System Design for an AI-Enabled Online Film Database

System Design

The goal of this AI-driven film recommendation system is to deliver a highly engaging platform that offers users film recommendations which accurately reflect their film taste. Through integrating a sophisticated recommendation engine into our platform, the business will align closer to its three key goals: maximising user retention, increasing platform monetisation, and gaining a competitive technological edge in the industry.

System Requirements

1. **Rich User Experience -** The platform must be seamless to use across devices, maintaining average page load speeds of sub one second. This will ensure higher user engagement which will lead to a lower bounce rate, a stronger SEO ranking and will in turn drive more users to the platform.

2. **Comprehensive User Data Collection -** The platform must collect a large quantity of user data for the effective training of the recommender models. This will ensure the models output accurate film recommendations, enhancing user satisfaction.

3. **Accurate Film Recommendations –** The platform must deliver highly relevant film recommendations to users, maintaining a minimum 95% accuracy is providing positive recommendations.

4. **Dynamic Model Development and Deployment –** New models should be tested and deployed periodically, training on the expanding user interaction data and a model retraining system should be triggered when the served models recommendation accuracy falls below 80%.

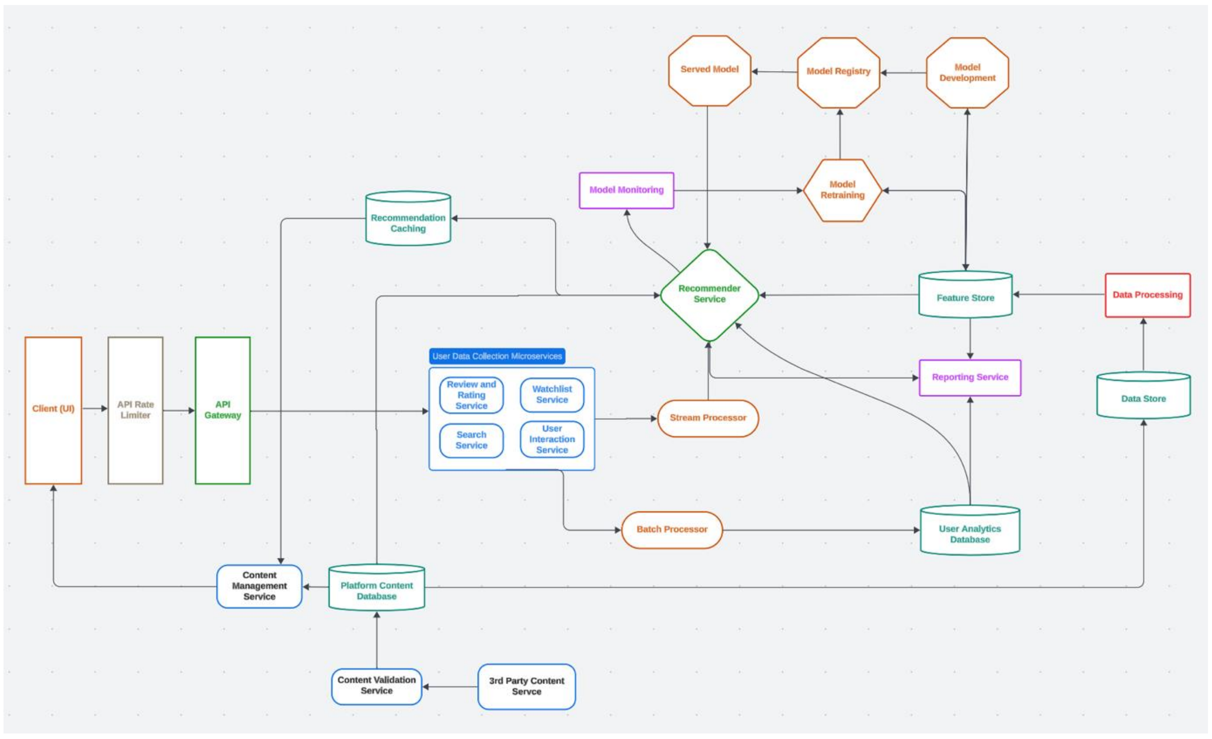
5. **Reliable Data Processing and Storage –** Collected user data must be processed, features must be extracted and stored. Data entering the system must pass validation. This is important to ensure the machine learning pipeline and the recommender service has the most exhaustive data set possible, maximising model proficiency and as a result recommendation accuracy.

6. **Scalable Platform –** The platform should scale seamlessly, maintaining low latency, whilst handling 100,000 active users using the platform. The data collection pipeline and the recommender service must maintain standard performance when under this level of traffic.

7. **Secure Private Platform –** Personalised user data be security stored and accessed using encryption protocols. The system must also comply with GDPR privacy regulations.

8. **System Monitoring and Reporting -** The platform must implement real-time model monitoring to track the accuracy and performance of deployed machine learning models. There must also be comprehensive system health monitoring to track latency, throughput, and errors for all the core services.

System Architecture



User Interface and API Management

The Client UI is the interface where users interact with the platform across multiple devices. Users navigate the site by searching for films, adding content to watchlists, providing ratings, and browsing personalised recommendations. The API Rate Limiter manages HTTP request traffic from users and protects the backend services from being overwhelmed from high volumes of requests, via request throttling. The API Rate Limiter will throttle at 1000 messages per minute, per user. The API Gateway manages the routing of requests to various microservices.

User Data Collection Microservices

There are four microservices collecting user data in real time.

- The Review and Rating Service - Collects user ratings (1-5 stars) and reviews for films and series. Sentiment analysis is run on review text. A total sentiment weighting is calculated from the users review and rating and a sentiment integer is passed to the recommender service tagged to the film that was reviewed.
- Watchlist Service - Manages the user’s watchlist, tracking content they intend to view. A stronger weighting is applied to less popular films added.
- Search Service - Logs user searches, as this provides insights into content preferences. The weighing of a search query is 20% of the weighing of a review/rating/watch list data point, however, aggregating searching over the user’s time of the platform, does expect to provide valuable data for our recommendation system.
- User Interaction Service - Tracks more subtle interactions, such as clicks, time spent on content pages, and browsing behaviour. This data will come in at a higher volume and in some cases may be arbitrary, therefore the recommender service weighs this data the least out of the four micro services.

To handle large-scale data, the system has a Stream Processor for real-time data processing, using Apache Kafka, and a Batch Processor for processing large datasets over time, using Apache Spark. The Batch Processor periodically updates the Data Store, ensuring that the data and features used in machine learning models are continuously refreshed.

Content Management

The content for the platform is managed through several services:

- Platform Content Database - Stores metadata for all available films and series on the platform, including descriptions, genres, cast information etc. This content is constantly updated to reflect the latest film releases.
- Content Management Service - Manages the content for the platform, passing the content to the client UI and also passing the data to the Recommender Service.
- Content Validation Service - Before new content is added to the Platform Content Database, it is validated through the Content Validation Service, ensuring quality and compliance with platform standards.
- The 3rd Party Content Service -The platform integrates content from third-party providers. New content such as a new film release or new peer reviewed review which will provide the system with an understanding of popular films. This content has to pass validation prior to entering the Platform Content Database.

Recommendation System

The Recommender Service generates personalised film suggestions for users by ingesting live data from the Data Collection Microservices via a Stream Processor. It accesses the User Analytics Database to gather a long-term history of user interactions, while the Platform Content Database ensures it has the latest film content available for recommendations. The Feature Store feeds data into the Recommender Service, holding a set of processed features from user interaction data, which reduces the computational load on the machine learning model by eliminating the need to extract features on the fly. The model processes this data, outputs film recommendations, which are then cached for faster response times, and served to the Client UI.

The recommendation system enhances personalization by combining Content-Based Filtering and Collaborative Filtering techniques. Content-based filtering analyzes attributes like genre, director, and cast, allowing the system to identify similar films based on their characteristics. Collaborative filtering leverages user similarity analysis, discovering latent features in the content and user interaction data. To ensure recommendations reflect current preferences, the system applies a decay to older interactions, gradually reducing their influence.

To further enhance recommendation accuracy, deep learning models, such as Multilayer Perceptrons (MLPs), are utilised to capture complex relationships between users and films through user and item embeddings. Additionally, sequence-based models like RNNs or LSTMs help track the evolution of user preferences over time, adjusting recommendations dynamically based on viewing patterns. This results in a higher degree of personalization by modeling temporal trends in user behavior.

In terms of Response Time Optimisation, pre-trained embeddings for films and users speed up the recommendation process by quickly matching user profiles with film attributes. Additionally, using models such as autoencoders to preprocess and compress large-scale interaction data allows faster retrieval of recommendations while maintaining accuracy, even as the platform scales. To support consistent performance, Redis caching stores frequently accessed recommendations for quick retrieval, with cache invalidation triggered by user actions and periodic refreshes for new popular content.

The system also addresses the Cold Start Problem through content-based techniques, which leverage metadata such as genres and descriptions to generate recommendations for newly added films. For new users, minimal initial data is used, relying on cross-domain recommendations to provide relevant suggestions. Meta-embedding models combine various data sources to produce robust recommendations even in sparse-data scenarios, ensuring a smooth experience for both new users and new content.

Recommendation Model Development and Maintenance

The model management pipeline ensures that the system continuously delivers accurate recommendations. The process begins with feature engineering, where raw user and content data are transformed into useful inputs for the machine learning models. The data pipeline is automated to update the Feature Store every 12 hours, ensuring data freshness. To fine-tune model performance, advanced hyperparameter optimisation techniques are employed.

The Model Registry tracks each model version, its features, and associated performance metrics. This registry enables version control and allows for A/B testing to compare new models against the current one, supporting data-driven deployment decisions.

The Model Monitoring Service tracks prediction quality in real-time, alerting the system to any feature drift that could lead to a decline in recommendation performance. If accuracy falls below 80%, the monitoring service triggers Model Retraining. The newly trained model is A/B tested against the current model, and if it achieves an accuracy above 80%, it is registered and deployed. If the new model consistently meets performance expectations over three days, it remains in production; otherwise, the system reverts to the previous model, balancing improvement with stability.

Reporting Service

The Reporting Service ensures there is comprehensive visibility into the platform's functionality. The User Analytics Database, the Feature Store and the Recommender System all feed into this service. The data is visialised using the dashboard Grafana. This service allows the business to track the behaviour of users activity, the features systematically being added and the core competence of the recommendation system. This service enables the business to make data-driven decisions to improve the recommendation systems through either manually adding features to the store, deciding on different user data to collect or to manually trigger a model training cycle.

Risks

Risk 1: General Model Accuracy Decline

The system risk - The platform's recommendation system can face a significant decline in its performance, meaning a low recommendation accuracy rate. This degradation could occur due to the system's inability to adapt to changes in user behaviour patterns. This may be driven by potential data pipeline issues that compromise the quality of input to the Recommender Service.

The impact - The impact of a decline in recommendation accuracy is that users will be recommended movies which do not suit their taste. This fundamentally leads to a decrease in user satisfaction, a lower user engagement and less users returning to the platform. Hence, this risk can have a material impact on the business achieving its key goals. There is a direct correlation between the recommendation system performance and business revenue, this makes this risk critical for the platform's long-term viability.

How to mitigate the risk - The platform implements a robust Model Monitoring and Retraining System to effectively manage this risk. This system continuously evaluates recommendation accuracy and automatically triggers retraining when performance metrics fall below an 80% threshold. To ensure the most accurate recommendations, the platform uses a comprehensive data pipeline where user interaction data flows through the Stream Processor in real time, giving the Recommendation System an up to date view of user behaviour.

The Feature Store is regularly updated with new film content to maintain current user and content features, while real-time feedback processing enables immediate refinement of recommendations based on user interactions. This risk is further mitigated through quick model deployment, supported by sophisticated monitoring systems and efficient rollback procedures. This approach ensures that long-term changes to the recommendation system are thoroughly evaluated through data analytics before implementation.

Risk 2: Cold Start Problem

The system risk - The cold start problem is a significant risk for the platform, particularly in providing effective recommendations for new users or when recommending new film content added to the platform, with little review or recommendation data. This risk is due to having insufficient data available to be able to recommend a film to a user.

The impact - This risk can lead to low engagement and retention rates for new users. The system may also exhibit bias towards recommending only popular or established content, limiting the discovery potential for users. This bias may lead to the user not deeming the platform intelligent enough, leading to a higher bounce rate of new users.

How to mitigate the risk - To address this risk, the platform uses smart onboarding and balanced recommendation strategies. When new users join the platform, they are surveyed upon signup, asking them about their favorite movies, genres, directions etc. The platform also uses user meta data such as age, to make initial recommendations. This provides the Recommender Service with a fairly good starting point to start recommending films that are close to the user's taste.

When recommending new content, the Recommender Service gives new movies a fair chance at being recommended. The system combines popular and less well known films in the recommendations and tracks how users respond to new content to quickly learn what works. The platform also uses movie metadata like genre, cast, and director to connect new films with similar ones that users already like. The platform also surveys new users after their third visit day on the platform, asking how they rate the recommendations quality and the system then uses this rating to optimise the recommendations. These surveys help build up the data the platform needs for increasingly precise recommendations over time.

Risk 3: Problems with Content Quality

The system risk - Poor or erroneous content is a critical risk that can lead to delivering poor recommendations to users. This risk can come from inadequate or erroneous data coming from third-party services. This will lead to the Platform Content Database being filled with bad data, which will be displayed to users and fed into the recommendation system.

The impact - These issues can severely reduce the system's ability to provide accurate content recommendations, leading to a decline in user satisfaction, decreasing the effectiveness of the platform and leading to less users returning to the platform.

How to mitigate the risk - The platform addresses these content related risks via a robust Content Validation Service. This service implements advanced data quality checks and automated content categorisation, this service is also supported by human oversight if the automated validation checks fail to pass the content into the platform. The Content Validation Service also employs a reconciliation mechanism to maintain consistency across different content sources.

Risk 4: Stream Processor Overload

The system risk - The Stream Processor handles real time user interaction data, which feeds into the Recommender Service to ensure timely and relevant recommendations. However, as the platform scales, the Stream Processor could become overloaded.

The impact - This overload can lead to delays in processing interaction data, causing the recommendation system to work with outdated information and potentially reducing recommendation accuracy. This misalignment may reduce user engagement, as the recommendations may not reflect users' current preferences or viewing patterns, which impacts user satisfaction and again leading to less users returning to the platform.

How to mitigate the risk - To prevent the overloading, the system could implement dynamic load balancing and partitioning strategies within the Stream Processor. The Stream Processor could then handle data in parallel across multiple nodes, distributing the processing load more evenly. Another option would be to integrate a Kubernetes infrastructure, which has auto scaling capabilities, which would allow the Stream Processor to automatically adjust its resources based on incoming data volume. During peak periods, extra processing nodes could be temporarily deployed to aid in the increased load, this would prevent delays in recommendation updates, and maintain a high recommendation accuracy and high user satisfaction.

Deployment Strategies

Deployment Strategy 1 - Server Side Deployment

In this deployment strategy, all ML and AI components are deployed on a centralised, high performance server infrastructure.

Cost Implications - A centralised architecture requires an initial investment in high performance servers, particularly for ML models that have a high computation cost due to the requirement of specialised hardware. This approach is cost effective in the long term as it reduces the need for multi device compatibility and simplifies maintenance by centralising updates and resources. Server maintenance costs are limited due to the ease of implementing server wide updates, which are rolled out without any interaction on the client side.

Technical Complexity - This approach simplifies the overall system complexity, as updates and bug fixes are deployed centrally. Maintenance is more manageable since all AI/ML components are on the server side, reducing the operational burden compared to models with distributed components.

Scalability - To ensure scalability for global users, this model benefits from auto scaling and global traffic management. CDNs (Content Delivery Networks) handle frequently accessed content closer to users, which lowers response times and server load while improving performance during traffic surges.

Monitoring - With all components on the server side, supervision and monitoring costs are optimised by using centralised tools such as Grafana and Prometheus. These tools provide real time alerts, reducing the need for extensive human intervention.

Latency - CDNs further minimise delays by caching content closer to users. Database indexing and optimised queries ensure efficient data retrieval, leading to page content response times to be sub one second.

Privacy - Centralised data processing leads to a strong data control, simplifying compliance with privacy regulations, such as GDPR. Centralisation also reduces potential exposure points, which enhances data security.

Deployment Strategy 2 - Client Side Deployment

In this client side deployment, some ML components are deployed directly on the user's device.

Cost Implications - Deployment on the client side lowers server and bandwidth costs but introduces significant initial development complexity due to the need for compatibility across a variety of devices. Ensuring a strong performance on different device configurations requires extra testing and optimisation, leading to increased development time and cost. Maintenance is complex as updates must be pushed to user devices, which may rely on the user updating their application, driving a relatively high maintenance cost.

Technical Complexity - This model is technically complex due to the need to manage multiple device environments, which can lead to inconsistencies in performance. Different bugs can occur on different devices. Hence, more resources are required to monitor and resolve these issues. This increases the time and effort required for quality assurance testing, as compatibility must be verified on each device type.

Scalability - Scalability is high in a client side design, as each device handles its own processing, offloading work from centralised servers.

Monitoring - Monitoring is more difficult and costly due to the lack of centralised data access. To achieve consistent performance monitoring, the system would need to rely on user feedback or require embedded monitoring software on each device.

Privacy - Privacy is strong in this deployment design, since data processing occurs locally, reducing the amount of data sent to the server. However, maintaining compliance with data regulations becomes more challenging, as user data is stored on numerous devices instead of a single controlled environment.

Latency - Processing on the client device can reduce latency, as there is no need to send data to a central server for processing. However, this performance gain depends on device capabilities. For example, lower powered devices may experience slower processing times, affecting user experience.

Comparative Analysis

Cost Implications – The server side requires higher upfront server costs, while the client side needs more development resources. The server side offers simplified, centralised maintenance, whereas client side involves ongoing device specific support. The server side is typically more cost effective due to centralised management

Technical Complexity – The server side simplifies development with a single environment, where the client side requires multi device compatibility. The server side enables straightforward updates and fixes whereas the client side requires troubleshooting, specific to a device. The server side testing is centralised where the client side requires extensive validation across devices.

Scalability – The server side leverages auto scaling and CDNs for consistent performance. Whereas the client side naturally distributed processing load but may face limitations on specific devices. The server side manages through infrastructure, where the client side depends on device capabilities.

Monitoring – The server side offers comprehensive visibility through centralised tools (Grafana, Prometheus), where there is a challenge to implement such monitoring due to device distribution.

Privacy – Centralised control simplifies compliance on the server side, but concentrates data. On the client side there is enhanced privacy through local processing but this complicates compliance monitoring. The server side has fewer exposure points whereas the client side reduces the risk in data transmission.

Latency – Latency is optimised through CDNs on the server side, whereas on the client side, local processing can lead to higher latency if running on a low performing device. The server side provides more predictable performance across users.

Data

A telemetry metric called the Recommendation Acceptance Rate (RAR) has been designed to monitor the recommendation system's performance. This metric leverages data from the existing User Data Collection microservices to measure how successfully the system’s recommendations lead to meaningful user engagement, providing a direct measure of recommendation relevance and accuracy. With this telemetry metric, the business can more likely achieve its goals in maximising user retention, increasing platform monetisation, and gaining a competitive technological edge in the industry.

Telemetry Metric Calculation

The Recommendation Acceptance Rate (RAR) measures the percentage of recommended films that users actually engage with meaningfully. This is calculated by combining data from the following Data Collection microservices:

- Review and Rating Service - A positive rating or review (4 or more stars) or positive sentiment analysis score from the review text is counted as a Positive Engagement.
- Watchlist Service - When a recommended film is added to the watchlist, this is counted as a Positive Engagement.
- User Interaction Service - Time spent on recommended film pages, click patterns and browsing behaviour around recommended content. These passive signals help validate active engagement with recommendations and once a threshold is reached then this is counted as a Positive Engagement.

The RAR is calculated daily for each user:

RAR = (Number of Recommendations with Positive Engagement) / (Total Number of Films Recommended) × 100

The business level RAR is the average of all users RARs

Thresholds and Monitoring

The system establishes three performance levels: Optimal (RAR 75 percent or higher), Warning (RAR between 50 and 75 percent), and Critical (RAR below 50 percent). If the RAR drops to the Warning level, the system triggers an alert to the ML team. At the Critical level, immediate alerts are sent, leading to automated model health checks and possible rollback triggers if manual review confirms sustained poor performance. This ensures that the platform’s recommendation quality remains high, supporting user satisfaction and engagement.

Data Volume Analysis

With a user base of 100,000 active users, the system processes approximately 1GB of data daily through the existing microservices:

- Review and Rating Service - Generates about 50MB daily (10MB ratings, 40MB reviews)
- Watchlist Service - Generates 50MB daily
- Search Service - Generates 200MB daily
- User Interaction Service - Generates 700MB daily

Storage Requirements

The system maintains a tiered storage approach for RAR data:

- Hot Storage - Retains seven days of full resolution data (7GB), enabling detailed recent analysis of recommendation performance.
- Warm Storage - Keeps 30 days of aggregated hourly metrics (40GB), allowing medium term trend analysis.
- Cold Storage - Maintains 365 days of daily aggregates (500GB), supporting long term performance evaluation and seasonal pattern analysis.

Data is automatically deleted from each storage tier after the retention period expires, ensuring compliance with GDPR data privacy regulations while managing storage costs efficiently.

False Positive and Negative Detection
The system leverages data from the microservices to identify False Positives /Negatives in recommendations. False Positives are detected through low ratings or negative sentiment from the Review and Rating Service, removal from watchlists, and minimal interaction time with recommended content from the User Interaction Service. The system is expected to achieve approximately a 80 percent accuracy in detecting false positives, calculated based on model feedback loops and user correction inputs. False Negatives are identified through the Search Service and User Interaction Service, which show when users discover and engage positively with content similar to their preferences that wasn't recommended. High ratings for self discovered content in the Review and Rating Service help confirm these cases. The system is expected to achieve approximately a 70 percent accuracy in detecting False Negatives.

Implementation
The platform implements real time RAR calculation through the existing Stream Processor, with continuous monitoring via dashboards for visualisation. The RAR metric data flows through the Batch Processor, enabling aggregated metrics for trend analysis and monthly model performance reviews. This approach ensures both immediate detection of recommendation issues and long term improvement of the recommendation system while maintaining manageable data volumes and processing requirements.

References

1. 3 Kiran, U., Hassan, M.U., Javed, A., Siddique, U. and Ahmad, S. (2023) 'A comprehensive survey of recommender systems based on deep learning', - https://www.mdpi.com/2076-3417/13/20/11378

2. 4 Lehmann, J., Lalmas, M., Yom-Tov, E. and Dupret, G. (2012) 'Models of user engagement' (164–175) - https://link.springer.com/chapter/10.1007/978-3-642-31454-4_14

3. 5 Shani, G. and Gunawardana, A. (2011) 'Evaluating recommendation systems' - https://www.researchgate.net/publication/226264572_Evaluating_Recommendation_Systems

4. 7 shang, S., Yao, L., Sun, A. and Tay, Y. (2017) 'Deep learning based recommender system: A survey and new perspectives' - https://www.researchgate.net/publication/318671349_Deep_Learning_Based_Recommender_System_A_Survey_ and_New_Perspectives

5. Charlieinden.com - https://charlieinden.github.io/System-Design/2021-01-25_Microservices--Designing-Highly-Scalable-Systems-55dbb6f64c94.html

6. Guy Hardonag (2024) ‘Machine Learning Architecture: What it is, Key Components & Types’ - https://lakefs.io/blog/machine-learning-architecture/

7. Stephen Oladele (2024) ‘MLOps Architecture Guide’ - https://neptune.ai/blog/mlops-architecture-guide

8. Hasan, E., Rahman, M., Ding, C., Huang, J. X., & Rasa, S. (2024). Review-based recommender systems: A survey of approaches, challenges and future perspectives. arXiv. - https://arxiv.org/abs/2405.05562

9. Kasmierczak, J., Salama, K., Huerta, V., & Jang Bahadur, S. K. (n.d.). MLOps: Continuous delivery and automation pipelines in machine learning. - https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

10. Park, J., Naumov, M., Basu, P., Deng, S., Kalaiah, A., Khudia, D., Law, J., Malani, P., Malevich, A., Nadathur, S., Pino, J., Schatz, M., Sidorov, A., Sivakumar, V., Tulloch, A., Wang, X., Wu, Y., Yuen, H., Diril, U., Hazelwood, K., Jia, B., Jia, Y., Qiao, L., Rao, V., Rotem, N., Yoo, S., & Smelyanskiy, M. (2018) - https://research.facebook.com/publications/deep-learning-inference-in-facebook-data-centers-characterization-performance-optimizations-and-hardware-implications/

11. Sculley, D., Holt, G., Golovin, D., & Dennison, D., et al. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems. - https://www.researchgate.net/publication/319769912_Hidden_Technical_Debt_in_Machine_Learning_Systems

12. Tatbul, N., Lee, T. J., Zdonik, S., et al. (2018). ‘Precision and recall for time series’ - https://proceedings.neurips.cc/paper/2018/file/8f468c873a32bb0619eaeb2050ba45d1-Paper.pdf