2ND EDITION

databricks

ULTIMATE GUIDE TO GAME DATA AND AI

DATA TEAMS EDITION

SAFTLER | DAVIS | BUCKLEY

4-1-20-1-2-18-9-3-11-19

ULTIMATE GUIDE TO GAME DATA AND AI – DATA TEAMS 2ND EDITION

Authors: Bryan Saftler, Duncan Davis, Huntting Buckley

TABLE OF CONTENTS

Foreword

We proudly present this year's Guide to Game Data and AI, the flagship ebook of Databrick's gaming industry. Gaming continues to be a leader across industries in cutting-edge data and AI use cases, driving innovation and entertainment for billions of players around the globe.

Since our first ebook on game data and AI, we've received incredible feedback on the need for gaming studios to adopt a more data-driven approach to game development and operations. According to Newzoo gaming industry research, more than 3.4 billion players will drive 187.7 billion dollars in revenue generated from game content alone this year. Over the past decade, data has played a critical role — and increasingly artificial intelligence — in the development, launch, and operation of video games.

In this year's guide, we will explore a new data architecture category entering the game, the Data Intelligence Platform. We'll discuss what it is, how it can form the foundation of your analytics practice, and the core gaming use cases unlocked as a result. We'll then dig into Three Fundamental Things to Do and Know in 2024, including how generative AI is being used (responsibly) in game development, the need for democratizing insights, and the critical role streaming data plays in driving player engagement. Finally, we'll jump into interesting game data and AI use cases that surfaced this past year across pre-production, shipping a game from beta to global launch, and live operations workloads.

At the end of the guide, we cover a series of great 101 topic areas including data and AI fundamentals, materials to get started, and additional use cases relevant to gaming teams. We want to thank you for taking the time to read this guide, and if you find this interesting, we hope you'll share it with a colleague.

The Data Intelligence Platform for Gaming

The video game industry is a hits-driven business. Building and operating a game without a data strategy is akin to throwing darts blindfolded. You may hit the target occasionally, but you'll have no frame of reference for how or why. Furthermore, you won't know where to aim as changes in the environment occur. When it comes to game data and AI, there is a maturity curve that helps to illustrate how game teams can (at times) struggle to move up the curve and take advantage of more valuable data and AI use cases. (see **Figure 1**)

The data and AI maturity curve

On the left of the curve, basic reporting is displayed in the form of business intelligence and data warehousing, helping game teams unlock ***what's already happened*** in their title(s). As you move to the right on the curve, data and AI use cases begin to add a layer of competitive advantage for your studio. You move from understanding ***what's already happened***, to ***understanding what's happening in real time***, to being able to more accurately ***predict what will happen in the future***. This progression is incredibly important for acquiring, engaging, and ultimately retaining players over time.

Why game teams struggle with data and AI

The gaming industry is not short of data, from engine telemetry to game services, marketplace data, and sources beyond the game (such as social media). It's no wonder many game teams struggle to take full advantage of all the data swarming around their games. Whether it's fragmented views of their playerbase, legacy data warehouse solutions that are unable to scale, challenges moving from descriptive (what has happened) to predictive insights (what will happen), or a general lack of real-time capabilities — game studios feel this

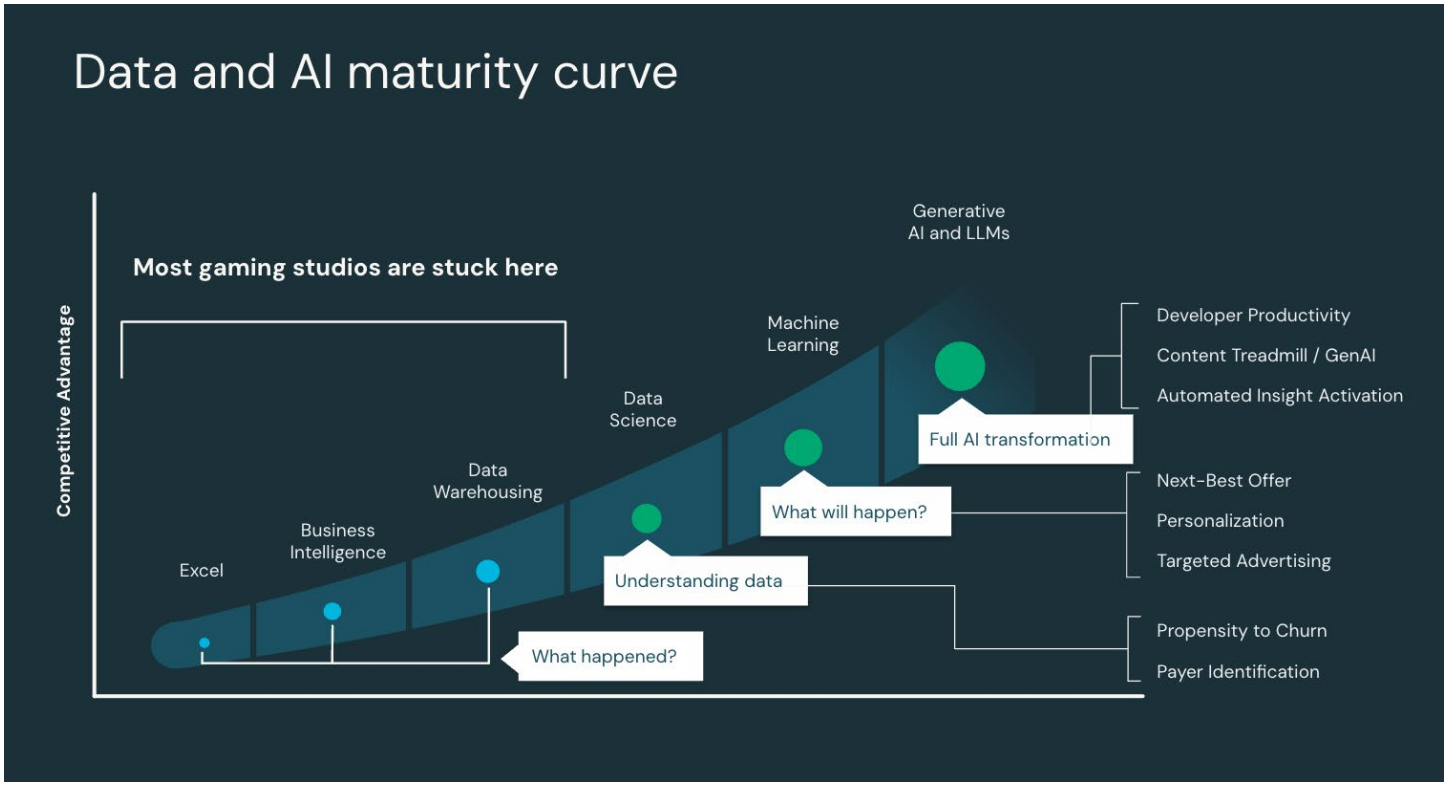
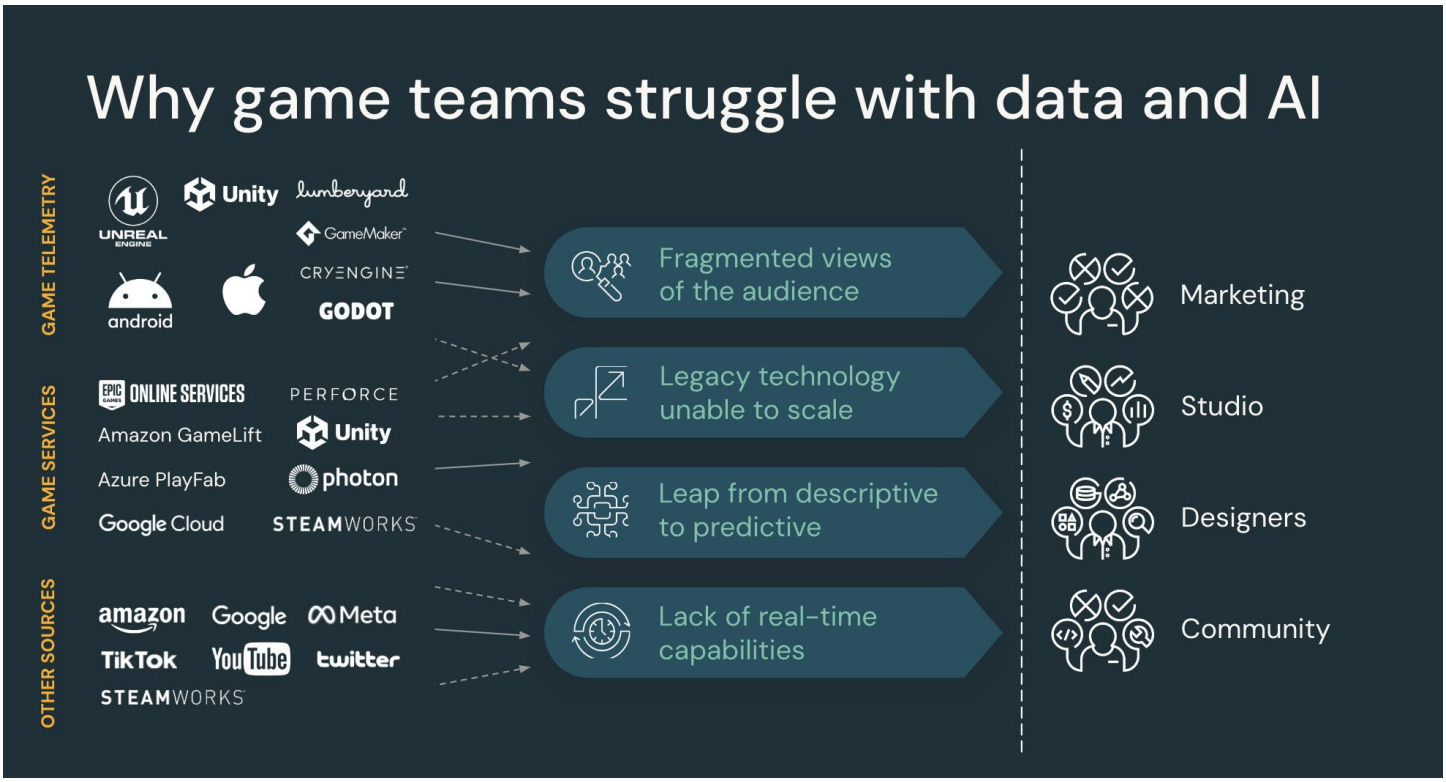


FIGURE 1



pain more acutely than most industries. Additionally, teams often struggle when it comes to collaborating and sharing data internally with key stakeholders, be that marketing, community managers, designers, finance and more. Combine these realities, and even some of the most experienced game teams are behind the curve when it comes to data engineering and data science.

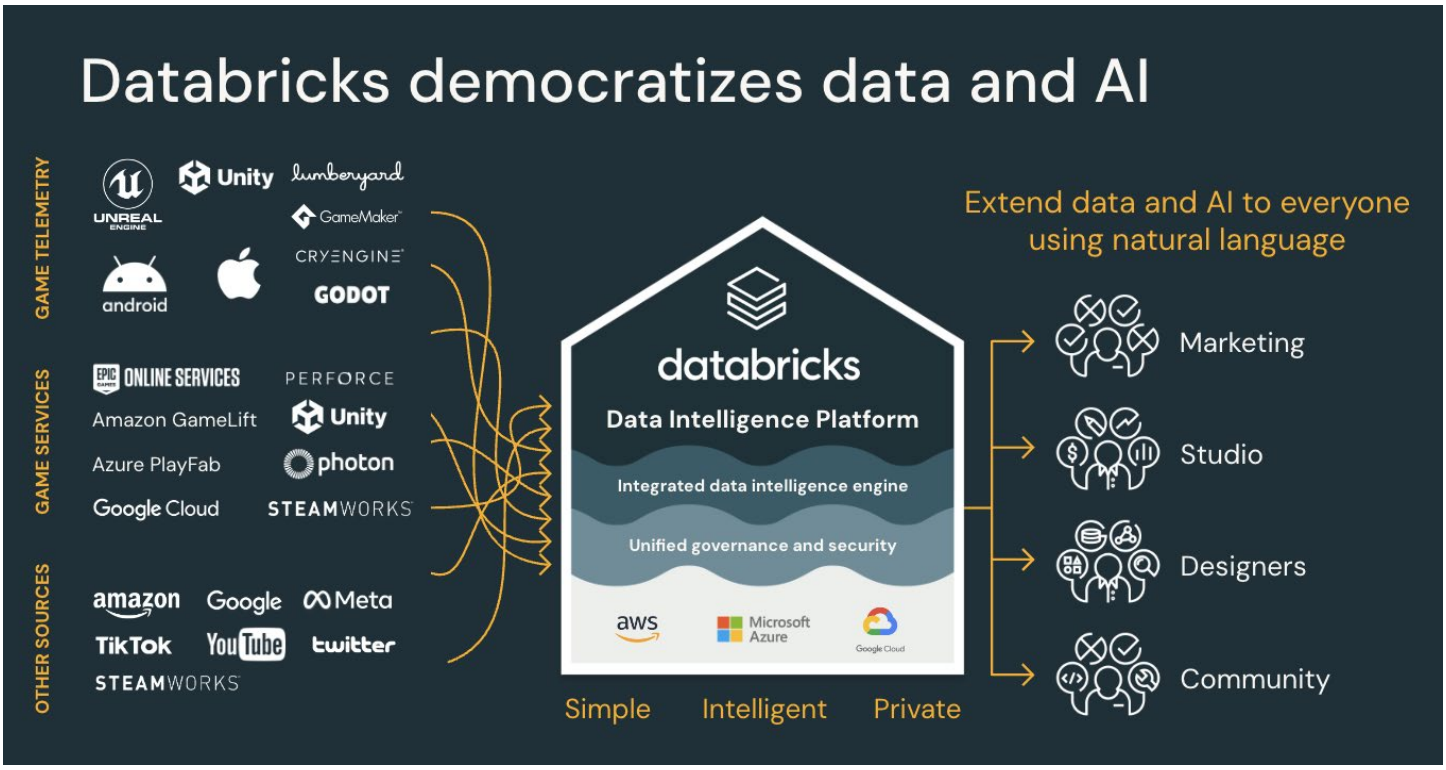
It takes incredible human effort to build a great game and foster a healthy community. Game teams that are doing this at scale and finding repeated success **know how to use data and AI to their advantage**. It starts with a unified data platform that democratizes end-user access, while simultaneously simplifying the management, security, and governance of that data.

Why is this important? When your data, AI, and governance is siloed, access to that data is restricted, costs rise, and time-to-value slows way down. This is especially true when each silo requires a different approach to security and governance. What about using AI? Without a unified approach, implementing AI opens the door to new data privacy and control questions. Finally, technical teams are a premium in game studios, and when you're reliant on your technical team to create data products, you artificially bottleneck the team members who might benefit from insights.

> Data builds better games, and fosters healthier communities.

Introducing the Data Intelligence Platform

If you've been around game data, you may be aware of the term data lakehouse — a unified data platform that brings together all of your game data and AI to unlock critical use cases across player experience, revenue generation, content pipeline, and more. It does this by combining data warehouse capabilities with the low cost storage of a data lake, thus reducing costs while greatly simplifying your data stack. In 2023, Databricks announced the next step in our data and AI journey, the Data Intelligence Platform, a data platform that integrates generative AI capabilities into our lakehouse architecture, giving you one architecture for all your data and AI workloads.



Databricks built the Data Intelligence Platform to help teams better succeed when using AI and democratizing insights, while driving down costs. Game teams can develop generative AI applications on their data without sacrificing data privacy or confidential IP. More people across the organization can discover insights from more game data in the language of their business. In addition, teams can gain efficiency and simplify complexity by unifying data, governance, and AI together. This is only possible on a Data Intelligence Platform.

Three Fundamental Things to Do and Know in 2024

Game data, and increasingly artificial intelligence (AI), have become central to the development and operation of today's top-grossing video games. In 2024, we highlight three fundamental trends that game developers should focus on to harness the full potential of their game data and AI. These include the responsible use of generative AI, the democratization of insights across game teams, and the critical role of streaming data in driving real-time actionable insights.

The responsible use of generative AI

The rapid advancement of generative AI has sent waves throughout the video game industry, from the role it plays in game development, testing, and gameplay experiences, to community management and moderation. As gaming companies increasingly explore the capabilities of generative AI, it is essential for developers to understand and embrace the responsible use of this technology. Job displacement

GenAI is not just about code or asset development, it's also about augmenting traditional machine learning for better insights.

is real and it is happening. As developers and industry advocates, we need to guide our use of generative AI as a tool to augment human creativity and decision-making, not replace it.

The responsible integration of generative AI into the video game industry requires a thoughtful approach that prioritizes human oversight, guidance, and iteration. By understanding the potential and limitations of generative AI, game developers can harness this technology to enhance creativity and innovation while mitigating potential risks and upholding ethical standards.

When generative AI is used to augment traditional machine learning, interesting scenarios are surfaced, such as story generation, immersive non-playable characters, and interactive narratives that are tailored to individual players.

Here are a few real-world examples of generative AI where Databricks is helping teams explore, develop, and deploy:

- **Interactive Creative Boards:** Helping teams to visualize and iterate on game elements trained from first-party art and data assets, reducing IP and copyright risks.
- **Personalized Lore and Discoveries:** A game's lore and discoveries personalized to reflect each player's unique experiences and actions within the game world. This also can be applied to character dialogue and storyline creation, where generative AI can be integrated into the game itself, allowing for real-time content generation — within reason — based on individual player actions.

- **Character AMAs:** Trained on diverse datasets, including gameplay videos, dialog, asset libraries, player feedback, and gaming narratives, generative AI powered chatbots have incredible potential for community engagement.

Additionally, voice synthesis and text-to-speech capabilities can be integrated to further enhance the chatbot's ability to interact with players.

Do This in 2024

Leveraging Generative AI for Enhanced Player Segmentation in Gaming

Overview

Generative AI significantly enhances player segmentation by handling vast datasets beyond human interpretive capacity. It reduces manual analysis and biases prevalent in machine learning techniques, like K-Means Clustering. GenAI streamlines the segmentation process, increases accuracy, and accelerates updates, while also enabling natural language querying for broader business accessibility.

Detail

Player segmentation involves analyzing extensive, clean data to identify key features and understand past performance for model training. Traditional methods require laborious data interpretation to determine cluster uniqueness and utility, often leading to human error and cognitive biases. A human can only process so many groups, limiting your number of segments to 3-6 is often not enough to identify truly unique clusters in your playerbase. GenAI mitigates many of these issues by automating cluster identification and feature analysis, allowing for more frequent and accurate segment revisions in response to changing player behaviors.

Outcome

Adopting GenAI for player segmentation optimizes personalization and accelerates time-to-value by minimizing manual labor and human error. It enables more frequent reassessment of segments, resulting in better-aligned, valid marketing strategies and improved product engagement insights. Furthermore, GenAI democratizes data insights, empowering all business units to generate insights independently, fostering experimentation and enhancing trust in data-driven decisions.

Implementation Considerations

In this use case, it is essential to start with comprehensive data preparation. This involves ensuring the availability of extensive and clean player data — including relevant features and past performance metrics — to effectively train the generative AI model. It's important to take into account factors such as data volume, the complexity of segmentation, and the specific gaming use case. Integration with existing systems is also crucial. Generative AI can be integrated with current machine learning and analytics systems to leverage its capabilities for player segmentation without disrupting workflows. You'll want to assess the scalability and performance of the model to handle large volumes of player data and provide real-time or near-real-time segmentation results. Otherwise, it's not very helpful. Finally, ethical and regulatory compliance should be a key focus. Ensure that the use of your model for player segmentation complies with ethical guidelines, data privacy regulations and industry best practices to maintain trust and transparency with players.

Best Practices

The most important considerations in GenAI player segmentation are continuous model evaluation and prioritizing the interpretability and explainability of your AI-driven segmentation results. This practice ensures that the AI model remains accurate and relevant, mitigates potential biases and empowers stakeholders across your game studio with actionable insights, ultimately contributing to improved player segmentation outcomes.

Democratizing insights across game teams

Increasingly, the democratization of game data across the studio will be a strategic differentiator, separating the top grossing games from the rest. With advances in AI-powered analytics, teams can provide invaluable near-real time insights into player behavior, demographics and engagement patterns that enable data-driven decisions during the development process. These insights can also help drive deeper player engagement through automated action and human intervention during live operations.

However, making insights accessible across various teams, including design, development, and marketing, is not as straightforward as it should be. Game studios need to consider how to foster a collaborative environment where decisions are informed by a comprehensive understanding of player dynamics and preferences.

Data sharing is a core component of this. In 2021, Databricks started the open source Delta Sharing project, the industry's first open protocol for secure data sharing. This includes any data that is read and processed across organizations. Teams can share and consume data from any platform or vendor that supports the open Delta Sharing protocol. By lowering barriers for sharing data, teams can more quickly accelerate innovation, increase collaboration and lower costs.

Here are a few real-world examples of democratizing insights where Databricks is helping teams explore, develop and deploy:

- **Accessibility of Analytics:** Making AI-powered analytics accessible across various teams from product, to operations, revenue and design, enables informed decisions in areas such as game mechanics, level design and player experiences.
- **Collaborative Decision-Making:** Democratizing insights empowers teams to optimize game elements based on a comprehensive understanding of player dynamics and preferences and not preconceived beliefs or stereotypes.
- **Personalized Player Experiences:** Generative AI can be used to create personalized player, and immersive, captivating experiences, enhancing player engagement and satisfaction.

Do This in 2024

Democratizing Game Development Insights with Social Media Analytics

Overview

Social media analytics is crucial for understanding community feedback, needs and interests. Initially, teams may manually review player comments, but this approach is not scalable and is prone to confirmation bias. Automated analytics can process large volumes of data, providing a more comprehensive and unbiased understanding of player sentiment.

Detail

Effective social media analytics combines traditional machine learning (ML) models with large language models (LLMs) to analyze player discussions across various platforms, such as Steam Reviews, Reddit, forums and support tickets. This approach helps identify prevalent topics and sentiments within the gaming community. Leveraging LLMs allows the data to help determine the right labels and terms to monitor.

Outcome

By integrating ML and LLMs, game developers can detect sentiment trends and community hotspots, enabling product and operations teams to address in-game issues and designers to refine future development cycles. This proactive use of social media analytics can lead to quicker issue resolution, risk mitigation and deeper community engagement.

Implementation considerations

Start with game reviews from platforms like Steam, EPIC Games, Xbox, and PlayStation. Preserve raw data for incremental insight curation and maintain time series data to correlate findings with game updates. Be prepared to adapt your analysis as new questions arise from your team.

Best practices

Avoid confirmation bias by letting data guide decisions, and cross-reference social media feedback with in-game behavior. Focus on the broader player base rather than a vocal minority, and validate language-based insights with actual gameplay data to ensure accuracy.

Maximizing the value of streaming game data

It is natural to be overwhelmed by the volume of data with which game teams must contend. What should we focus on? What metrics should we be tracking? How can we better inform our game development? When it comes to data, streaming data is critical and often under-utilized. With the improvement of AI-driven analytics tools, teams are now capable of processing and analyzing streaming data, up-to-the-moment player interaction, in-game behavior and performance metrics.

This near-real-time feedback loop enables teams to make informed adjustments and refinements to the game environment, ensuring the player experience remains dynamic and responsive to player and community actions. As the model improves, data teams can implement algorithmic decision-making and further speed the impact this insight can have on the player experience. Streaming data is the key game teams require to unlock better engagement and retention of players over time.

Here are a few real-world examples of streaming data where Databricks is helping teams explore, develop and deploy:

- **Real-time feedback loop:** Streaming data enables real-time feedback on player interactions, in-game behaviors and performance metrics, allowing developers to make informed adjustments and refinements to the game environment.
- **Dynamic gaming experiences:** By leveraging streaming data, game developers can ensure that the gaming experience remains dynamic and responsive to player actions, leading to more engaging and immersive games.
- **Optimizing game performance:** AI-driven analytics tools can automatically adjust game settings to ensure maximum performance across multiple devices and platforms.
- **Re-engagement and next best offer:** These are two of the most critical streaming data use cases for any live game. If you predict someone will churn, but it takes 24 hours for your batch process to kick in or even longer for a human to review it, you've lost that player and their related UA investment. Similarly providing the right offer, at the right time, based on the most recent status of the player will improve your yield.

- **Automated experience tuning:** This is expressed in many forms, from difficulty tuning to matchmaking, social recommendations and hints. In reality this is not too dissimilar from next best offer, but broader in scope. If you are solely looking at match results, you can do that on the device. However, if you look at greater datasets: match results, current entitlements, inventory, results compared to peers, applicable offers, etc., you benefit from larger ML models that can be found in your backend. This applies socially as well. Who am I near? How do they relate to me? What is our likelihood to enjoy playing together? Taking full advantage of this requires one to build into the game the hooks to make it possible, so this isn't simply an add-on, it's highly effective.

Do This in 2024

Mitigating Player Toxicity and Creating Positive Gaming Environments

Overview

Addressing player toxicity is critical due to its immediate negative impacts on the health of your game community and the financial bottom line of your title. Toxic experiences significantly increase player churn. A study from Michigan State University revealed that 80% of players recently experienced toxicity, and of those, 20% reported leaving the game due to these interactions. Real-time data analytics and machine learning are essential in identifying and mitigating such behaviors promptly, as delayed responses can exacerbate the issue. Unlike churn, where interventions can also lag, the real-time nature of gaming makes swift action imperative to prevent further damage and revenue loss.

Detail

Player toxicity manifests in three primary categories: communication, playstyle and cheating. Each requires distinct strategies for combating toxicity and promoting positive interactions.

For communication, implementing word filters, utilizing natural language processing (NLP) to assess message content and monitoring responses can help identify and address toxic messages. Voice communications present additional challenges but can be analyzed by converting speech to text for similar scrutiny. Engaging with offenders quickly to correct behavior is crucial, as is supporting affected players through real-time interventions.

Playstyle toxicity, while subjective, can be addressed by analyzing gameplay sessions and feedback from other players. Patterns of reports against a player for disruptive playstyles offer opportunities for corrective dialogue. All frustrating playstyles are not necessarily toxicity. For these situations — such as individuals "camping" a spawn point — educating players who are feeling negatively impacted on ways to deal with these frustrating playstyles can enhance their experience and lead to long term player engagement.

Cheating, the most clear-cut form of toxicity, requires a combination of anti-cheat software and machine learning to detect anomalies. Community management teams play a vital role in investigating flagged behaviors, contributing to a supervised learning model that refines detection over time. This dual-model approach allows for more efficient identification of cheaters, potentially enabling real-time interventions.

Outcome

Investing in mechanisms to combat player toxicity is not only about preserving brand image and financial health but also about fostering a supportive and loyal community. Ignoring toxicity risks alienates players and undermines game success. Proactive measures and investments in positive play initiatives are essential for long-term engagement and community building.

Best practices

Beyond technological solutions, clear communication regarding the consequences of toxic behavior is essential. Providing feedback on why certain actions are penalized can help modify behavior. Encouraging community discussions on positive play and supporting players who report toxicity are also key. Rewarding long-term account holders can discourage banned players from simply creating new accounts, addressing the issue more effectively. Engaging with both offenders and those affected by toxicity in a constructive manner can promote a healthier gaming environment.

Diving In

Before we get to the core use cases of game data, analytics, and AI, let's cover the different types of game data and how they are produced, followed by how that data is received in the cloud to collect, clean and prepare for analysis.

Producing game data

Speaking in generalities, there are four buckets of data as it relates to your video game.

1. Game telemetry

Game telemetry refers to the data collected about player behavior and interactions within a video game. The primary data source is the game engine. And the goal of game telemetry is to gather information that can help game developers understand player behavior and improve the overall game experience.

Some of the primary metrics that are typically tracked in game telemetry include:

- **Player engagement:** Track the amount of time players spend playing the game, and their level of engagement with different parts of the game.
- **Game progress:** Monitor player progress through different levels and milestones in the game.
- **In-game purchases:** Track the number and value of in-game purchases made by players.
- **Player demographics:** Collect demographic information about players, such as age, gender, location, and device type.
- **Session length:** Monitor the length of each player session, and how often players return to the game.
- **Retention:** Track the percentage of players who return to the game after their first session.
- **Player behavior:** Track player behavior within the game, such as the types of actions taken, the number of deaths, and the use of power-ups.
- **User acquisition:** Track the number of new players acquired through different marketing channels.

2. Marketplace data

The second bucket of data is business key performance indicators (or KPIs) pulled from marketplace sources on mobile, console and PC devices; such as Apple and Google App Stores, Nintendo, PlayStation and Xbox marketplaces, Epic Games Store, Origin, and Steam. Marketplace data helps measure the performance and success of a video game from a business perspective. These KPIs help game studios understand the financial and operational performance of their games and make informed decisions about future development and growth.

Some of the primary business metrics that are typically tracked include:

- **Revenue:** Track the total revenue generated by the game, including sales of the game itself, in-game purchases, and advertising.
- **Player Acquisition Cost (CAC):** Calculate the cost of acquiring a new player, including marketing and advertising expenses.
- **Lifetime Value (LTV):** Estimate the amount of revenue a player will generate over the course of their time playing the game.
- **Player retention:** Track the percentage of players who continue to play the game over time, and how long they play.
- **Engagement:** Measure the level of engagement of players with the game, such as the number of sessions played, time spent playing, and in-game actions taken.
- **User acquisition:** Track the number of new players acquired through different marketing channels and the cost of acquiring each player.

- **Conversion rate:** Measure the percentage of players who make an in-game purchase or complete a specific action.
- **Gross margin:** Calculate the profit generated by the game after subtracting the cost of goods sold, such as the cost of game development and server hosting.

3. Game services

Similar to game telemetry, game services provide critical infrastructure that requires careful monitoring and management. These services include things like game server hosting, multiplayer networking, matchmaking, player voice and chat and more. Here the source of data is the game services used.

Some of the common metrics game teams typically track for these services include:

- **Concurrent players:** Track the number of players who are simultaneously connected to the game servers to ensure that the servers have enough capacity to handle the player demand.
- **Server availability:** Monitor the uptime and downtime of the game servers to ensure that players have access to the game when they want to play, particularly important for global live service games where demand fluctuates throughout the day.
- **Latency:** Measure the time it takes for data to travel from the player's device to the game server and back, to ensure that players have a smooth and responsive gaming experience.
- **Network bandwidth:** Monitor the amount of data transmitted between the player's device and the game server to ensure players have a high-quality gaming experience, even on slow internet connections.
- **Live operations:** Monitor the success of in-game events, promotions and other live operations to understand what resonates with players and what does not.
- **Player feedback:** Monitor player feedback and reviews, including ratings and comments on social media, forums and app stores, to understand what players like and dislike about the game.
- **Chat activity:** Track the number of messages and interactions between players in the game's chat channels to understand the level of social engagement and community building in the game.

4. Data beyond the game

The last bucket comes from data sources beyond the video game. These typically include the following:

- **Social media data:** Social media platforms, such as Facebook, Twitter, TikTok and Instagram, can provide valuable insights into player behavior, feedback and preferences, as well as help game teams understand how players are talking about their games online with different communities.
- **Forum data:** Online forums and discussion boards, such as Reddit and Discord, can be rich sources of player feedback and opinions about the game.
- **Player reviews:** Ratings and reviews on app stores, such as Steam, Epic, Google Play and the Apple App Store, can provide valuable feedback on player experiences and help game teams identify areas for improvement.
- **Third-party data:** Third-party data sources, such as market research firms, industry data providers and external data and application providers, can provide valuable insights into broader gaming trends and help game teams make informed decisions about their games and marketing strategies.

This is a lot of data. It's no wonder that studios globally struggle with fragmented views of their audience, with data often outpacing legacy technologies. Today, the need for real-time capabilities and the leap from descriptive to predictive analytics has made it so that data, analytics and AI are now table stakes for a game to be successful. By tapping into these four buckets of data sources, you will find actionable insights that drive better understanding of your playerbase, more efficient acquisition, stronger and longer lasting engagement, and monetization that deepens the relationship with your players.

The secret to success is bringing all of the disparate data sources together, so you have as complete a 360-degree view as possible of what's happening in and around your game.

That is what we are going to dig into throughout the rest of this book.

Let's begin with how to get data out of your game!

There are a variety of ways to get data out of the game and into cloud resources. In this section, we will provide resources for producing data streams in Unity and Unreal. In addition, we will also provide a generic approach that will work for any game engine, as long as you are able to send HTTP requests.

Unity

Since Unity supports C#, you would use a .NET SDK from the cloud provider of your choice. All three major cloud providers have .NET SDKs and the documentation for each is linked below.

No matter the cloud provider, if you want to use an SDK, install it through the NuGet package manager into your Unity project. Here is a walkthrough of how to implement the .NET SDK using AWS.

- **AWS:** AWS .NET SDK - Unity considerations
- **GCP:** GCP .NET SDK Documentation
- **Azure:** Azure .NET SDK Overview
- **Kafka (Open-source alternative):** Kafka .NET connector

From this point, the SDK is used to send data to a messaging service. These messaging services will be covered in more detail in the next section.

Unreal Engine

Unreal supports development in C++, so you would use the C++ SDKs.

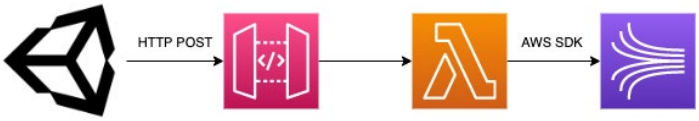
The resources for each SDK are provided here:

- **AWS:** How to integrate AWS C++ SDK with Unreal Engine
- **Azure:** Azure C++ SDK with PlayFab
- **Kafka (Open-source alternative):** Getting started with Kafka and C++

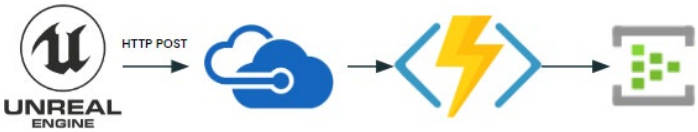
Just like with the Unity example above, from this point the data is sent to a messaging streaming service.

Other engines may not support C++ or C#, but there is still a way to get your data into the cloud, no matter the language! By hitting an API Gateway with a HTTP POST request, you are able to send data to cloud services from many more types of applications. A sample high-level architecture of this solution in AWS and Azure can be seen below:

AWS:



Azure:



… and receiving it in cloud

Once the data is sent from the game into an event-streaming service, how do we get that data to a more permanent home? We will start by outlining what these messaging services do and how we can use them to point our data to a desired location.

Messaging services will ingest real-time event data streamed from a number of different sources, and then send them to their appropriate target locations. These target locations can be databases, compute clusters or cloud object stores. A key property of the messaging services is to preserve the time in which the events arrive, ensuring the order that events occurred is always known.

Examples of cloud messaging services include AWS Kinesis Firehose, Google PubSub and Azure Service Bus Messaging. If you prefer to use open-source products, Apache Kafka is a very popular open-source alternative.

Getting data from your game to the cloud

Moving to the cloud platform part of the journey involves building a gaming lakehouse. The gaming lakehouse allows gaming companies to store, manage and analyze large volumes of gaming data, such as player behavior, performance metrics, and financial transactions, to gain valuable insights and make data-driven decisions to improve their business outcomes.

Here are the basics of the Databricks platform, simplified:

Data ingestion:

- Data can be ingested into the gaming lakehouse using various built-in data ingestion capabilities provided by Databricks, such as Structured Streaming and Delta Live Tables for a single simple API that handles streaming or batch pipelines.
- Data can be ingested in real-time, near-real-time or batch mode from various sources, such as game clients, servers or APIs.
- Data can be cleaned, transformed and enriched with additional data sources, making it ready for analysis.

Data storage:

- Data is stored in object storage such as S3, Azure Storage or GCP Buckets using Delta Lake.
- Delta Lake provides a transactional storage layer on top of a distributed file system that allows to maintain data consistency and track changes.

Data governance and cataloging:

- Unity Catalog in Databricks provides tools for data governance that helps in compliance and controlling access to data in the lake.
- Unity Catalog also allows to track data lineage, auditing and data discovery with the use of data catalogs and governance
- Metadata about the data including the structure, format, and location of the data can be stored in a data catalog.

Data quality:

- Databricks platform allows you to validate, clean and enrich data using built-in libraries and rule-based validation using Delta Live Tables.
- It also allows the tracking of data quality issues and missing values by using Databricks Delta Live Tables.

Data security:

- Databricks provides a comprehensive security model to secure data stored in the lake.
- Access to data can be controlled through robust access controls on objects such as catalogs, schemas, tables, models, experiments and clusters.

Analytics:

- The processed data can be analyzed using various tools provided by Databricks, such as SQL dashboards, notebooks, visualization and ML.
- Game studios can gain insights into player behavior and performance, and use these insights to improve their games for better player engagement.

Jump to: Get started with the three major clouds →

The Value of Data Throughout the Game Development Lifecycle

Lifecycle overview

Over the last decade, the way games have been developed and monetized has changed dramatically. Most, if not all, top grossing games are now built using a games-as-service strategy, meaning titles shipped in cycles of constant iteration to increase engagement and monetization of players over time. Games-as-a-service models have the ability to create sticky, high-margin games, but they also heavily depend on cloud-based services, such as game play analytics, multiplayer servers and matchmaking, player relationship management, performance marketing and more.

Data plays an integral role in the development and operation of video games. Teams need tools and services to optimize player lifetime value (LTV) with databases that can process terabytes-to-petabytes of evolving data, analytics solutions that can access that data with near real-time latency, and machine learning (ML) models that can translate insights into actionable and innovative gameplay features.

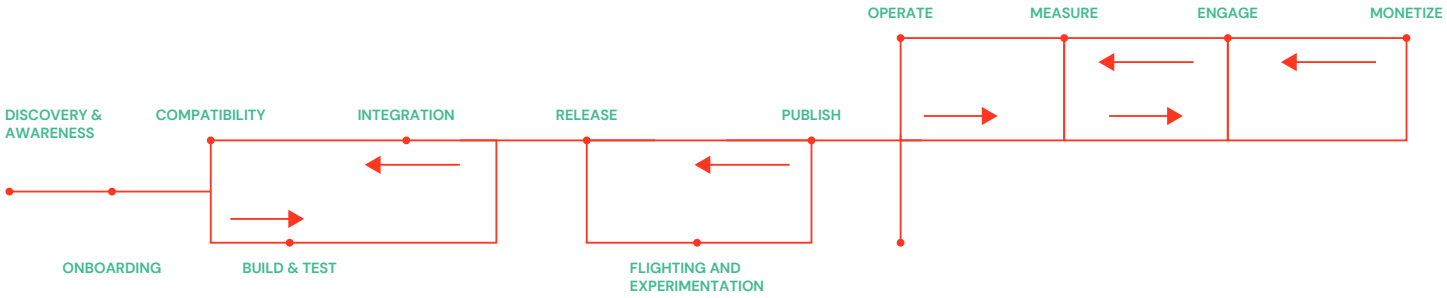
Game Development Lifecycle

Game-as-a-service (GaaS) / Game-as-a-Community (GaaC)

1. Pre-Production
Brainstorm how to give life to the many ideas laid out in the planning phase

3. Testing
Every feature and mechanic in the game needs to be tested for game loop and quality control

5. Operation
As studios increasingly adopt games-as-a-service models, the ongoing operation of a video game is as critical as the launch itself



2. Production
Most of the time, effort, and resources spent on developing video games are spent in production stage

4. Launch
Whether developing alongside the community with alpha and beta releases, or launching into general availability, a game launch is a critical milestone

A game's development lifecycle is unique to each studio. With different skillsets, resources and genres of games, there is no one model. Below is a simplified view of a game development lifecycle for a studio running a games-as-a-service model.

What's important to remember is that throughout your title's development lifecycle, there is data that can help you better understand your audience, more effectively find and acquire players, and more easily activate and engage them. Whether using game play data to optimize creative decision making during pre-production, tapping machine learning models to predict and prevent churn, or identifying the next best offer or action for your players in real-time, **data is your friend**.

Throughout the rest of this guide, we will dig into the most common use cases for data, analytics, and AI in game development.

TL;DR:

This eBook was created to help game developers better wrap their heads around the general concepts in which data, analytics and AI can be used to support the development and growth of video games. If you only have 5 minutes, these takeaways are critical to your success.

For more information on advanced data, analytics, and AI use cases, as well as education resources, we highly recommend Databricks training portal dbricks.co/training.

Top takeaways:

If you take nothing else from this guide, here are the most important takeaways we want to leave with you on your journey.

1. **Data is fundamental.** Data, analytics, and AI play a role throughout the entire game development lifecycle — from discovery to pre-production, development to operating a game as a live service. Build better games, cultivate deeper player engagements, and operate more effectively by utilizing the full potential of your data.
2. **Define your goals.** Start by establishing the goal(s) of what you are hoping to learn and or understand around your game. Clear goals make it easier to identify key metrics to track. Example goals include: developing high-quality games that provide engaging and satisfying player experiences, increasing player engagement and retention by analyzing and improving gameplay and mechanics, and building a strong and positive brand reputation through effective marketing and community outreach.
3. **Identify and understand your data sources.** Spend time identifying and understanding the breadth of the data sources you are already collecting, be that game telemetry, marketplace, game services or sources beyond the game like social media. It's critical to collect the right data and track the right metrics based on the goals and objectives you've set for your game.
4. **Start small, and iterate quickly.** Recognize that goals and objectives evolve as you learn more about the interaction between your game and your playerbase. Data projects are most effective when scoped small with tight feedback loops, allowing you to quickly adapt with your community and alongside shifting market conditions.
5. **Game analytics forms the foundation.** Start by getting a game analytics dashboard up and running. The process of building out a dashboard will naturally require connecting and transforming your data in to unlock more advanced use cases down the road.
6. **Check your assumptions at the door.** Heuristics are a critical part to making decisions. It's important, however, when you are looking at the data — the analytics insight that you get from your data platform — that you look at it with an open heart to avoid confirmation bias and other logical fallacies that are common in data work.
7. **Consider the problem from multiple angles and datasets.** Nowhere is this more critical than when segmenting your players. You want to move fast, you want it to be easy, so you look through one lens and then apply that across multiple use cases. Try taking a step back and considering, "The data that underpins this segmentation, does it make sense for the question I'm asking?" If you want to solve for engagement and retention, leveraging segmentation based on LTV or Buy/No Buy data will be less effective than looking at journey data, social engagement and milestones. It may seem overkill to have a single player represented across 3-4 different segments, but that is what will help you ensure your success.
8. **Plan and revisit your data strategy frequently.** Once dashboarding is set up, you'll have a better picture of what downstream data use cases make the most sense for your game and business objectives. As you move to use cases such as player segmentation, churn analysis and player lifetime value, revisit your data strategy frequently to ensure you're spending time on use cases that drive actionable insights for you and your team.
9. **Show value broad and wide.** Whether your data strategy is new or well established on the team, build the habit of communicating broadly to stakeholders across the company. It's important to gather critical feedback early in the process regarding what data is helpful and where there are opportunities for improvement. The worst thing that can happen is you create something that no one uses. That is a waste of your time and money.
10. **Ask for help. Engage with your technical partners.** There are people who can help ensure you're developing your data and analytics platform in a way that is efficient

and effective. There are numerous partners with domain expertise in data science and data engineering that can accelerate your data journey. Here is our recommended partner list for data, analytics, and AI workloads.

11. **Participate in the community.** The community for game analytics is large and growing. It's important to research and explore different communities to find the ones that best fit your needs and interests. Here are a few of our favorites:
    a. IGDA Game Analytics: The IGDA has a number of Special Interest Groups that bring together user researchers, designers, data engineers and data scientists focused on understanding player behavior and experiences. They offer resources and events for those working in games user research, including a yearly Games User Research Summit.
    b. Data Science Society: The Data Science Society is a global community of data scientists and engineers. While not specifically focused on game development, they offer a wealth of resources and opportunities for learning, networking, and collaboration in the field of data science.
    c. Hugging Face: A great hub of open source models for Natural Language Processing, computer vision, and other fields where AI plays its role. They also provide an online platform where users can access pre-trained models and tools, share their own models and datasets, and collaborate with other developers in the community.
    d. Data Engineering subreddit: The Data Engineering subreddit is a forum for data engineers to discuss topics related to building and managing data pipelines, data warehousing and related technologies. While not specifically focused on game development, it can be a valuable resource for those working on data engineering in the gaming industry.
12. **Go beyond dashboards.** Looking at dashboards is only the first step in your data journey. Imagine how the output of your data can be presented in a way to help stakeholders across your company achieve more. For example, dropping data into an application that can help game designers make balancing decisions based on player events.

Getting Started with Gaming Use Cases

For simplicity sake, we're going to take the game development lifecycle and categorize use cases into three core buckets.

Pre-production:

How to leverage data and AI to drive efficiencies in development.

Jump to: Pre-production →

From Beta to Global Launch:

Maximizing insights to reduce iteration cycles and mitigate risk.

Jump to: From Beta to Global Launch →

Live Operations:

Using data and AI to better acquire, engage, and retain your playerbase.

Jump to: Live Operations →

As we highlighted in the section above, data and AI can and should be used in every stage of game development. If you're already running down this path, fantastic! If not, it's never too late to start. Game teams who embrace data and AI build better games, their communities are healthier, and they can more easily maneuver as market forces shift and change. Note that many of the use cases we highlight in this ebook will apply across multiple stages of development. That said, we've tried to bucket them into the section where it is most applicable and/or critical. Let's jump in.

Pre-production use cases

Automating Your Build Pipeline

Overview

A build pipeline is a set of automated processes used to compile and assemble the code, assets and resources that make up a game project. The build pipeline typically includes several stages, such as code compilation, optimization, testing and release. The purpose of a build pipeline is to streamline the game development process and ensure that each stage of development is completed efficiently and effectively. A build pipeline can be configured to run automatically, so new builds are generated whenever changes are made to the code or assets. This helps to ensure the game is always up-to-date and ready for testing and release. The logs that are collected are in near-real time from build servers; a simplified example could be: Developer X is committing code on Y title, submitted on Z day along with the log files from the pipeline and build server. Builds typically take multiple hours to complete, requiring significant amounts of compute via build farms. Being able to understand which builds are relevant for testing, which builds are wasting compute, and being able to predict which builds will fail as they go through the pipeline are ways to curb operational expenses.

What we're trying to solve/achieve

With this use case, we're seeking to reduce wasted compute and build a foundational view of what was developed, by who, when and how testing performed. When we have this view, we're able to more easily find problem areas within our build process, which check-ins tend to be more problematic, what pieces are most brittle and-therefore-take preventative action to speed development. In an ideal state, our automated build pipeline could send a notification to the developer with a confidence metric on the build making it through, allowing them to decide whether to continue or move another build through the pipeline. Often, developers do not have easy visibility until the build has completed or failed. By providing more insight to developers into the build pipeline process, we can increase the rate at which builds are completed efficiently and effectively.

Getting started

An operational goal for game projects is to ensure game project builds are generated and delivered quickly and efficiently to internal testing and external users.

Here are a few of the key metrics and capabilities when analyzing your build pipelines:

- **Build time and speed:** The time it takes to create a build, number of builds and compute spent.
- **Build size and storage:** The size of the builds, amount of storage and network costs.
- **Bug tracking and resolution:** The number of bugs reported, the time it takes to resolve them and the number of bugs that are resolved in each build.
- **Code quality and efficiency:** Code complexity, code duplication and the number of code lines written.
- **Collaboration and communication:** The number of code reviews, the number of team meetings and the number of code commits.
- **Advanced capabilities:** Predicting real time build failure to reduce spend and combining build data with Crash Analytics (see below) to have "commit to build" visibility for accelerated bug fixing.

Before you start implementing your build pipeline, it's important to define your requirements. What are the key goals of your build pipeline? Choosing the right CI/CD tools is critical to the success of your build pipeline. There are many different tools available, including Jenkins, Azure Devops, Perforce, gitlab and more. When choosing a CI/CD tool, consider factors such as ease of use, scalability and cost. In addition, consider the specific needs of your game project, and choose a tool that can meet those needs.

The general recommendation is to look at automating your build process early. Once you've chosen your CI/CD tools, you can automate your build process by setting up a build server, configuring your CI/CD tool and creating a script to build your game project. The build process should be automated as much as possible, and it should include steps to compile your code, run automated tests and generate a build of your project.

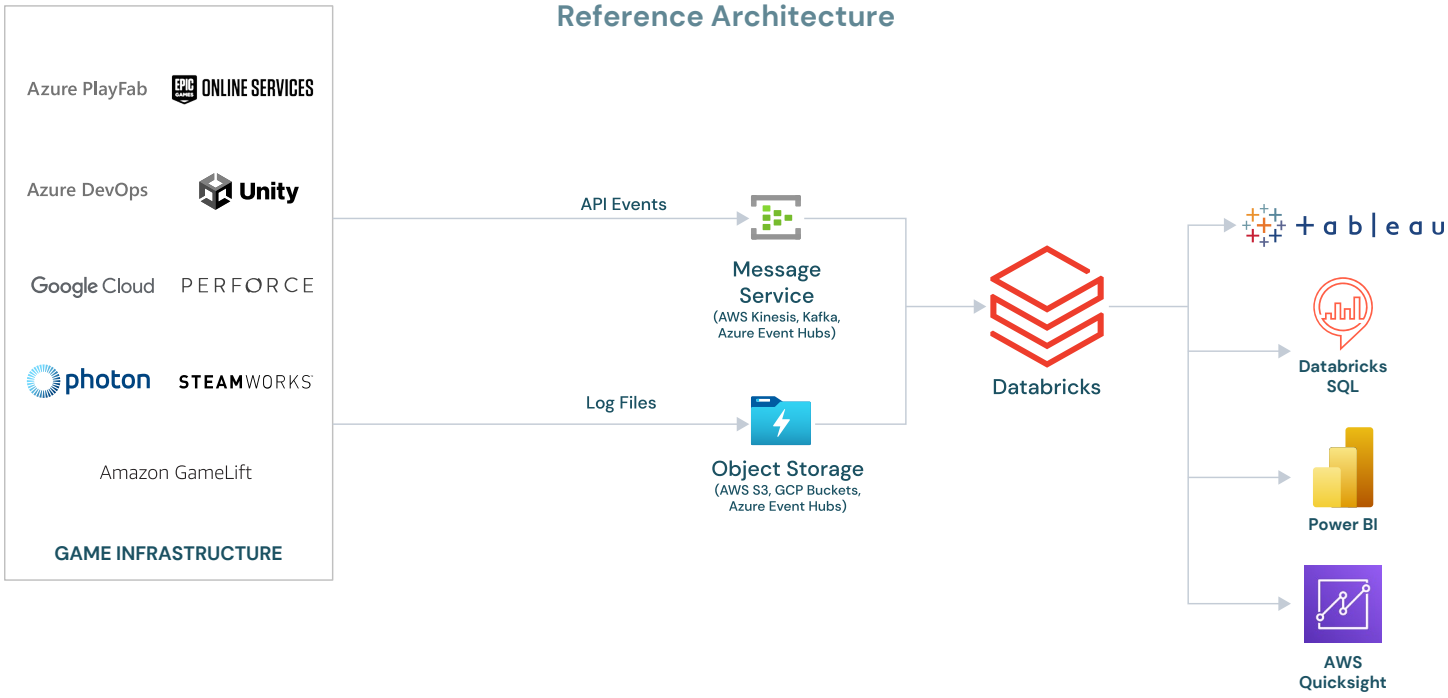
Once you have automated your build process, often the next step is to implement CD (Continuous Delivery). This involves automating the deployment of your game builds delivery to stakeholders, such as QA testers, beta testers or end-users via publishing platforms. CD can help ensure that stakeholders have access to the latest version of your game as soon as possible, allowing them to provide feedback and help drive the development process forward.

Finally, it's important to monitor and measure your build pipeline to ensure that it's working as expected. This can involve using tools such as Databricks Dashboards to visualize the status of your pipeline, or using metrics such as build times, test results and deployment success rates to evaluate the performance of your pipeline. By monitoring and measuring your build pipeline, you can identify areas for improvement and make changes as needed to ensure that your pipeline continues to meet your needs.

Tips/Best Practices

- **Seek to automate early and often:** Automate as much of the build process as possible, from checking code into version control to generating builds and distributing them to stakeholders. This can help reduce errors and save time, allowing game teams to focus on more high value tasks.
- **Version control, version control, version control:** Use a version control system to manage the source code and other assets. This ensures that changes to the codebase are tracked and can be easily undone if needed.
- **Implement continuous integration and delivery:** Continuous integration (CI) involves automatically building and testing after code changes are checked into version control. With CI, new changes to the codebase do not break existing functionality. By automating the build process, CI helps to reduce errors and save time. Continuous delivery (CD), on the other hand, involves automatically delivering builds to stakeholders, such as QA testers, beta testers, or end-users, after they have passed the automated tests. By combining CI and CD, a video game project can ensure that builds are generated and delivered quickly and efficiently, without the need for manual intervention.
- **Build for scalability:** As your game project grows, you will need a build pipeline solution that is scalable and can handle the needs of your game team.
- **Integration with other tools:** Integrate the build pipeline solution with other tools and systems, such as issue tracking, testing and deployment tools, to ensure a smooth and efficient workflow.
- **Analytics can be scary for developers:** How will the data be used? Will it impact my performance rating, my job, etc. Remember the purpose is to identify issues early to take corrective action: train, perform additional tests, pair up developers. When done well, this improves the efficiency of the development team and mitigates risk to morale.

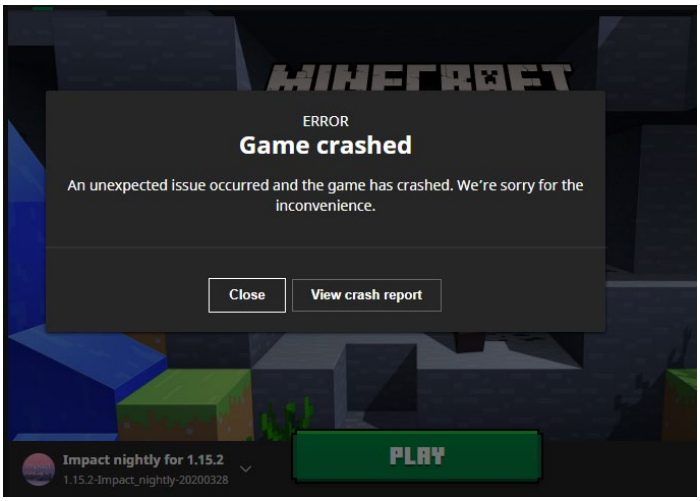
Reference Architecture



Crash Analytics

Overview

Games crash, it is a factor of game development. The combination of drivers, hardware, software and configurations create unique challenges in tracking, resolving and managing the user experience.



Crash analytics and reporting is the process of collecting information about crashes or unexpected failures in a software application, in this case, a video game. A crash report typically includes information about the state of the game at the time of the crash, such as what the player was doing, what assets were being loaded and what system resources were being used. How long crash testing takes can vary, depending on the game's business model, amount of content and scope. For a title with a one-time release, where there is a large amount of content and a complex storyline, the chances of hidden crashes causing errors while in development are high, making it require more time to perform testing before the game can be published. For titles built in a game-as-a-service model, i.e. a game shipped in cycles of constant iteration, crash detection should be done continuously, since errors in newly released content might affect the base game and lead to crashes.

Increasingly, titles are being released in alpha (where developers do the testing), closed beta (which includes a limited group of testers/sample-users who do the gameplay testing) and open betas (where anyone interested can register to try the game). All of which happens before the game is "officially" released. Regardless of alpha, beta, or GA, players may stumble over game crashes, which triggers crash reports that are sent to the developers for fixing. Sometimes it can be challenging to understand the issue that caused the crash from crash reports provided by your game's platform.

What we're trying to solve/achieve

Ultimately, the purpose of crash analytics is to identify the root cause of a crash and take steps to prevent similar crashes from happening in the future. This feedback loop can be tightened through automation in the data pipeline. For example, by tracking crashes caused on builds from committers, our data can provide build suggestions to improve crash rate. Furthermore, teams can automate deduplication when multiple players experience the same errors, helping to reduce noise in the alerts received.

Getting started

Building a pipeline to gain a holistic view to support crash analytics means collecting data coming from multiple different sources and velocities and joining the data together.

The number of data sources depends on your game projects publishing platforms; some may come from console-based providers such as Sony, Xbox, and Nintendo or pc platforms like Steam, Epic Games Marketplace, GoG and many others.

High level steps

- **Collect crash and related (meta)data:** Implement crash reporting tools in your game to collect data on crashes. You can store the crash data in a variety of formats, including JSON, CSV or Parquet.
- **Load crash data into Databricks:** Use Databricks' data ingestion tools to load the crash data into your workspace. This could involve using Databricks' built-in data source connectors or programmatically ingesting files to load the data.
- **Transform and cleanse the crash data:** Use Databricks' data processing and transformation tools to clean and prepare the crash data for analysis. This could involve using Databricks' capabilities like DLT or using SQL to perform custom transformations.
- **Visualize crash data:** Use Databricks' dashboarding tools to create visualizations that help you understand the patterns and trends in your crash data. This could involve using Databricks' built-in visualization tools or integrating with external visualization tools like Tableau or PowerBI.

- **Analyze crash data:** Use Databricks' machine learning and statistical analysis tools to identify the root causes of crashes. This could involve using Spark MLlib or many of the popular tools to build ML models, or using SQL to perform custom analyses.
- **Monitor and refine your pipeline:** Regularly review your pipeline to ensure it remains relevant and useful. Refine your pipeline as necessary to reflect changes in your crash data or your goals.

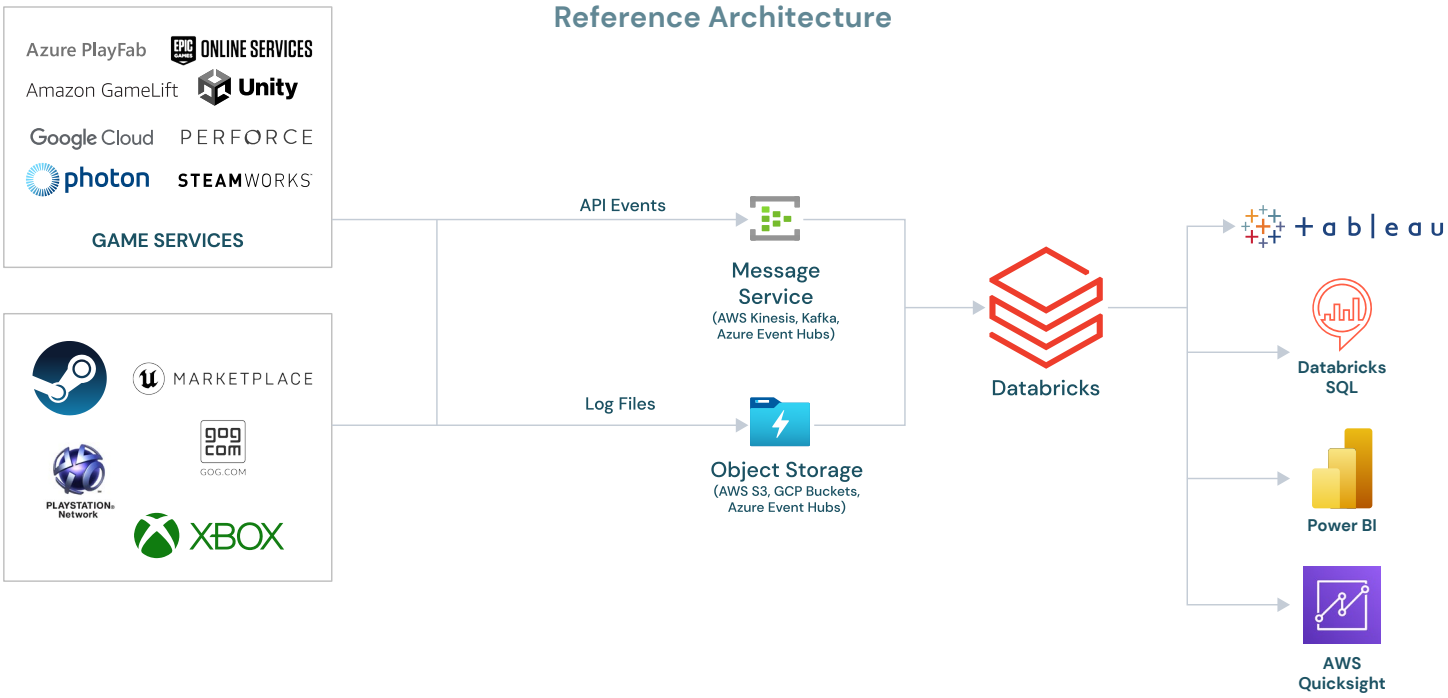
Tips / Best Practices

- **Automated collection and aggregation of crash reports:** Collecting crash reports should be an automated process integrated into the output of the build pipeline for the game. The crash reports should be automatically aggregated and made available for analysis in near real-time.
- **Clear reporting and prioritization of issues:** The solution should provide clear reporting on the most common issues and allow game developers to prioritize fixing the most impactful problems.
- **Integration with other analytics tools:** The crash analytics solution should integrate with other analytics tools, such as player behavior tracking, to provide a more complete picture of how crashes are impact the player experience.
- **Flexibility and scalability:** As the game grows, the solution should be able to scale to accommodate an increasing number of players and crashes.
- **The value of metadata:** The crash report itself will help your development team find issues with the code but equally valuable is the context of that crash. Were there network connectivity issues? Was it an old device? Did the device crash?
- **Data privacy and security:** It's important to consider data privacy and security when implementing a crash analytics solution. This may involve implementing measures to anonymize crash reports or taking steps to ensure that sensitive information is not included in the reports.

Additionally, there are some key gotchas to look out for when working with crash reporting analytics:

- **Data privacy and security:** Ensure that crash reports do not contain sensitive information that could be used to identify individual players.
- **Scalability:** As the number of players and crashes increase, it may become difficult to manage and analyze the growing volume of data.
- **Integration with other tools:** Be aware when integrating crash analytics with other tools and systems, especially if the tools use different data formats or data structures.

Reference Architecture



- **Prioritization of issues:** Determine which crashes are the most impactful and prioritize fixes accordingly. This can be a complex process, especially if there are a large number of different crash types and causes.

Anomaly Detection

Overview

Anomaly detection plays an important role in the operation of a live service video game by helping identify and diagnose unexpected behaviors in real-time. By identifying patterns and anomalies in player behavior, system performance and network traffic, this information can be used to detect and diagnose server crashes, performance bottlenecks and hacking attempts. The ability to understand if there will be an issue before it becomes widespread is immensely valuable. Without anomaly detection, which is a form of advanced analytics, you're always in a reactive (rather than proactive) state. Anomaly detection is a more robust quality of service solution.

What we're trying to solve/achieve

The goal of anomaly detection is to ensure that players have a stable and enjoyable gaming experience. This has an impact across your game, from reducing downtime, to minimizing player churn and improving your game's reputation and revenue. Additionally, the insights gained from anomaly detection can also be used to mitigate cheating and disruptive behavior.

Getting started

The first thing is to begin collecting the data, game server/client logs out of your project. The next step is to consume this into Databricks delta and run your continuous anomaly detection model. Focus on the key pieces of information you want to monitor. For example, live service games. This is going to be infrastructure and network-related metrics such as Ping and Server Health (# of clients connected, server uptime, server usage, CPU/RAM, # of sessions, time of sessions).

Once the model is ingesting and tuned specifically for the metrics based on the information you have above, build out alerts or notifications based on these specific metrics hitting a threshold that you define as needing attention. From here, you can build out automated systems to mitigate those effects, such as migrating players to a different server, canceling matches, scaling infrastructure and creating tickets for admins to review.

If you have any questions about how this solution can be deployed in your environment, please don't hesitate to reach out.

Tips / Best Practices

- **Define the problem and objectives clearly:** Before implementing an anomaly detection solution, it is important to define the problem you are trying to solve and your specific objectives. This will help ensure that you have the right data sources and use the appropriate algorithms to achieve your goals.
- **Choose the right data sources:** To effectively detect anomalies, you need to have the right data sources. Consider data from player behavior, system performance and network traffic, as well as any other data sources that are relevant to your problem and objectives.
- **Clean and preprocess the data:** To ensure that the data you use for anomaly detection is accurate and meaningful, it is important to clean and preprocess the data. This includes removing any irrelevant or invalid data, handling missing values and normalizing the data if necessary.
- **Choose the right algorithms:** There are many algorithms for anomaly detection, including statistical methods, machine learning algorithms and rule-based systems. Choose the algorithms that are best suited to your data and problem and that provide the right level of accuracy, speed and scalability.
- **Validate the results:** Before deploying the anomaly detection solution in production, it is important to validate the results by testing the solution on a small subset of data and comparing the results to expected outcomes.
- **Monitor and update the solution:** Once the anomaly detection solution is deployed, it is important to monitor its performance and accuracy and update the solution as needed. This may include retraining the algorithms, adding or removing data sources and updating the parameters and thresholds used by the algorithms.
- **Iterative process:** Tackling anomaly detection is often an iterative process, particularly as you build out your features. Explore, deploy, analyze your outcomes and improve. It will take more than one go at it.

Additionally, there are some key gotchas to look out for when implementing an anomaly detection solution.

- **Avoid overfitting:** Overfitting occurs when the anomaly detection solution is too complex and learns the noise in the data rather than the underlying patterns. To avoid overfitting, it is important to choose algorithms that are

appropriate for the size and complexity of the data and to validate the results using a separate test dataset.

- **False positive and false negative results:** False positive and false negative results can occur when the anomaly detection solution is not properly calibrated or when the solution is applied to data that is significantly different from the training data. To minimize the risk of false positive and false negative results, it is important to validate the results using a separate test dataset and to fine-tune the parameters and thresholds used by the algorithms as needed.
- **Scalability:** Scalability can be a concern when implementing an anomaly detection solution, especially when dealing with large amounts of data. To ensure the solution can scale to meet the demands of a growing player base, it is important to choose algorithms that are fast and scalable, and to deploy the solution using a scalable infrastructure.

From Beta to Global Launch Use Cases

From the moment a gamer interacts with your title, you're receiving invaluable data that will help determine the near and long-term direction of your game. During this phase, you want a robust, 360-degree view of your player base and what's happening inside and out of your title.

Start with a strong foundation in analytics, and quickly expand to valuable use cases such as player segmentation, player feedback analysis and more. This can help you inform a variety of key business decisions, from the highest macro order of "what game(s) to develop", to how to market and monetize those games and how to optimize the player experience.

By understanding the demographics, preferences and behaviors of your audience, your game studio can create player experiences that are more likely to drive sticky engagement and community appeal. You can also use this understanding to tailor your marketing and monetization strategies to the needs and preferences of your community.

Let's start with building a strong foundation in game analytics.

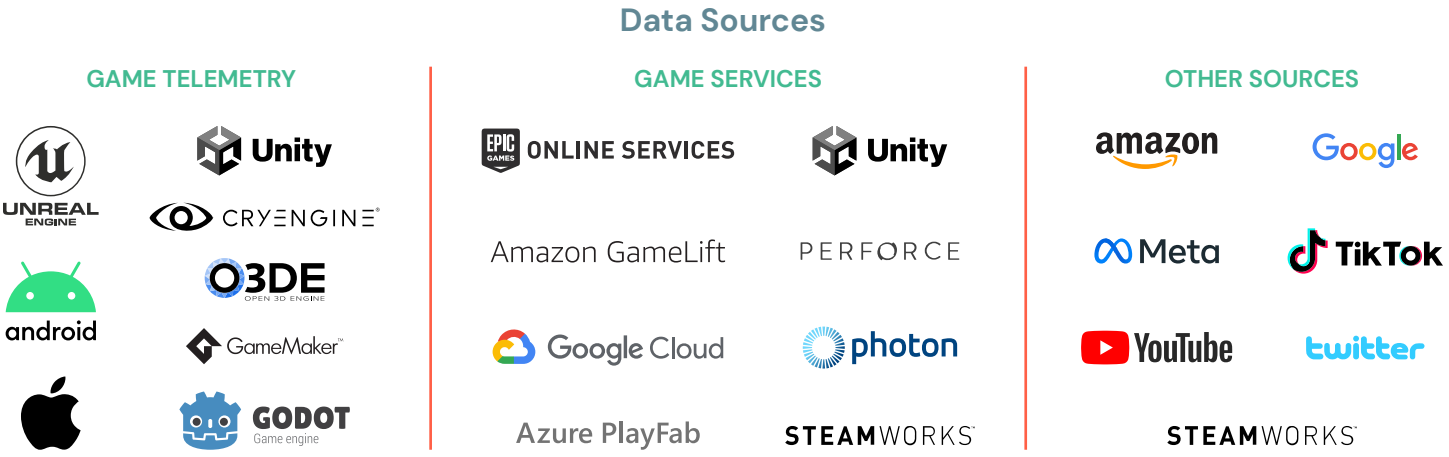


FIGURE 2

Game Analytics

Overview

Big question: Where is the best place to start when it comes to game data, analytics, and AI?

For most game studios, the best place to start is with game analytics. Setting up a real-time dashboard for your game analytics that helps correlate data across disparate sources is infinitely valuable in a world where there is no one gaming data source to rule them all. An effective dashboard should include your game telemetry data, data from any game services and data sources outside of your game such as stores, marketplaces and social media. See **Figure 2**.

What we're trying to solve/achieve

Buidling a strong foundation in game analytics unlocks more advanced data, analytics and AI use cases. For example, concurrent player count plus store and marketplace data gives you the building blocks for player segmentation and lifetime value. Usage telemetry combined with crash reporting and social media listening helps you quickly uncover where players might be getting frustrated. Correlating chat logs, voice transcriptions and/or discord and reddit forums can help you identify disruptive behavior before it gets out of hand, giving you the tools to take actionable steps to mitigate toxicity within your community.

Getting started

Fortunately, standing up an effective analytics dashboard is getting easier. It all starts with getting your data into an architecture that sets your team up for success. Selecting any of the major cloud providers — AWS, Azure, GCP — you can land all your data into a cloud data lake, then use a lakehouse architecture to run real-time and reliable processing. Databricks can then help you visualize that data in a dashboard, or send to a visual analytics platform, such as Tableau.

- **Sign up for a Databricks account:** You'll need to create an account on the Databricks website in order to use the platform.
- **Access the Databricks portal:** Interact with the Databricks platform and run tasks such as creating clusters, running jobs and accessing data
- **Set up a development environment:** You'll need a development environment where you can write and test your code, whether you're using a local IDE or the Databricks Workspace.
- **Collect data:** Once you have your development environment set up, you can start collecting data from your game. This can involve integrating or building a SDK into your game code or using another tool to send data to Databricks
- **Process and analyze the data:** Once you have collected your data, you can use Databricks to process and analyze it. This can involve cleaning and transforming the data, running queries or machine learning algorithms, or creating visualizations.
- **Monitor and optimize:** Regularly monitor your analytics to ensure that they are accurate and relevant, and use the insights you gain to optimize your game.

Keep in mind that these are just general steps to get started with Databricks for game analytics. The specific steps you'll need to take will depend on your specific use case and needs.

Tips / Best Practices

- **Define your goals:** What do you want to learn from your analytics data? Having clear goals will help you focus on collecting the right data and making meaningful use of it.
- **Plan your data collection:** Determine what data you need to collect, how you will collect it and how you will store it.
- **Consider privacy:** Make sure you are transparent with your players about what data you are collecting and how you will use it and give them the option to opt out if they wish.
- **Use analytics to inform design:** Leverage your analytics data to inform decisions around game design, such a s any balance changes or new content targeting a specific audience.
- **Monitor and test your analytics implementation:** Regularly check your analytics to ensure that data is being collected correctly and conduct tests to validate the accuracy of your data.
- **Visualize your data:** Dashboarding your data is one of the most effective ways to quickly and effectively make sense of what's happening at a given moment in time.
- **Use data to improve player retention:** Analyze player behavior and use the insights you gain to improve player retention, such as identifying and addressing pain points or providing personalized content.
- **Collaborate with your team:** Share your analytics findings with your team and encourage them to use the data to inform their work.

- **Keep it simple:** Don't try to collect too much data or create overly complex analytics systems. Keep it simple and focused on your goals.
- **Start where you are:** If you have yet to gather all of your data, don't initially build complex models. Start with the data you have available to you and build from there.
- **Avoid traps and assumptions:** Golden cohorts are incredibly useful when preparing for a launch but they can be deceiving. Be careful making long-term assumptions based on these cohorts. Also don't fall into the trap of confirmation bias and overly relying on heuristics you have from previous experience. Listen to the data, investigate if it doesn't align with your expectations and try to keep a neutral eye.

Player Segmentation

Overview

Player segmentation is the practice of dividing players into groups based on shared characteristics or behaviors. Segmentation has a number of benefits. You can better understand your players, create more personalized content, improve player retention and optimize monetization, all of which contribute to an improved player experience.

What we're trying to solve/achieve

The primary objective of segmentation is to ensure you're not treating your entire playerbase the exact same. People are different, and your players have different motivations, preferences and behaviors. Recognizing this and engaging with them in a way that meets them where they're at is one of the most impactful ways you can cultivate engagement with your game. As we mentioned above, the benefits of segmentation are broad reaching. Through better understanding of your playerbase, you can optimize personalized experiences, tailoring content and customer experience to specific groups of players that increases engagement and satisfaction. Better understanding of your players also helps in improving player retention. By identifying common characteristics of players who are at risk of churning (i.e., stopping play), you can develop targeted strategies that only reach specific audiences.

Create advanced customer segments to build out more effective user stories, and identify potential purchasing predictions based on behaviors. Leverage existing sales data, campaigns and promotions systems to create robust segments with actionable behavior insights to inform your product roadmap. You can then use this information to build useful customer clusters that are targetable with different promos and offers to drive lower costs of acquisition or deeper engagement with existing players.

Getting started

Player segmentation is crucial to studios as it allows them to better understand their audience and tailor their game experience to meet their specific needs and preferences. By dividing players into different segments based on factors such as demographics, playing styles and in-game behavior, studios can gain valuable insights into what motivates and engages their players. This information can then be used to design games that not only provide a more enjoyable experience for players, but also drive player retention and increase revenue for the studio. In a competitive industry where player satisfaction is key to success, player segmentation is an essential tool for studios to stay ahead of the game.

Start by evaluating the segmentation goals such as:

- **Personalize the experience:** Change or create experience specific designs for the player.
- **Create relevant content:** Surface the best content for players based on features and behaviors that will matter the most, depending on the player's place in the games life cycle.
- **Monetization:** Create tailored monetization strategies that effectively reach and convert each player group. For example, you may have a group of highly engaged players who are more likely to make in-app purchases, while another group is less likely to spend money but may be more receptive to advertisements.

The next steps would be to identify, Collect and Analyze player data. By gathering information on player behavior, preferences, and demographics, you can gain insights into their motivations, pain points, and what drives their engagement with your game.

There are many types of player data to consider, including:

- **Player Behavior:** Track player behavior and actions within your game to gain insights into their play style, preferences and patterns.
- **Surveys:** Ask players directly about their preferences, motivations and feedback through in-game surveys, email questionnaires or other forms of direct communication.

- **Focus groups:** Gather a small group of players to discuss and provide feedback on specific aspects of your game and player experience.
- **Social media listening:** Monitor social media platforms to gather insights into how players are engaging with and talking about your game.

Ready? Check out our solution accelerator on Customer Segmentation →

Tips / Best Practices

- **Define your segmentation goals:** Determine what you want to learn about your players and why. This will help you focus your analysis and ensure that your segments are meaningful and actionable.
- **Use meaningful criteria:** Choose criteria that are relevant to your goals and that differentiate players in meaningful ways. This could include demographic information, in-game behavior, spending habits or a combination of factors.
- **Analyze player data:** Use data from your players to inform your segmentation strategy. This could include data on in-game behavior, spending habits or demographic information.
- **Use multiple methods:** We recommend using a combination of methods, such as clustering and regression analysis, to create segments that are statistically meaningful and actionable to your game.
- **Use multiple lenses:** Don't try to over generalize a single segment across all your use cases. Gone are the days of trying to fit a player into a single segment. Segmentation across multiple types of data sets will enable you to combine this insight and focus your usage of segments for different use cases. You may segment on revenue, player behavior (game modes, ELO, toxicity), entitlements or social engagement (Clan/Guild involvement, chat engagement), and then you may create combined segments.
- **Validate your segments:** Test your segments to ensure that they accurately reflect the differences you observed in your player data. This could involve comparing the segments to each other or validating the segments against external data sources.
- **Consider ethical and privacy concerns:** Ensure that your segmentation strategy is ethical and complies with privacy laws and regulations. This could involve anonymizing your player data, obtaining consent from players or other measures to protect player privacy.
- **Monitor and refine your segments:** Regularly review your segments to ensure that they remain relevant and meaningful. Refine your segments as necessary to reflect changes in your player data or your goals.

Player Lifetime Value

Overview

Player lifetime value (LTV) is a measure of the value that a player brings to a game over the lifetime they play that game. It is typically calculated by multiplying the average revenue per user (ARPU) by the average player lifespan. For example, if the average player spends $50 and plays the game for 2 years, their LTV would be $50 * 2 = $100.

What we're trying to solve/achieve

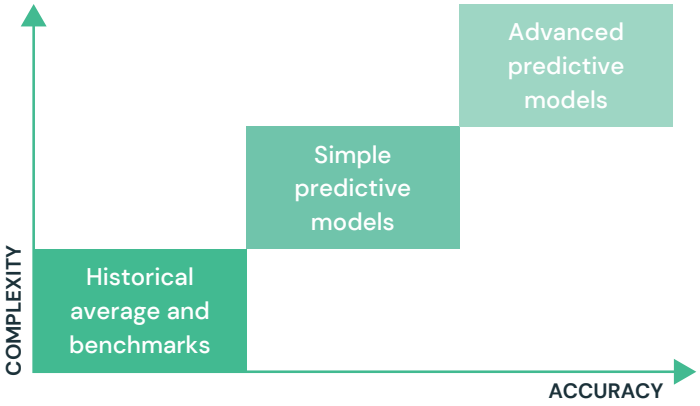
Game studios care about LTV because it helps them understand the long-term value of their players and make informed decisions about how to invest in player acquisition and retention. For example, if the LTV of a player is higher than the cost of acquiring them (e.g., through advertising), it may be worth investing more in player acquisition. On the other hand, if the LTV of a player is lower than the cost of acquiring them, it may be more cost-effective to focus on retaining existing players rather than acquiring new ones.

LTV is one of the more important metrics that game studios — particularly those building live service games — can use to understand the value of their players. It is important to consider other metrics as well, such as player retention, monetization and engagement.

Getting started

Assuming you've followed the steps to collect, store and prepare your player data for analysis, it's time to calculate LTV. The quick and dirty way is to divide the total revenue by the total number of registered players. Note, LTV is a critical calculation for return on investment, which is player lifetime spend versus the amount spent on player acquisition. Ideally, you want lifetime spend to be equal to or more than cost of acquisition.

As long as your game and its community are currently active, any player lifetime value calculations should be considered models, not exact numbers. This is because many of the players you're considering are likely actively registered and actively playing, so the exact player LTV number is a moving target.



The first calculation is relatively simple. We suggest using average revenue per user (ARPU) —a game's daily revenue divided by the number of active users — to help you calculate lifetime value. First, you'll need to define what is an active player using retention values, which can be set to a week, multi-day or multi-week period of time depending on how your game has performed to date. You can then look at the number of users who churn on a given day, averaging with the number of days from the player's first visit to the current date (or the specific date you've considered the end for said exercise). This is your playerbase lifetime value (note not Player Lifetime Value). To get individual player lifetime value, divide daily revenue by the number of daily active users, and multiply that by the playerbase lifetime value to get your player LTV.

It's important to note that while we are calculating player lifetime value, the term is not entirely accurate since most player lifetimes are not over (particularly true for live service games). However, for the purpose of modeling, we recommend keeping the amount of time that you consider a lifetime relatively short, allowing you to extrapolate. Keeping the time period shorter helps mitigate inaccuracies. The longer you stretch out what you consider a lifetime, the more likely you are to collect inactive users in your count.

Nevertheless, these models are not entirely accurate since it doesn't take into account the players who are registered but have yet to generate any revenue. Instead, a data-driven approach pivoted around player segmentation or cohorts will generally yield more actionable insights, far more than calculating a single LTV for the entire player base.

You can define your game's cohorts in multiple ways. Perhaps the most obvious in terms of calculating LTV is going by daily active cohorts, or users who joined your game on the same day. You could also organize cohorts by users who joined your game through a certain ad campaign or promotional effort, by country or geographic location or by the type of device used.

Ready? Get started quickly using our solution accelerator for Lifetime Value →

Tips / Best Practices

- **Use multiple data sources:** To get a complete picture of a player's value, be sure to consider data from a variety of sources, including in-game purchases, ad revenue and other monetization strategies.
- **Consider player retention:** Player retention is a key factor in LTV. Be sure to consider how long players are likely to play your game when calculating LTV.
- **Use accurate data:** Make sure you are using accurate data when calculating LTV. This might involve cleaning and processing your data or using trusted sources such as in-game analytics tools.
- **Regularly review and update your LTV estimates:** Player LTV can change over time. Be sure to regularly review and update your estimates to ensure they are accurate.
- **Test and optimize:** Use experimentation methods such as A/B testing to see how different variables affect LTV, such as in-game events or pricing strategies. Use the insights you gain to optimize your LTV calculations.
- **Be aware of outside factors:** Your calculations should consider the many outside factors that can affect your LTV. Factors such as the virality of your game, spikes or surges in visitors due to unexpected promotions (influencers, reviewers talking about your game), significant changes to your game that generate positive user response and other organic lifts that are difficult to predict with existing data.
- **Consider sub-KPIs:** Which LTV is most meaningful to you? pLTV (Player), cLTV (Customer), pLTV (Predicted), pLTVD# (Predicted for Day X), tcLTV (Top Customer LTV... yeah, we made that one up!), ARPPU, ARPU, ARPPUxCohort. The point is that each of these give you a different aspect of your business's health. Consider which one is most valuable for your specific problem.
- **Unify these measurements:** There is nothing worse than having cLTV defined four different ways within an organization. One person says it is $10, another says $3, who is right? Maybe both are, but both sides look at it differently. One might be at a studio level while the other makes considerations from a publisher level. When you get more granular — game or genre level — you may benefit from

having a separate LTV because you have additional data sets that you can leverage to calculate it. Maybe the speed by which they complete the tutorial is a meaningful indicator for a time management game, but not for a design/fashion game. Why would the time management game not leverage that to better predict and impact their success? Why don't we call that gLTV (game Lifetime Value, or pgLTV. You get the point).

Social Media Monitoring

Overview

As the great Warren Buffet once said, "It takes 20 years to build a reputation and five minutes to ruin it. If you think about that, you'll do things differently." Now more than ever, people are able to use social media and instantly amplify their voices to thousands of people who share similar interests and hobbies. Take Reddit as an example. As of this writing, the largest video game community (also called a subreddit), r/gaming, has over 35 million members with nearly five hundred new posts and ten thousand new comments per day. There are over 120 game-specific subreddits that have more than ten thousand members (the largest being League of Legends with over 700,000 members). The discourse that takes place on online social platforms generates massive amounts of raw and organic data coming from passionate and opinionated users. This is a great opportunity to understand how customers think and discover exactly what they want.

Monitoring content online — across the internet and social media — for keyword mentions and trends for downstream processing and analytics is referred to as media monitoring. By applying media monitoring to social media platforms, game developers are able to gain new advantages that previously might not have been possible, including:

- Programmatically aggregate product ideas for new feature prioritization
- Promote a better user experience by automatically responding to positive or negative comments
- Understand the top influencers in the industry who can sway public opinion
- Monitor broader industry trends and emerging segments such as free-to-play games
- Detect and react to controversies or crises as they begin
- Get organic and unfiltered feedback of games and features
- Understand customer sentiment at scale
- Make changes faster to keep customer satisfaction high and prevent churn

By failing to monitor, understand and act on what customers are saying about the games and content you release as well as broader industry trends, you risk those customers leaving for a better experience that meets the demands and requirements of what they want.

What we're trying to solve/achieve

By monitoring and listening to what existing and potential customers are saying on social media, game developers are able to get a natural and organic understanding of how customers actually feel about the games and products they release, or gauge consumer interest before investing time and money in a new idea. The main process for social media monitoring is to gather data from different social media platforms, such as Twitter or YouTube, process those comments or tweets, then take action on the processed data. While customer feedback can be manually discovered and processed in search of certain keyword mentions or feedback, it is a much better idea to automate it and do it programmatically.

Getting started

Social media monitoring has three primary components: collecting the data, processing the results and taking action on the findings. When it comes to collecting social media data, whether you're looking for tweets, YouTube comments, or Reddit posts, it can be very easy to get started. Many social media platforms, such as Twitter, YouTube and Reddit all provide their own detailed and comprehensive APIs, making it easy to start gathering data from those platforms. They also provide documentation and code examples to help along the way. Once the data is collected, the next step is to process and prepare it to be used in the next step. Processing your data can range in complexity from a simple keyword filter or a more complicated approach, such as filtering by location, removing emojis, and censoring and substituting words. With the data collected and processed, it can move to the final stage and be analyzed for downstream use and actionable insights by applying sentiment analysis or text mining.

If a game studio is looking to save time and have the above steps performed for them, it may be appealing to buy a pre-built tool. The primary benefits of buying an off the shelf solution is that it is often faster and easier to get started. Additionally, tool development is handled by a third party with experience building media monitoring solutions.

On the other hand, building your own custom solution will provide more flexibility and control. Many pre-built media monitoring tools might not have the capabilities required to effectively process video, audio and image data, and may not be able to control the frequency in which data is processed, whether it be near real-time or batch. Additionally, pre-built solutions tend to take a generalist approach for NLP, whether it be keyword extraction, topic filtering, or sentiment analysis. This often leads to poor results and feedback, especially for an industry as unique as the gaming industry where certain industry-specific slang or terminology is frequently used. Overall, building your own media monitoring tool will provide greater control and flexibility leading to a better tailored return on investment, and luckily Databricks makes it even easier to get started. With the Databricks Data Intelligence Platform, built on lakehouse architecture, all data engineering, data science, machine learning and data analytics can be done in a single platform without having to stitch multiple systems and tools together. See **Figure 3**.

Data engineers can use Workflows and Jobs to call social media platform APIs on a scheduled basis and use Delta Live Tables to create declarative data pipelines for cleaning and processing the incoming data.

Data scientists can use tools such as ML-specific Databricks runtimes (DBRs) that come with many of the most popular and common libraries already installed, including MLflow which makes model development, tracking and serving easy and efficient, and various other tools such as AutoML and Bamboolib.

Data analysts are able to create real-time alerts, dashboards and visualizations using Databricks SQL.

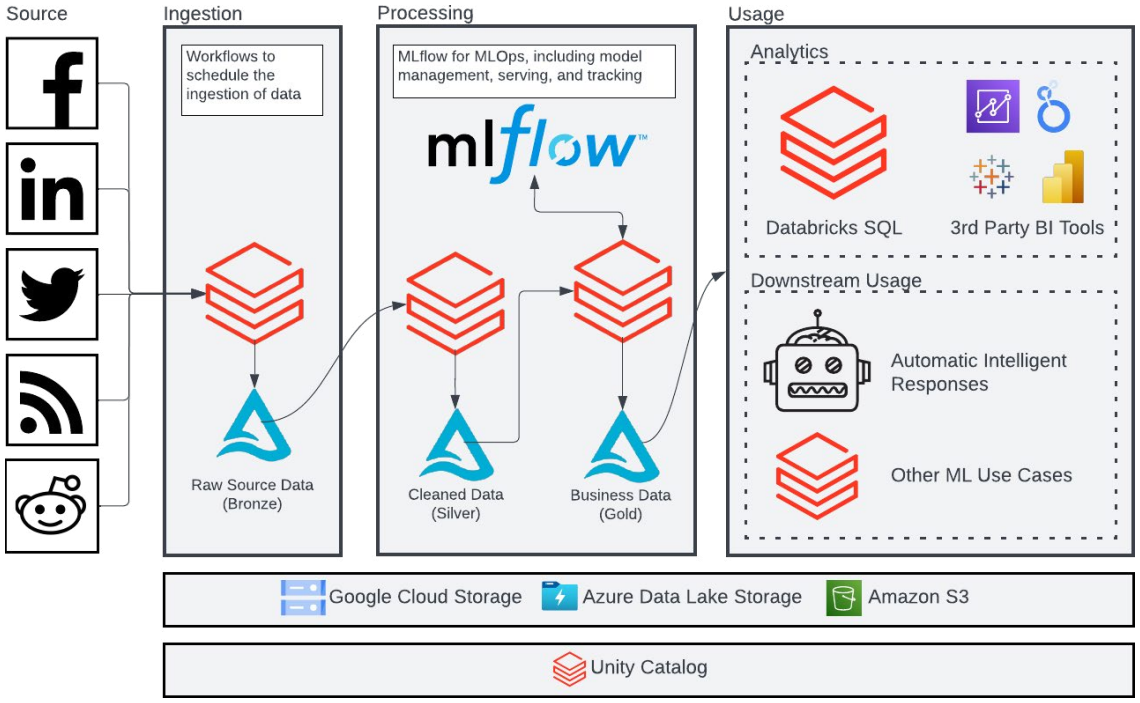
Each of the three personas will be able to effectively collaborate with each other and integrate each piece of their work into the broader data architecture.

Tips / Best Practices

While social media monitoring can be easy to get started with, there are a few key points to keep in mind.

- Remember the Pareto principle (roughly 80% of impact comes from 20% of activity) and diminishing returns. While it's important to monitor large platforms such as Reddit, Twitter and YouTube, it might not be worthwhile to monitor smaller platforms (in terms of engagement) as the bulk of customer feedback will be on those major platforms.
- Monitor other sources of information. It is also useful to monitor mentions of key company personnel such as executives or public facing employees.
- While follower count does matter on platforms such as Twitter, don't ignore users with low-follower counts. It only takes one or two re-tweets from other users to become a large issue.

FIGURE 3



- On social media, customers can see through generic corporate responses to complaints, so it is important to get a clear understanding of the issue and provide a clear response.

Player Feedback Analysis

Overview

Player feedback analysis is the process of collecting, analyzing, and acting on player feedback to inform game development. It involves collecting player feedback from multiple sources, such as in-game surveys, customer support tickets, social media, marketplace reviews and forums, and using data analytics tools to identify patterns, trends and insights. The goal of player feedback analysis is to better understand player needs, preferences and pain points, and use this information to inform game development decisions and improve the overall player experience.

Player feedback analysis is an important part of game development as it helps ensure that the game continues to meet player needs and expectations. By regularly collecting and analyzing player feedback, game studios can make data-driven decisions to improve the game, increase player engagement and retention and ultimately drive success and growth.

For this use case, we're going to focus on taking online reviews for your video game and categorizing the different topics players are talking about (bucketing topics) in order to better understand the themes (via positive or negative sentiment) affecting your community.

What we're trying to solve/achieve

Providing data-driven customer insight into your development process is incredibly helpful. Whether used in pre-production, such as looking at games that are similar with reviews to learn where those games have strengths and weaknesses or using player feedback analysis with a live service title to identify themes that can apply to your product roadmap, player feedback analysis helps teams better support and cultivate engagement with the player community.

Ultimately, player feedback analysis does two things:

1. Help you stack rank themes according to positive and negative sentiment
2. Weight those themes according to impact on player engagement, toxicity, monetization, churn, and more

We've all read reviews that are overly positive or overly negative. The process of player feedback analysis helps to normalize feedback across the community (keeping in mind, only for those who have written a review), so you're not over indexing on one review or a single theme that may seem in the moment as very pressing.

Getting started

The easiest place to start is gathering your data. With accounts set up on Steam, Epic, Apple, Google, Xbox, Sony, Nintendo (or whatever platform you're using), identify the ID for your game(s) and pull the reviews corresponding to that game into Databricks through an API call.

From here, clean the data using some of the pre-processing available in Python that removes any emojis and ascii characters. Once complete, run through Spark NLP pipeline which does the basic natural language processing steps such as normalization, stemming and lemmatization. We recommend running through pre-trained models, such as Word Embeddings and Named Entity Recognition models from John Snow Labs. This should complete the pipeline and generate the aspects for the reviews provided by the community.

This data is then loaded into a Delta table for further analysis, such as using a visual dashboard — built on SQL queries inside Databricks — to analyze and understand the aspects the community is talking about, then share back with the development team for analysis and action. This is a great exercise to perform one or more times per month.

Tips / Best Practices

- **Check for word groupings:** Make sure your word groupings are accurate to improve the analysis. For example, if your game is called Football Manager, and the shorthand is FM, make sure both of those are grouped appropriately.
- **Leverage domain knowledge:** Clean the reviews based on your domain knowledge. There are generic steps one could take, but that will not be as effective as someone with domain and specific game knowledge of your title
- **Experiment with models:** Feel free to try multiple pre-trained models and/or tweak the pre-trained models based on your understanding of the domain to improve the accuracy of your results.
- **Work one title at a time:** This process works best when pulling reviews for a single title, specifically one version of one title at a time.

- **Let the model to the heavy lift, but use humans to double-check:** The sentiment corresponding to the aspects in the model will be labeled as Positive or Negative. In the case of a neutral review, the model will do its best to determine whether that is more positive or negative. A best practice is to spend time going back through the aspects early to determine model accuracy and make updates accordingly.

Toxicity/Positive Play

Overview

Toxic behavior manifests in many forms, such as the varying degrees of griefing, cyberbullying and sexual harassment that are illustrated in the matrix below from Behaviour Interactive, which lists the types of interactions seen within the multiplayer game, *Dead by Daylight*. See **Figure 4**.

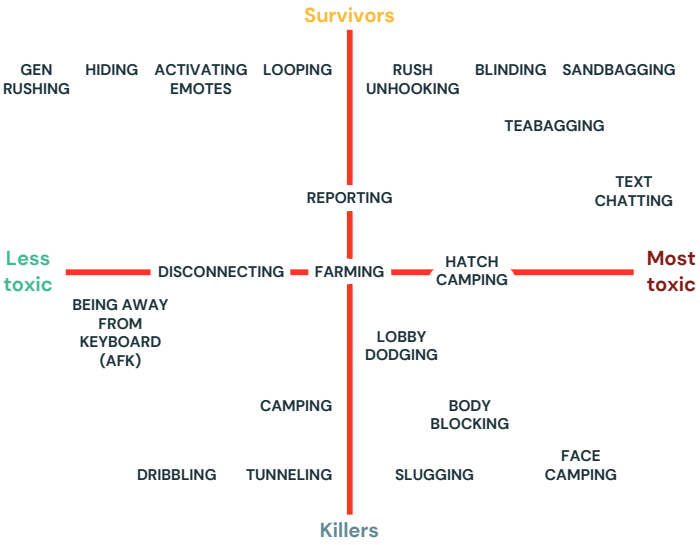


FIGURE 4

In addition to the personal toll that toxic behavior can have on gamers and the community — an issue that cannot be overstated — it is also damaging to the bottom line of many game studios. For example, a study from Michigan State University revealed that 80% of players recently experienced toxicity, and of those, 20% reported leaving the game due to these interactions.

Similarly, a study from Tilburg University showed that having a disruptive or toxic encounter in the first session of the game led to players being over three times more likely to leave the game without returning. While the damage is done in real time, too often corporate responses lag by days or even weeks. The earlier you can intervene, the more you can mitigate risk. Unfortunately, the impact of doing this wrong can be even more detrimental.

Compounding this issue related to churn, some companies face challenges related to toxicity early in development, even before launch. For example, Amazon's Crucible was released into testing without text or voice chat due in part to not having a system in place to monitor or manage toxic gamers and interactions. This illustrates that the scale of the gaming space has far surpassed most teams' ability to manage such behavior through reports or by intervening in disruptive interactions. Given this, it's essential for studios to integrate analytics into games early in the development lifecycle and then design for the ongoing management of toxic interactions.

A banned player produces no revenue. Often times they'll sign up with a new account and having just experienced a ban are unlikely to purchase additional goods within the game. Not only have you eliminated a source of revenue, you've created a drain against your profitability. Inaction, however, isn't acceptable either as their toxic behavior will cause other players to leave the game. For these reasons, consider the toxic player's player journey and look for ways to intervene in an attempt to correct the behavior before it becomes a habit. The goal is to promote a positive play experience, banning players only when all other options are exhausted.

What we're trying to solve/achieve

Toxicity in gaming is clearly a multifaceted issue that has become a part of video game culture and cannot be addressed universally in a single way. With that said, addressing toxicity within in-game chat can have a huge impact given the frequency of toxic behavior and the ability

to automate the detection of it using natural language processing (NLP). In summary, leveraging machine learning can help to better identify disruptive behavior and enable better-informed decisions regarding how to handle such actions.

Getting started

There are three core toxic behavior categories: Communication, Playstyle and Cheating. Each one will require attention to promote positive play and each is improved when you leverage streaming data techniques. For this use case, we'll focus on communication.

Our recommendation on tackling the toxicity issue is to leverage cloud-agnostic and flexible tooling that can consume chat data from a variety of sources, such as chat logs, voice transcriptions or sources like discord and Reddit forums. No matter if the data is in log form from game servers or events from a message system, Databricks can provide quick and easy ways to ingest the data.

Leveraging a simplified architecture like the diagram below shows that, no matter the source, getting chat data for inferencing and model development can simple. While we leveraged a pre-built model from John Snow Labs to accelerate development, you can bring the ML framework of your choice to the platform. See **Figure 5**.

For text communication you may implement word filters in chat, leverage NLP to score the intensity, negativity or hatefulness of a full message, or even score the response to a message to validate the impact it has on others.

Voice communications are harder to modify in flight, but you may convert to text and run through the same process as above to determine whether or not to flag that conversation.

You may look at intensity, volume, speed of speech and other metrics to see the intention beyond the words. Ultimately, you want to engage with the toxic player to correct the behavior, and do so quickly. The faster you contact them and let them know they're coming close to a line, or have crossed it, the more likely the player will remember the situation, understand the impact and have an opportunity to address it.

Similarly you want to engage with the players who were negatively impacted by the experience, ensure they're OK, offer help if there's a way to do so, provide them an opportunity to self-ban that player so they aren't in another match with them, etc. You want to do this in near-real-time as well to avoid them rage quitting, churning, talking bad about the brand or being reminded of the incident after they had forgotten about it

Ready? Get started quickly using our solution accelerator for Gaming Toxicity →

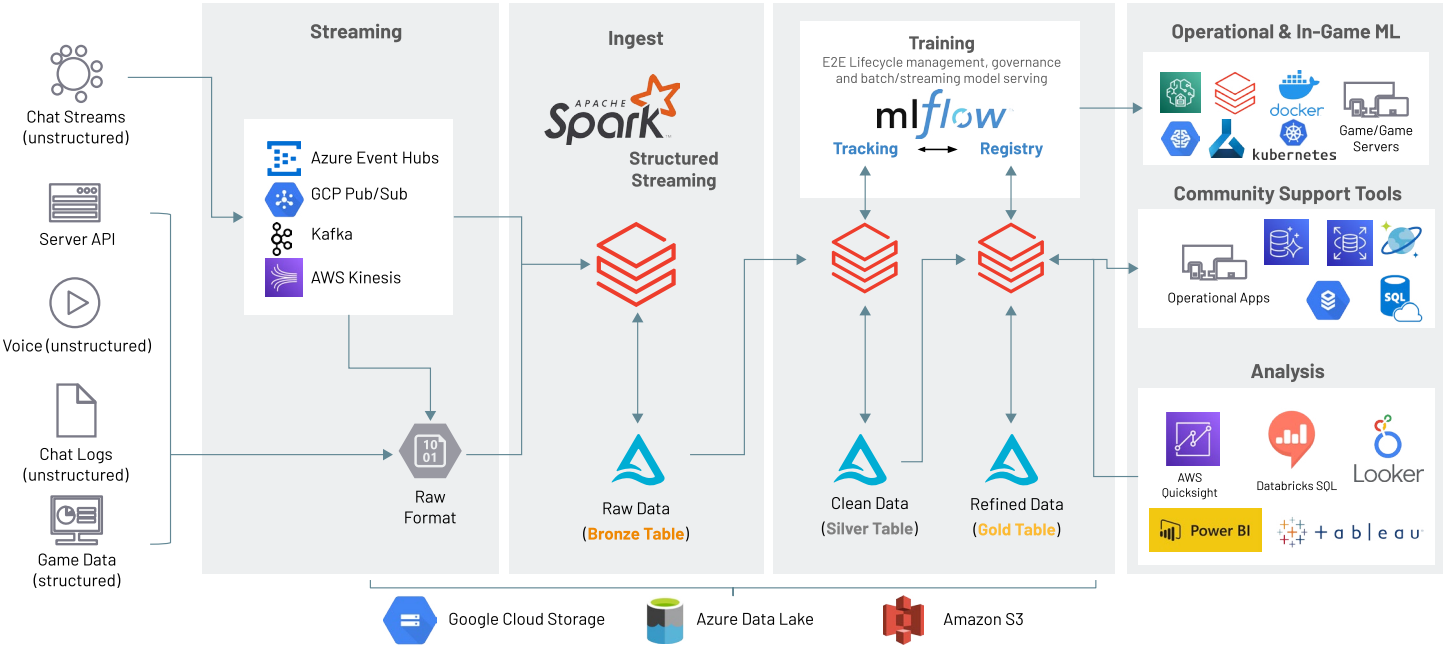


FIGURE 5

Tips/Best Practices - things to consider

- **Define what toxic and disruptive behavior looks like within your community:** Clearly define what you consider to be toxic behavior, as this will determine how you measure and detect it. This might include things like hateful language, harassment, or cheating.
- **Collect relevant data:** Make sure you are collecting the right data to help you detect toxicity. This might include data on in-game chat, player reports, and other sources.
- **Use machine learning:** Use machine learning algorithms to analyze your data and identify patterns of toxic behavior. This will allow you to more accurately detect toxicity and prioritize cases for review.
- **Be transparent:** Should a ban, or the threat of one, be warranted it is critical to provide feedback as to why. Engage your community in a conversation on the impact of positive play, help them understand what they're doing and the impact it has on the game as a whole.
- **Take action:** When toxic behavior is detected, take appropriate action to address it and make sure to stand up for players who choose to report, take time to educate and engage with them, and the offending players.

Live Operations Use Cases

Next, let's explore how data can enable you to create a more engaged and loyal playerbase that stays with your game for the long-term as well as strategies that differentiate your gamer experience.

Player Recommendations

Overview

Player recommendations are suggestions for content or actions that a game studio makes to individual players based on their interests and behaviors. These recommendations can be used to promote specific in-game items, encourage players to try new features or simply provide a personalized experience.

What we're trying to solve/achieve

Player recommendations matter to game studios because they can help improve player retention, engagement and monetization. By providing players with recommendations that are relevant and engaging, studios can increase the likelihood that players will continue to play their games and make in-game purchases. Additionally, personalized recommendations can help improve the overall player experience and increase satisfaction.

Game studios can use a variety of techniques to create player recommendations, such as machine learning algorithms, collaborative filtering, and manual curation. It is important to regularly review and optimize these recommendations to ensure that they are effective and relevant to players.

Getting started

Recommendations is an advanced use case and relies on the output of other analytic and machine learning outputs. As such we don't recommend (hehe) that you start here. Instead, we're assuming that you've done the work to set up your game analytics (collecting, cleaning, and preparing data for analysis) and that you've done basic segmentation to place your players in cohorts based on their interests and behaviors. Once you've done that, the groundwork has been laid to make player recommendations. Without meaningful segmentation your recommendations will be no less random then throwing a dart at the wall.

Recommendations can come in many forms for video games. For this context, we're going to focus on the wide-and-deep learning for recommender systems, which has the ability to both memorize and generalize recommendations based on player behavior and interactions. First introduced by Google for use in its Google Play app store, the wide-and-deep machine learning (ML) model has become popular in a variety of online scenarios due to its ability to personalize user engagements, even in 'cold start problem' scenarios with sparse data inputs.

The goal with wide-and-deep recommenders is to provide an intimate level of player understanding. This model uses

explicit and implicit feedback to expand the considerations set for players. Wide-and-deep recommenders go beyond simple weighted averaging of player feedback found in some collaborative filters to balance what is understood about the individual with what is known about similar gamers. If done properly, the recommendations make the gamer feel understood (by your title), and this should translate into greater value for both the player and you as the business.

Understanding the model design

To understand the concept of wide-and-deep recommendations, it's best to think of it as two separate, but collaborating, engines. The wide model, often referred to in the literature as the linear model, memorizes users and their past choices. Its inputs may consist simply of a user identifier and a product identifier, though other attributes relevant to the pattern (such as time of day) may also be incorporated. <<insert image033>>

The deep portion of the model — so named, as it is a deep neural network — examines the generalizable attributes of a user and their choices. From these, the model learns the broader characteristics that tend to favor users' selections.

Together, the wide-and-deep submodels are trained on historical product selections by individual users to predict future selections. The result is a single model capable of calculating the probability with which a user will purchase a given item, given both memorized past choices and generalizations about a user's preferences. These probabilities form the basis for user-specific rankings, which can be used for making recommendations.

Building the model

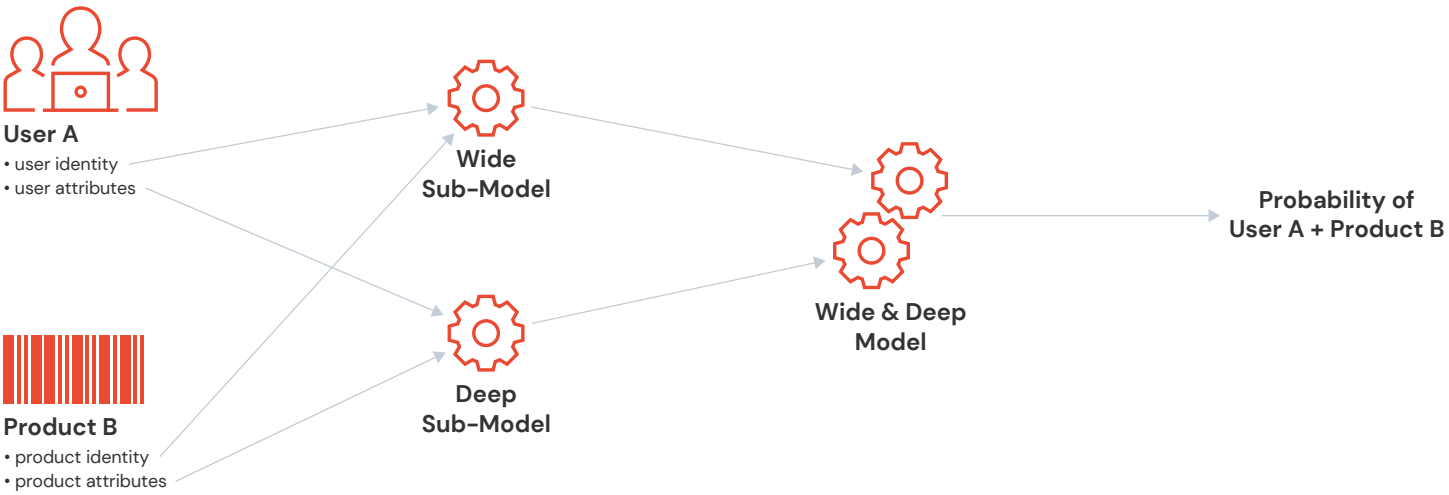
The intuitive logic of the wide-and-deep recommender belies the complexity of its actual construction. Inputs must be defined separately for each of the wide-and-deep portions of the model and each must be trained in a coordinated manner to arrive at a single output. However, they must be tuned using optimizers specific to the nature of each submodel. Thankfully, the Tensorflow DNNLinearCombinedClassifier estimator provides a pre-packaged architecture, greatly simplifying the assembly of an overall model.

Training

The challenge for most teams is then training the recommender on the large number of user-product combinations found within their data. By using Petastorm, an open-source library for serving large datasets assembled in Apache Spark™ to Tensorflow (and other ML libraries), one can cache the data on high-speed, temporary storage and then read that data in manageable increments to the model during training. In doing so, we limit the memory overhead associated with the training exercise while preserving performance.

Tuning

Tuning the model becomes the next challenge. Various model parameters control its ability to arrive at an optimal solution. The most efficient way to work through the potential parameter combinations is simply to iterate through some number of training cycles, comparing the models' evaluation metrics with each run to identify the



ideal parameter combinations. By leveraging hyperopt with SparkTrails, we can parallelize this work across many compute nodes, allowing the optimizations to be performed in a timely manner.

Deploying

Finally, we need to deploy the model for integration with various retail applications. Leveraging MLflow allows us to both persist our model and package it for deployment across a wide variety of microservices layers, including Azure Machine Learning, AWS Sagemaker, Kubernetes and Databricks Model Serving.

Although, in this case, a number of technologies are required to build a single model, Databricks integrates all of these technologies within a single platform, providing data scientists, data engineers and MLOps Engineers a unified experience. The pre-integration of these technologies means various personas can work faster and leverage additional capabilities, such as the automated tracking of models, to enhance the transparency of the organization's model building efforts.

To see an end-to-end example of how a wide and deep recommender model may be built on Databricks, please check out the following notebooks: Get the notebook

Ready? Get started quickly with our solution accelerator on Recommendation Engines →

Tips/Best Practices – things to consider

- **Use data to inform recommendations:** Use data from your analytics, player feedback and other sources to understand what players like and dislike. This will help you create recommendations that are more likely to be relevant and engaging for individual players.
- **Segment your players:** Consider segmenting your players based on characteristics such as playstyle, spending habits, their game journey, game outcomes, device type and other demographic information. This will allow you to create more targeted recommendations for different groups of players.
- **Consider the player's current context:** When creating recommendations, consider the player's current context, such as what they are doing in the game and what content they have already consumed. This will help you create recommendations that are more likely to be relevant and timely.
- **Be timely in usage:** Once you've considered context and created a recommendation, the recommendation needs to be made within the timeline of that context to be effective.
- **Test and optimize your recommendations:** Use experimentation methods such as A/B testing to see how different recommendations perform with different player segments. Use the insights you gain to optimize your recommendations.
- **Be transparent:** Make sure you are transparent with players about how you are creating recommendations and give them the option to opt out if they wish.
- **Use recommendations to improve the player experience:** Use personalized recommendations to improve the player experience and increase engagement and satisfaction.

Next Best Offer/ Next Best Action

Overview

Next best offer (NBO) and next best action (NBA) are techniques that businesses use to make personalized recommendations to their customers. NBO refers to the practice of recommending the most relevant product or service to a customer based on their past purchases and behaviors. NBA refers to the practice of recommending the most relevant action or interaction to a customer based on the same information.

For example, a game studio might use NBO to recommend an in-game purchase to a player based on their past spending habits and the items in which they've shown interest. They might use NBA to recommend a specific level or event to a player based on their progress and interests.

What we're trying to solve/achieve

It's important to remember that next best offer is a specific use case within personalization that involves making recommendations to players on the most valuable in-game item or action they should take next. For example, a next best offer recommendation in a mobile game might suggest that a player purchase a specific in-game currency or unlock a new character.

Both NBO and NBA can be used to improve customer retention, engagement and monetization by providing personalized recommendations that are more likely to be relevant and appealing to individual customers. They can be implemented using a variety of techniques, such as machine learning algorithms or manual curation.

Getting started

Since NBO/NBA is a specific use case of personalization, how a team might get started implementing this will look very similar to how they would with broader personalization activities.

Begin with ensuring you are appropriately collecting player data (behavior, preferences, in-game purchases, etc), and storing it in your cloud data lake using a service such as Delta Lake from Databricks. Then use Databricks to clean, transform and prepare the data for analysis. This may include aggregating data from multiple sources, removing duplicates and outliers, and transforming the data into a format suitable for analysis. As you analyze the player data, seek to identify patterns and trends in player behavior and preferences that will give you signal on which actions are more likely to be successful.

Then you can build a recommendation model based on the player data analysis and incorporate information on in-game items and player preferences to make personalized recommendations.

Tips/Best Practices

- **Define your goals:** Like every use case, starting with clearly defined goals helps to ensure your implementation of NBO and NBA will be as effective and efficient as possible. Your goals will also help you determine what data to collect and how it will be used.
- **Collect relevant data:** Based on your goals, make sure you are collecting the right data to inform your NBO and NBA recommendations. This might include data on player behavior, engagement and spending habits (e.g. provide an offer in the price range they've shown a willingness to spend).
- **Leverage machine learning to scale your recommendations:** Use machine learning algorithms to analyze your data and make personalized recommendations to your players. This will allow you to identify trends and patterns that might not be immediately apparent.
- **Test and optimize:** THIS IS CRITICAL. Use experimentation methods, such as A/B testing, to see how different recommendations perform with different player segments. Past performance is not a perfect indicator of future success. Consistent testing allows you to tune your NBO and NBA recommendations so they evolve with your playerbase.
- **Consider the player's context:** When making recommendations, consider the player's current context, such as what they are doing in the game and what content they have already consumed. This will help you create recommendations that are more likely to be relevant and timely. Make sure you deploy the offer, or action, within that context.
- **Be transparent:** Make sure you are transparent with your players about how you are using their data to make recommendations and give them the option to opt out if they wish.
- **Collaborate with your team:** Share your NBO and NBA efforts with your team and encourage them to use the data to inform their work.

Churn Prediction and Prevention

Overview

Video games live and die by their player base. For Games-as-a-Service (GaaS) titles, engagement is the most important metric a team can measure. Proactively preventing churn is critical to sustained engagement and growth. Through churn prediction and prevention, you will be able to analyze behavioral data to identify subscribers with an increased risk of churn. Next, you will use machine learning to quantify the likelihood of a subscriber to churn, as well as indicate which factors create that risk.

What we're trying to solve/achieve

Balancing customer acquisition and retention is critical. This is the central challenge to the long-term success of any live service game. This is particularly challenging as successful customer acquisition strategies, those needed to get games to scale, tend to be followed by service disruptions or declines in quality and customer experience, accelerating player abandonment. To replenish lost subscribers, the acquisition engine continues to grind and expenses mount. As games reach for customers beyond the core playerbase they initially targeted, the title may not resonate with new subscribers over the same durations of time or may overwhelm the ability of these players to consume, reinforcing the overall problem of player churn.

At some point, it becomes critical for teams to take a cold, hard look at the cost of acquisition relative to the subscriber lifetime value (LTV) earned. These figures need to be brought into a healthy balance and retention needs to be actively managed, not as a point-in-time problem to be solved, but as a "chronic condition" which needs to be managed for the ongoing health of the title.

Headroom for continued acquisition-driven growth can be created by carefully examining why some players leave and some players stay. When centered on factors known at the time of acquisition, gaming studios may have the opportunity to rethink key aspects of their acquisition strategy that promote higher average retention rates, which can lead to higher average revenue per user.

Prerequisites for use case

This use case assumes a certain level of existing data collection infrastructure in the studio. Notably, a studio ready to implement a churn prediction and prevention model should have:

- A cloud environment where player data is stored
- Source data that contains player actions within the game

These are the foundational elements on which churn prediction and prevention insights are built.

Getting started

The exciting part of this analysis is that not only does it help to quantify the risk of customer churn; it paints a quantitative picture of exactly which factors explain that risk. It's important that we not draw too rash of a conclusion with regards to the causal linkage between a particular attribute and its associated hazard. However, it's an excellent starting point for identifying where an organization needs to focus its attention for further investigation.

The hard part in this analysis is not the analytic techniques. The Kaplan-Meier curves and Cox Proportional Hazard models used to perform the analysis above are well established and widely supported across analytics platforms. The principal challenge is organizing the input data.

The vast majority of subscription services are fairly new as businesses. As such, the data required to examine customer attrition may be scattered across multiple systems, making an integrated analysis more difficult. Data Lakes are a starting point for solving this problem. However, the complex transformations required to cleanse and restructure data that have evolved as the business itself has (often rapidly) evolved requires considerable processing power. This is certainly the case with the KKBox information assets and is a point noted by the data provider in their public challenge.

The key to successfully completing this work is the establishment of transparent, maintainable data processing pipelines executed on an elastically scalable (and therefore cost-efficient) infrastructure, a key driver behind the Delta Lake pattern. While most organizations may not be overly cost-conscious in their initial approach, it's important to remember the point made above that churn is a chronic condition to be managed. As such, this is an analysis that should be periodically revisited to ensure acquisition and retention practices are aligned.

To support this, we are making the code behind our analysis available for download and review. If you have any questions about how this solution can be deployed in your environment, please don't hesitate to reach out.

Ready? Get started quickly using our solution accelerator for Churn Prediction →

Tips/Best Practices

- **Define churn:** Clearly define what you consider to be player churn, as this will determine how you measure and predict it. For example, you might consider churn to be when a player stops playing your game for a certain number of days, or when they uninstall it.
- **Collect relevant data:** Make sure you are collecting the right data to help you predict and prevent churn. This might include data on player behavior, engagement and spending habits.
- **Use machine learning:** Use machine learning algorithms to analyze your data and predict which players are at risk of churning. This will allow you to identify trends and patterns that might not be immediately apparent.
- **Test and optimize:** Use experimentation methods such as A/B testing to see how different strategies impact churn rates. Use the insights you gain to optimize your churn prevention efforts.
- **Focus on retention:** Implement retention strategies tailored to the needs and preferences of your players. This might involve providing personalized content, addressing

pain points or offering incentives to continue playing.

- **Consider player lifetime value when determining your course of action:** For a low value player, or a non-spender and predicted non-spender, you may take no action whereas a high-value player may warrant near real-time engagement.
- **Be transparent:** Make sure you are transparent with your players about how their data is used to predict and prevent churn, and give them the option to opt out if they wish.
- **Collaborate with your team:** Share your churn prediction and prevention efforts with your team and encourage them to use the data to inform their work.

Multi-touch Attribution

Overview

Multi-touch attribution is a method of attributing credit to different marketing channels or touchpoints that contribute to a sale or conversion. In other words, it is a way of understanding how different marketing efforts influence a customer's decision to make a purchase or take a desired action.

There are a variety of different attribution models to assign credit to different touchpoints, each with its own strengths and limitations. For example, the last-click model attributes all credit to the last touchpoint that the customer interacted with before making a purchase, while the first-click model attributes all credit to the first touchpoint. Other models, such as the linear model or the time decay model, distribute credit across multiple touchpoints based on different algorithms.

What we're trying to solve/achieve

Multi-touch attribution can be useful for game studios because it can help them understand which marketing channels or efforts are most effective at driving conversions and inform their marketing strategy. However, it's important to choose the right attribution model for your title based on your business model (one-time purchase, subscription, free-to-play, freemium, in-game advertising, etc) and regularly review and optimize your attribution efforts to ensure they are accurate and effective.

Getting started

To get started with multi-touch attribution, you need to first select an attribution model. There are a variety of different attribution models to choose from, each with its own strengths and limitations.

1. **Last-click model:** This model attributes all credit to the last touchpoint that the customer interacted with before making a purchase or taking a desired action.
2. **First-click model:** This model attributes all credit to the first touchpoint with which the customer interacted.
3. **Linear model:** This model attributes equal credit to each touchpoint with which the customer interacted.
4. **Time decay model:** This model attributes more credit to touchpoints that are closer in time to the purchase or desired action.
5. **Position-based model:** This model attributes a portion of the credit to the first and last touchpoints and the remainder is distributed evenly among the other touchpoints.
6. **Custom model:** Some businesses create their own attribution model based on specific business needs or goals.

Each attribution model has its own strengths and limitations. The right model for a particular video game will depend on a variety of factors, including the goals of your title, the customer journey and the types of marketing channels used. It is important to carefully consider the pros and cons of each model and choose the one that best aligns with the needs of your game.

Next, set up tracking. In order to attribute credit to different touchpoints, you'll need to set up tracking to capture data on customer interactions. This might involve integrating tracking code into the game or using a third-party tracking tool.

With tracking set up, you'll start collecting data on player interactions and be able to use that information to calculate attribution credit according to your chosen model (above). We highly recommend you regularly review and test your attribution efforts to ensure they are accurate and effective. Use experimentation methods, such as A/B testing, to see how different strategies affect conversion rates.

Ready? Get started quickly using our solution accelerator for Multi-touch Attribution →

Tips/Best Practices

- **Define clear goals:** By clearly defining the goals of your acquisition campaign and what success looks like, you will be able to guide your decision-making and ensure that you are measuring the right metrics, such as cost per install, return on ad spend, conversion rate, lifetime value, retention rate and more.

- **Use a data-driven approach:** Use data to inform your decision-making. Collect data on all touchpoints in the player journey, including ad impressions, clicks, installs and in-game actions.
- **Choose the right attribution model:** Select the right attribution model that accurately reflects the player journey for your specific genre of game. This can be a complex process. A couple of things to keep in mind include:
    - Consider the touchpoints that are most important for your player journey, such as first ad impression, first click or first in-game action
    - Consider the business goals you're trying to achieve. For example, if you are focused on maximizing return on investment, a last-click attribution model may be most appropriate. On the other hand, if you are looking to understand the impact of each touchpoint, a multi-touch attribution model may be more appropriate.
    - Consider the data you have available, including ad impressions, clicks, installs and in-game actions.
- **Continuously monitor and optimize:** Continuously monitor and optimize your acquisition campaigns based on the data. Test different approaches, make adjustments as needed and use A/B testing to determine what works best.
- **Test multiple attribution models:** When selecting the model that best fits your game, studio or organization, start by testing all the models against your desired outcomes. You may select a model that best correlates to high LTV, long-term retention metrics or social engagement, for use as your core metric for success over time.

Real-time Ad Targeting

Overview

Real-time ad targeting, in the context of game development, focuses on using data to deliver personalized and relevant advertisements to players in near real-time, while they are playing a game. Real-time targeting is performance based, using highly personalized messages, which are achieved by using data to precisely determine the most opportune moments to display ads, based on factors such as player behavior, game state, and other contextual information. Knowing when to send those ads is based on data. This use case is specific to titles using in-game advertising as a business model. It's important to note that in-game real-time ad targeting requires a sophisticated tech stack — with data processing and analytics tools — as well as integrations with a bigger ad ecosystem, ad networks and partners. The data intelligence platform is an optimal foundation as it already contains many of the connectors required to enable this use case.

What we're trying to solve/achieve

The goal of in-game real-time ad targeting is to provide a more immersive and relevant advertising experience for players, while also increasing the effectiveness of the ads for advertisers. Delivering targeted ads relevant to each player's interests allows game developers to create a more enjoyable and personalized gaming experience. This helps reduce churn and increase the lifetime value of each player. Additionally, real-time ad targeting can also help game developers monetize their games more effectively, as advertisers are willing to pay a premium for hyper-targeted and engaged audiences.

Getting started

Typically, implementing a real-time ad targeting strategy begins outside of your game (in services such as Google Ads, Unity Advertising), where your game becomes the delivery point for the advertisement. To do so, you will need to integrate with Ad networks that provide real-time ad targeting capabilities. That will allow you to access a range of available ad assets to dynamically select and display the most relevant ads to players. Both Google AdMob and Unity Ads are great for banner ads, native ads and rewarded video ads. Your role is to ensure that the data you're collecting is fed back into the advertising platform to better serve targeted ads to your playerbase.

To use a service like Databricks to manage the data needed to provide real-time ad targeting in your application, you can follow the below steps:

- **Collect and store player data:** Collect data on player behavior, preferences and demographics, and store it in a data lake using Databricks. Popular analytics tools such as Google Analytics or Mixpanel can be integrated into the game to collect data on player behavior. These tools, just like tracking website traffic, can track in-game events, provide insights on player behavior and demographics and they give you access to detailed reports and dashboards. Another option is to build in-house tracking systems to collect data on player behavior — logging events, such as in-game purchases or player actions, activities, such as "at which level does a player

quit playing — and storing it in a database for analysis. The downside of building in-house tracking systems is you will need to host and maintain your own logging servers.

- **Prepare the data:** Use Databricks to clean, transform and prepare the player data for analysis. This may include aggregating data from multiple sources, removing duplicates and outliers and transforming the data into a format suitable for analysis.
- **Analyze the data:** Use Databricks' built-in machine learning and data analytics capabilities to analyze the player data and identify patterns and trends.
- **Create audience segments:** Based on the analysis, use Databricks to create audience segments based on common characteristics such as interests, behaviors and preferences.
- **Integrate with the ad platform:** Next, your game needs to communicate with the ad network's platform and retrieve relevant ads, using each ad network's SDK. Within the app code is where the real-time ad targeting logic is implemented. For example, when building the model on Databricks resembles "if Gender=F, Location=New York, then Show ads image=XYZ game," you will still need to translate/embed the model into your game code and communicate with the Ad networks/Ad servers.
- **Monitor and optimize:** Use Databricks to monitor the performance of the ad targeting and make optimizations as needed, such as adjusting the audience segments or adjusting the targeting algorithms.

Using a service like Databricks to manage the data needed for real-time ad targeting allows game developers to effectively leverage player data to create more personalized and engaging experiences, increase revenue and reduce churn.

Tips/Best Practices

- **Focus on player data:** Make player data the center of your targeting strategy by collecting and storing comprehensive information on player behavior, preferences and demographics. It's critical to ensure the game code data trackers are properly implemented in order to collect this data (see Game Analytics section for detail).
- **Segment your audience:** Create audience segments based on common characteristics such as interests, behaviors and preferences, and use these segments to deliver targeted ads.
- **Test and iterate:** Continuously test and iterate your targeting strategy to refine your audience segments and improve targeting accuracy.
- **Balance relevance and privacy:** Balance the need for relevant, personalized ads with players' privacy by only collecting and using data that is necessary for targeting and obtaining player consent.
- **Monitor performance:** Regularly monitor the performance of your targeting strategy to ensure that it is delivering the desired results and make optimizations as needed.
- **Partner with the right ad platform:** Choose an ad platform that is well suited to your needs and aligns with your goals, and work closely with them to ensure that your targeting strategy is delivering the best results.

APPENDIX
Ultimate Class Build Guide

Who makes up data teams in game studios?

The heart and soul of mature data teams are formed by this trio of classes. There are many aspects to these roles, but they can be summarized in that Data Engineers create and maintain critical data workflows, Data Analysts interpret data and create reports that keep the business teams running seamlessly and Data Scientists are responsible for making sense of large amounts of data. Depending on the size of the organization, individuals may be required to multiclass in order to address needs of the team. In smaller studios, it's often developers who wear multiple hats, including those in data engineering, analytics and data science.

Whether you're looking to stand-up an analytics dashboard to report on the health of a title or building a recommendation engine for your players, the guidance that follows will help you better understand the unique classes required to develop and maintain an effective data, analytics and AI platform.

Data Engineers

Data engineers build systems that collect, manage and convert source data into usable information for data scientists and business analysts to interpret. Their ultimate goal is to make data accessible so that teams can use it to evaluate and optimize a goal or objective.

Responsibilities:

- Data migration, manipulation and integration of data (joining dissimilar data systems).
- Setup and maintenance of ETL pipelines to convert source data into actionable data for insights. It is the responsibility of the data engineer to make sure these pipelines run efficiently and are well orchestrated.
- Setting up the workflow process to orchestrate pipelines for the studio's data and continuously validates it
- Managing workflows to enable data scientists and data analysts, and ensuring workflows are well-integrated with different parts of the studio (e.g., marketing, test/QA, etc)

Goals and priorities of Data Engineers:

- Enable access to usable data for real-time insights — data that both enables timely decision-making and is accurate and reproducible
- Increase user confidence and trust in data. This involves ensuring high consistency and reliability in ETL processes.
- Limit the issues and failures experienced by other engineers and data scientists, allowing those roles to focus less on troubleshooting and more on drawing meaningful conclusions from data and building new products and features

What Data Engineers care about:

- Enabling access to data for real-time insights — data that both enables timely decision-making and is accurate and reproducible
- Building high-performance, reliable and scalable pipelines for data processing
- Delivering data for consumption from a variety of sources by Data Analysts and Data Scientists against tight SLAs
- A Data Engineer's biggest challenge? Collaboration across teams.

Data Scientists

Data scientists determine the questions their team should be asking and figure out how to answer those questions using data. They often develop predictive models for theorizing and forecasting.

Responsibilities:

- Making sense of the large amounts of data collected for a given game title, such as game telemetry, business KPIs, game health and quality, and sources beyond the game such as social media listening
- The analytics portion of a Data Scientist's job means looking at new and existing data to try and discover new insights
- The engineering component may include writing out pipeline code and deploying it to a repository
- As algorithms are developed, provide internal support of tools and data they create

Goals and priorities:

- Developing new business capabilities (such as behavioral segmentation, churn prediction, recommendations) and optimizing processes around those capabilities
- Increase ROI by building algorithms and tools that are maintainable and reusable
- Exploring (or further expanding) the use of machine learning models for specific use cases
- Bridging the gap between engineering and analytics, and between the technology teams and business teams
- Providing business leaders with data critical for decision-making (for example, a churn model that helps predict the impact of a new feature set)

What Data Scientists care about:

- Creating exploratory analysis or models to accurately predict business metrics (for example, customer spend, churn, etc.), and provide data-driven recommendations
- Enable teams with actionable insights that are easy to understand and well curated
- Create and move models from experimentation to production
- A Data Scientist's biggest challenge? Keeping up with advancements and innovation in data science, and knowing which tools and libraries to use

Data Analysts

A data analyst reviews data to identify key insights into a game studio's customers and ways the data can be used to solve problems.

Responsibilities:

- Often serves as the go-to point of contact for non-technical business/operations colleagues for quick data access/analysis questions
- Analysts often interpret data and create reports or other documentation for studio leadership
- Analysts typically are responsible for mining and compiling data
- Streamline and or simplify processes when possible

Goals and priorities:

- Empower stakeholder and business teams with actionable data
- Proactively mitigate potential data issues before they occur (for internal and external customers). In other words, "Catch things before they break."
- Analysts are often recruited to assist other teams (i.e., BI teams) with their data solutions (i.e., aggregated tables)
- Driving business impact through documentation and reliable data

What Data Analysts care about:

- Easy access to high quality data
- Quickly find insights from data with SQL queries and interactive visualizations
- The ability to easily share insights and while creating impactful assets for others to consume (dashboards, reports)
- A Data Analyst's biggest challenge? Working with complex processes and complicated technologies that are filled with messy data. While fighting these challenges, analysts are often left alone or forced through paths that prevent collaboration with others across team/organization.
- Another major challenge is untrustworthy data. Often Analysts are asked to provide answers to leadership that will leverage the data to determine the direction of the company. Data that is untrustworthy or incorrect due to the previously mentioned challenges can lead to lack of trust in the data teams from leadership or the business.

Data Access and the Major Cloud Providers

Cloud Rosetta Stone

https://cloud.google.com/free/docs/aws-azure-gcp-service-comparison

If you are newer to the cloud computing space, it is easy to get lost between the hundreds of different services between the three major cloud providers. The table below is meant to highlight the important data, analytics and AI services used by the various hyperscale service providers: Amazon, Microsoft, and Google. In addition, it aims to pair up services from different cloud providers that serve the same purpose.

Getting started with the major cloud providers

Here are some quick ways to get started with the three major cloud providers: AWS, Azure, and GCP:

AWS:

1. **Create an AWS account:** The first step is to create an account on the AWS website. This will give you access to the AWS Management Console, which is the web-based interface for managing your AWS resources.
2. **Use the AWS free tier:** AWS offers a free tier of service that provides a limited amount of free resources each month. This is a great way to get started and try out various AWS services without incurring any charges.
3. **Explore the AWS Management Console:** Once you have an account and are logged in, take some time to explore the AWS Management Console and familiarize yourself with the various services that are available.
4. **Next you can search for Databricks:** In the AWS Management Console, use the search bar in the top-left corner of the page and search for "Databricks."
5. **Navigate to the Databricks page:** Once you have found the Databricks page, you can access it to get started with the Databricks service.
6. **Launch Databricks Workspace:** To launch the Databricks Workspace on AWS, you can use the CloudFormation template provided by Databricks. Databricks CloudFormation template creates an IAM role, security group, and Databricks Workspace in your AWS account.

Azure:

1. **Create an Azure account:** The first step is to create an account on Azure portal. This will give you access to the Azure portal, which is the web-based interface for managing your Azure resources.
2. **Take Azure tutorials:** Azure provides tutorials, documentation, and sample templates to help you get started. These resources can help you understand the basics of Azure and how to use its services.

Service Type	Service Description	AWS Service	Azure Service	GCP Service
Storage	Object storage for various file types and artifacts (CSV, JSON, Delta, JAR). Objects can be retrieved by other services	Amazon Simple Storage Service (S3)	Azure Blob Storage	Google Cloud Storage
Compute	High-performance VMs to run applications. Platform where data transformations are run in Big Data apps.	Amazon Elastic Compute (EC2)	Azure Virtual Machines	Google Compute Engine
Messaging	Real-time event streaming services to write data to object stores or data warehouses. One OSS version is Kafka	Amazon Kinesis	Azure Service Bus Messaging	Google Pub/Sub
Data Warehouse	Traditional data storage layer for structured data, to then be used by data analysts. Often used to read from a Data Lake, which acts as a single source of truth	Redshift or Databricks	Synapse or Databricks	BigQuery or Databricks

3. **Search for Databricks:** In the Azure portal, use the search bar at the top of the page and search for "Databricks."
4. **Navigate to the Databricks page:** Once you have found the Databricks page, you can access it to get started with the Databricks service.
5. **Create a new Databricks workspace:** To create a new Databricks workspace, you can use the Azure portal, Azure CLI or Azure Powershell. Once created, you'll be able to access your Databricks Workspace through the Azure portal.
6. **Other Azure Services:** Once you have a Databricks workspace setup, you can easily connect it to other Azure Services such as Azure Storage, Event Hubs, Azure Data Lake Storage, Azure SQL and Cosmos DB for example.

GCP:

1. **Create a GCP account:** The first step is to create an account on the GCP portal. This will give you access to the GCP Console, which is the web-based interface for managing your GCP resources.
2. **Explore the GCP Console:** Once you have an account and are logged in, take some time to explore the GCP Console and familiarize yourself with the various services that are available.
3. **Search for Databricks:** In the GCP Console, use the search bar in the top-left corner of the page and search for "Databricks."
4. **Navigate to the Databricks page:** Once you have found the Databricks page, you can access it to get started with the Databricks service.
5. **Create a new Databricks workspace:** To create a new Databricks workspace, you can use the GCP Console or the gcloud command-line tool. Once created, you'll be able to access your Databricks Workspace through the GCP Console.

Jargon Glossary

CDP	Customer Data Platform (CDP). A CDP is a piece of software that combines data from multiple tools to create a single centralized customer database containing data on all touch points and interactions with your product or service.
Data Intelligence Platform	A Data Intelligence Platform combines the capabilities of a lakehouse architecture (a single platform for all data and AI workloads) with generative AI tooling and functionality. This provides an open, unified foundation for all data and governance and is powered by a Data Intelligence Engine that understands the uniqueness of your data.
ETL	Extract, Transform, Load. In computing, extract, transform, load is a three-phase process where data is extracted, transformed and loaded into an output data container. The data can be collated from one or more sources and it can also be outputted to one or more destinations
Feature Engineering	Extracting useful patterns from data to help your models answer your questions. Feature Engineering, done well, will limit the variables needed to be considered to solve for a specific question.
Five V's of Data	Volume: How much data. Velocity: How fast is the data received or processed. Variety: How dissimilar is the data received. Veracity: How true, or reliable, is the data. Value: How useful is the data
Lakehouse	An architecture that combines Data Lake and Data Warehouse capabilities into the same data platform enabling you to ingest, track, govern, transform, query and perform machine learning against the same datasets.
KPI	Key Performance Indicator: A quantifiable measure of performance over time for a specific objective. KPIs provide targets for teams to shoot for, milestones to gauge progress and insights that help people across the organization make better decisions.
Neural Network	Algorithms that recognize relationships between datasets in an attempt to mimic the functionality of the human brain.
NLP	Natural Language Processing: Taking in input that looks like normal language, not a coding language.
RAG	Retrieval Augmented Generation (RAG) is a technique in natural language processing (NLP) that combines retrieval-based AI with generative AI models. This approach enhances the capabilities of large language models (LLMs) by integrating them with external datasets or knowledge bases allowing them to retrieve and use up-to-date, relevant information to generate responses. RAG models are particularly effective in reducing inaccuracies, hallucinations (fabricated information), and the leakage of sensitive information, making them superior to non-RAG models in various metrics.
ROI	Return on investment (ROI): A calculation made by dividing the profit earned on an investment by the cost of that investment.
Serverless Computing	Using compute platforms that are completely managed by service providers. When using serverless computing, you simply execute queries or deploy applications and the service provider (AWS, Databricks, etc.) handles necessary server maintenance.
Supervised Learning	Learning that happens against labeled data. You've taken your dataset, put metadata (data about data, e.g. a label) on it and then used that to solve your problem.
Unsupervised Learning	Learning with data that is not labeled. The algorithm looks at the data, as it is and comes up with an answer. Clustering, most commonly K-Means Clustering, is a great example. Here you'll pick the variables (features) that the model should consider and then the model classifies based on the data itself.
VPC	Virtual Private Cloud: A virtual cloud networking environment which helps organize and give you control of your resources. You also define how resources within your VPC can communicate with other regions, VPCs, and the public internet with traffic rules and security groups.