaluation is instrumental in detecting overfitting and verifying the selected hyperparameters.

5. Output

- Following the training phase, the model produces the top 5 game recommendations for every user.
- These recommendations are stored at the following HDFS location for future use:

  `/outputs/user_recommendations.parquet`

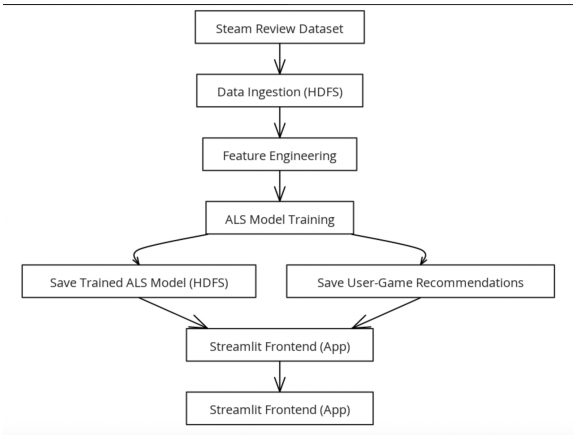

Fig. 19. High-level ALS model training and deployment flow integrated with frontend

6. Technological Advantage

- The ALS implementation in Spark MLlib takes advantage of distributed computing, enabling scalable matrix factorization across multiple nodes.
- This arrangement allows for efficient model training involving millions of users and games, greatly cutting down on computational time and memory requirements.

Each phase of this pipeline builds upon the previous one, refining data and optimizing model inputs to create a quick, precise, and scalable recommendation system that drives the GameSphere dashboard. This setup efficiently manages millions of user-game interactions, significantly speeding up both training and prediction processes across extensive clusters.

IX. Streamlit Dashboard

Streamlit is a free, open-source Python library that facilitates the development of interactive web applications for data science initiatives with minimal coding effort. It empowers developers and analysts to convert data scripts into shareable web applications in just a few minutes by solely using Python.

Streamlit was selected for this initiative due to its user-friendliness, rapid development process, and seamless integration with Python-driven data workflows. Its reactive structure automatically refreshes the frontend in response to changes in the backend, making it ideal for real-time dashboards.

In **GameSphere**, Streamlit serves as the user interface that connects individuals to insights derived from extensive data analysis. It interfaces with cleaned datasets and model outputs to deliver a smooth experience through an easy-to-navigate layout.

Technical Implementation

- Streamlit scripts were written in Python and deployed either locally or via web hosting.
- Data was accessed from Parquet files generated by the PySpark processing pipeline.
- Caching techniques such as `@st.cache_data` were used to avoid unnecessary reloading of datasets.
- Charts were built using Streamlit's built-in tools and external libraries like Plotly and Matplotlib.
- The layout was structured with columns and tabs to separate visual analytics from personalized recommendations.

Tabs in the Dashboard

- **Game Recommendations**
  - **Input:** Steam ID
  - **Output:** Top 5 recommended games, predicted ratings, sentiment score overlays
- **Analytics Dashboard**
  - **Filters:** Game selection and date range (monthly)
  - **Visuals:** Pie charts, bar graphs, and time-series trendlines for reviews, sentiments, and recommendation frequency

Integration

- Accesses cleaned data from Parquet files
- Employs caching to enhance performance and minimize redundant I/O operations

Streamlit's ease of use enabled quick prototyping and iterative enhancement of user experience while maintaining full compatibility with the backend machine learning workflows.

X. Conclusion and Future Scope

GameSphere showcases the effective integration of big data processing, sentiment analysis, and collaborative filtering to generate personalized, insightful gaming recommendations. The platform processes millions of user reviews using PySpark and HDFS, enriches them with sentiment via DistilBERT, and delivers real-time visualizations through Streamlit.

The structured pipeline—from raw data ingestion to frontend deployment—demonstrates a scalable, modular design that supports both analytical and predictive use cases.

Future Enhancements

- **Multilingual Support:** Expand sentiment analysis to support reviews in languages beyond English.
- **User-Level Insights:** Enable tracking of personalized sentiment and behavioral patterns.
- **Real-Time Processing:** Integrate Kafka and Spark Streaming for live review ingestion and dashboard updates.
- **Game Metadata Fusion:** Incorporate game genre, release dates, and pricing for more context-aware recommendations.
- **Mobile Optimization:** Adapt dashboard UI for mobile platforms to enhance accessibility.

Overall, GameSphere illustrates the powerful synergy of data engineering and machine learning in transforming large-scale unstructured data into actionable insights for the gaming ecosystem.

XI. Conclusion

GameSphere successfully integrates big data analytics, machine learning, and web technologies to deliver an innovative platform for gaming insights and personalized recommendations, demonstrating practical use of modern big data technologies.

References

- **Kaggle:** Steam Reviews 2021
- **HuggingFace Transformers**
- **Apache Spark MLlib**
- **Streamlit Documentation**
- **GitHub Repo:** https://github.com/tejashreebadve/DATA228_Group11