Project Report :

Milestone 4:

Fairness, Security, and Feedback Loops

Github Link :

https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML

Ameya Morbale
Ankit Kumar
Prakhar Pradeep
Praveen Ramesh
Kedi Xu

Fairness

Fairness Requirements

Fairness is a critical concern in developing machine learning models, particularly in domains where decisions made by these models can have significant impacts on people's lives. We used a thorough process to determine the fairness requirements for our movie recommendation system, one that involved understanding potential harms to fairness, investigating bias in data, identifying protected attributes, resolving conflicts between competing fairness goals, and taking into account how fairness interacts with other system goals.The following is an overview of it:

a) **Understanding Potential Fairness Harms**
   One potential issue that our movie recommendation system may face is the possibility of perpetuating stereotypes and biases in its recommendations. This can result in the exclusion or unequal treatment of certain user groups. For example, if the system only suggests action movies to male users and romantic comedies to female users, it could reinforce gender stereotypes and limit users' options. Additionally, age and occupation are two other protected attributes that may be impacted by biased recommendations. If the system only recommends children's movies to younger users and dramas to older users, it could perpetuate age-related stereotypes and limit the users' choices. Similarly, if it only suggests business-related movies to users with certain occupations, it could result in occupational discrimination and limit the variety
b) **Exploring Bias in data and the sources of such bias**
   To ensure the development of a fair movie recommendation system, exploring potential bias in data is a critical step. Data bias can originate from a range of sources such as inaccurate sampling, measurement errors, and other factors. For example, a bias may exist if the training data disproportionately represents certain groups or if user ratings are influenced by factors such as age, gender, or race. Also, one age group may tend to overrate "average" movies, while other groups may be more critical in their assessments. By examining data sources and identifying potential sources of bias, we can build a more equitable recommendation system that provides unbiased and personalized movie recommendations for all users, regardless of their demographic attributes.
c) **Identifying Protected Attributes**
   Identifying protected attributes is a critical step in developing a fair and inclusive movie recommendation system. Protected attributes may include race, ethnicity, religion, gender, sexual orientation, age, disability, and other characteristics that may be protected under anti-discrimination policies, depending on the jurisdiction. To ensure that the recommendation system does not unfairly discriminate against users based on these attributes, we avoid using them in the recommendation algorithm. By disregarding these protected attributes in our recommendations, we can create a system that provides personalized and relevant movie suggestions to all users, irrespective of their demographic characteristics. This approach ensures that our recommendation system is fair and equitable, promoting an inclusive user experience for everyone.

d) **Negotiating conflicting fairness goals**
Negotiating conflicting fairness goals is an essential part of developing a fair movie recommendation system. Conflicting fairness goals may arise when certain groups of users have different preferences or when the system aims to provide equal opportunities for all users while also providing personalized recommendations. For example, if the system recommends only popular movies to all users, it may disadvantage niche or lesser-known movies that appeal to specific groups of users. In such cases, we need to balance the competing goals and ensure that the recommendation system provides a fair and inclusive user experience for all users.In negotiating conflicting fairness goals, incorporating user feedback and utilizing multiple recommendation algorithms, such as collaborative and content-based filtering, can help provide personalized and diverse movie recommendations.

e) **Considering how fairness interacts with other system goals(e.g. profits)**
Developing a movie recommendation system requires balancing fairness with other system goals, such as profits. A user-centric approach that prioritizes user satisfaction while considering business objectives can help achieve this balance. Providing personalized recommendations that align with user preferences and interests can improve engagement and revenue. A tiered pricing model can offer premium features without compromising fairness and inclusivity. Achieving a balance between fairness and business objectives involves careful consideration of user needs, market dynamics, and profitability. By adopting such an approach, we can develop a fair, inclusive, and profitable movie recommendation system that caters to the diverse needs of all users.

System fairness requirement:
Our system must ensure a varied selection of movie recommendations a fair system must encompass a wide range of genres, to avoid genre-discrimination

Measure: Frequency counts of recommended movies
Data: We assessed the diversity of recommendations using movie metadata, specifically the genre types of the top-recommended movies from the Kafka stream.
Operationalization: To put this requirement into practice, we evaluated the genre distribution among the top-recommended movies

Recommendation model fairness requirement:
A fair model should maintain impartiality and avoid favoritism towards users on the basis of protected attributes such as age, gender, and occupation.

Measure: Frequency counts of recommended genres
Data: We use the online metric – Percentage of Users Watching Recommended Movies (recommendation quality referenced in Milestone 2 & 3) to evaluate fairness across different genders, age groups, and occupations.
Operationalization: We compare the metric within each group to the overall metric to determine the fairness of the movie recommendation model.

Fairness Improvement Suggestions

To improve fairness in the movie recommendation system, we could implement several measures focused on the two fairness requirements. Firstly, to improve the varied selection of movie recommendations, we could employ fairer data collection practices. This could involve ensuring a diverse range of movies are included in the training data to avoid genre-discrimination. We could also consider incorporating user feedback on recommended movies to better understand their preferences and improve the diversity of recommendations.

Secondly, to maintain impartiality and avoid favoritism towards users on the basis of protected attributes, we could implement system design to mitigate bias. For instance, we could adopt a hybrid recommendation approach that combines collaborative filtering and content-based filtering to provide a more diverse and personalized set of recommendations, while also avoiding demographic biases. Additionally, we could evaluate the fairness of our recommendation model by monitoring the frequency counts of recommended genres across different gender, age groups, and occupations, and compare the results within each group to the overall metric.

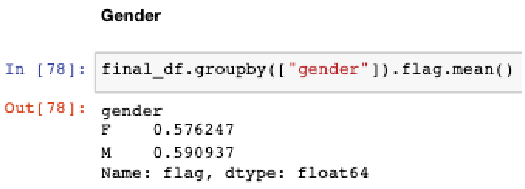
Finally, we could integrate the process of monitoring and operation into the development process to ensure continuous fairness improvement. This could involve regularly auditing the recommendation system to identify any potential biases and taking corrective actions, such as updating the training data, improving data preprocessing techniques, or refining the recommendation algorithms. By adopting such practices, we can continually improve the fairness of our movie recommendation system, providing users with diverse and equitable movie recommendations that cater to their preferences, without compromising on system profitability.

Fairness Analysis

In this fairness analysis, we examine whether our AI-based movie recommendation system meets the requirements for fairness across different user demographics such as gender, age, and occupation. We evaluated our model's performance using the root mean squared error (RMSE) metric across these demographic groups to assess model fairness. Furthermore, we also analyzed system fairness by investigating the distribution of recommended movies and genres.

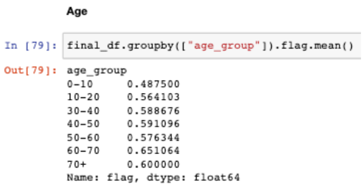
Model Fairness (Requirement 1)

We analyzed the RMSE across gender, age, and occupation groups to assess whether our recommendation system provides fair predictions for all users. Our analysis revealed fairly consistent RMSE values for these demographic groups, indicating that our model performs similarly across different user profiles. This suggests that our model does not inherently favor specific demographic groups. Following are results attached for RMSE across age & gender :


Gender

```
In [78]: final_df.groupby(["gender"]).flag.mean()

Out[78]: gender
         F    0.576247
         M    0.590937
         Name: flag, dtype: float64
```


Age

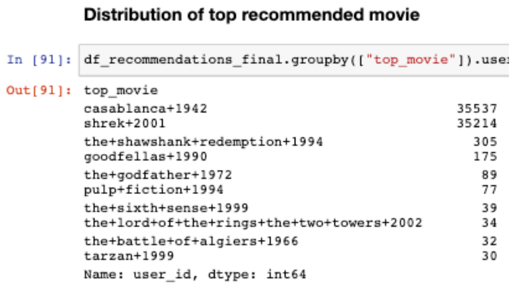
```
In [79]: final_df.groupby(["age_group"]).flag.mean()

Out[79]: age_group
         0-10     0.487500
         10-20    0.564103
         30-40    0.588676
         40-50    0.591096
         50-60    0.576344
         60-70    0.651064
         70+      0.600000
         Name: flag, dtype: float64
```

Github Link:https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/fairness_analysis.ipynb
(under heading Model Fairness)

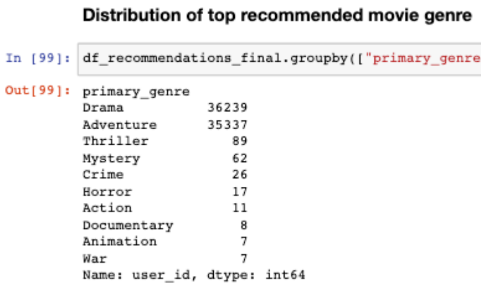
System Fairness (Requirement 2)

While our model demonstrated fairness in its performance, we observed a different outcome in the system's recommendations. Our analysis showed that certain movies and genres were disproportionately recommended compared to others. This indicates that our system might unintentionally prioritize specific content, thereby limiting the diversity of recommendations for users.


Distribution of top recommended movie

```
In [91]: df_recommendations_final.groupby(["top_movie"]).use

Out[91]: top_movie
         casablanca+1942                                35537
         shrek+2001                                     35214
         the+shawshank+redemption+1994                    305
         goodfellas+1990                                  175
         the+godfather+1972                                89
         pulp+fiction+1994                                 77
         the+sixth+sense+1999                              39
         the+lord+of+the+rings+the+two+towers+2002         34
         the+battle+of+algiers+1966                        32
         tarzan+1999                                       30
         Name: user_id, dtype: int64
```


Distribution of top recommended movie genre

```
In [99]: df_recommendations_final.groupby(["primary_genre

Out[99]: primary_genre
         Drama          36239
         Adventure      35337
         Thriller          89
         Mystery           62
         Crime             26
         Horror            17
         Action            11
         Documentary        8
         Animation          7
         War                7
         Name: user_id, dtype: int64
```

Github Link: https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/fairness_analysis.ipynb
(under heading System Fairness)

Conclusion & Negative Findings

Despite achieving model fairness, our system still exhibits some biases in the distribution of movie and genre recommendations. These negative findings present valuable insights and opportunities for further improvement in our recommendation system. By addressing these biases, we can enhance the overall fairness of our system and provide a more diverse and inclusive selection of movie recommendations for all users.

Our fairness analysis reveals that our recommendation system demonstrates fairness in model performance across different demographic groups. However, there is room for improvement in system fairness, as certain movies and genres are disproportionately recommended. By addressing these negative findings and refining our recommendation system, we aim to provide a more equitable and inclusive experience for all users.

Feedback Loops

Anticipating feedback loops **(2 page max):**

To anticipate possible feedback loops in a movie recommendation system, we used the world-vs-machine framework, which involves considering the assumptions made about the environment and how the system outputs (recommendations) interact with inputs (telemetry, ratings). This approach helps identify potential issues that may arise in different conditions or populations. Here's a description of the process:

1. **Identify assumptions:** The assumptions made about the environment, such as user behavior, input data, and how users interact with the systems, are:

   A. **User Behaviour**: We assume that users provide honest feedback, their preferences remain relatively stable over time, and the data we collect represent their true interests. Users are eager to rate or comment on films they have viewed since they have a certain amount of interest in them. Additionally, based on their prior movie-watching experiences, users can have specific biases or inclinations.

   B. **User Data:** The program has access to a vast library of movies and the metadata linked to them, such as the genre, cast, director, release date, and ratings. Additionally, user-generated information such as movie ratings, reviews, and watch history is accessible to the system.

   C. **User Interaction:** Users communicate with the system by entering data such as reviews, comments, and search requests. The system uses this information to customize recommendations and raise their accuracy over time. Users may also receive more information from the system, such as trailers, summaries, and related suggestion**s.**

2. **Analyze interactions between outputs and inputs:** The outputs of a movie recommendation system, i.e., the recommended movies, can interact with the inputs (telemetry, ratings) in several ways. Here are some potential interactions:

   A. **Feedback loop**: The system's outputs may affect user choices and ratings of movies, which may then give the system input on how to make better recommendations. For instance, if the system suggests a movie that the user likes, they are more inclined to score it highly, which aids the system in learning more about the user's interests and making better recommendations in the future.

   B. **Personalization:** Based on the user's input data, including telemetry and ratings, the system's outputs can be tailored to their needs. The system can use this information to build a user profile and suggest movies more likely to suit the user's tastes and interests.

   C. **Bias:** Biases in the incoming data may be reinforced by the system's outputs. For instance, if the algorithm only suggests well-known films, it can reinforce the popularity bias and ignore lesser-known films that the user might like. The system can use algorithms that take a more comprehensive range of input data into account and generate more varied recommendations to lessen this.

3. **Assess assumptions under varying conditions and populations:** It's crucial to evaluate the presumptions made for a movie recommendation system under various circumstances and populations to ensure the system gives all consumers accurate and helpful recommendations. Here are some things to think about:

A. **Varying levels of engagement:** The presumptions made for a movie recommendation system might not apply to users who engage with the site to varied degrees. For instance, people who are more active on the site and leave more ratings and comments may get recommendations that are more accurate than less active users. To improve suggestion accuracy, the system should be built to take into account the level of user engagement and combine data from both highly and moderately engaged users.

B. **User demographics:** Not all user demographics may fit the assumptions made for a movie recommendation system. Users from other nations or cultures may have varied interests in specific genres, as evidenced by the fact that elderly users may have different movie preferences than younger users. To increase suggestion accuracy, the system should be created with the variety of its user base in mind and incorporate user data from a wide range of demographics.

C. **Cultural differences:** It's possible that people from other cultures won't agree with the assumptions made for a movie recommendation system. Users from various cultures, for instance, could have various expectations and preferences for movie recommendations. To increase suggestion accuracy, the system should be created with the cultural variety of its user base in mind and incorporate data from users with various cultural backgrounds.

4. **Identify potential feedback loops with negative consequences**: Based on the analysis, Two possible feedback loops with negative consequences that I can anticipate are:

1. **Echo chamber effect:** If the recommendation system becomes overly focused on reinforcing users' existing preferences, it may create an echo chamber where users are only exposed to content that aligns with their current interests. This could result in users missing out on new or diverse content, ultimately leading to a less satisfying experience. The world-vs-machine assumption that user preferences remain stable over time might not hold true, as people's tastes can evolve or they might enjoy occasional surprises.

2. **Overfitting to a specific user group:** If the input data is biased towards certain user groups (e.g., specific age groups or geographic locations), the recommendation system might perform poorly for other users. This could lead to a negative feedback loop where users from underrepresented groups become dissatisfied with the recommendations, provide negative feedback, and eventually disengage from the platform. The assumption that the data collected is representative of all users' true interests might not be accurate under all conditions or for all populations.

- ***Mitigating a negative feedback loop* (0.5 pages max):** To mitigate potential negative effects of both anticipated feedback loops, the following changes can be made to the system:

  1. Echo chamber effect:

  a. Diversify recommendations: Introduce a diversity factor in the recommendation algorithm to ensure that users are exposed to a variety of content, including movies that may be slightly outside their current preferences. This will allow users to discover new content and prevent the formation of echo chambers.

  b. Periodic exploration: Regularly include exploration phases where the system recommends movies with a broader range of attributes or from less explored genres. This will enable users to encounter different content types and potentially expand their preferences.

  c. User-controlled recommendations: Allow users to control the degree of diversity in their recommendations, enabling them to choose between a more personalized or a more exploratory experience.

  2. Overfitting to a specific user group:

  a. Stratified sampling: When collecting data, ensure that it is representative of the entire user base by using stratified sampling techniques. This involves dividing the user population into different strata (e.g., age groups, geographic locations) and sampling proportionally from each stratum to avoid biases.

  b. Fairness-aware recommendation algorithms: Implement fairness-aware recommendation algorithms that aim to balance the trade-off between personalization and fairness. These algorithms ensure that different user groups receive equally relevant recommendations and reduce the risk of overfitting to a specific user group.

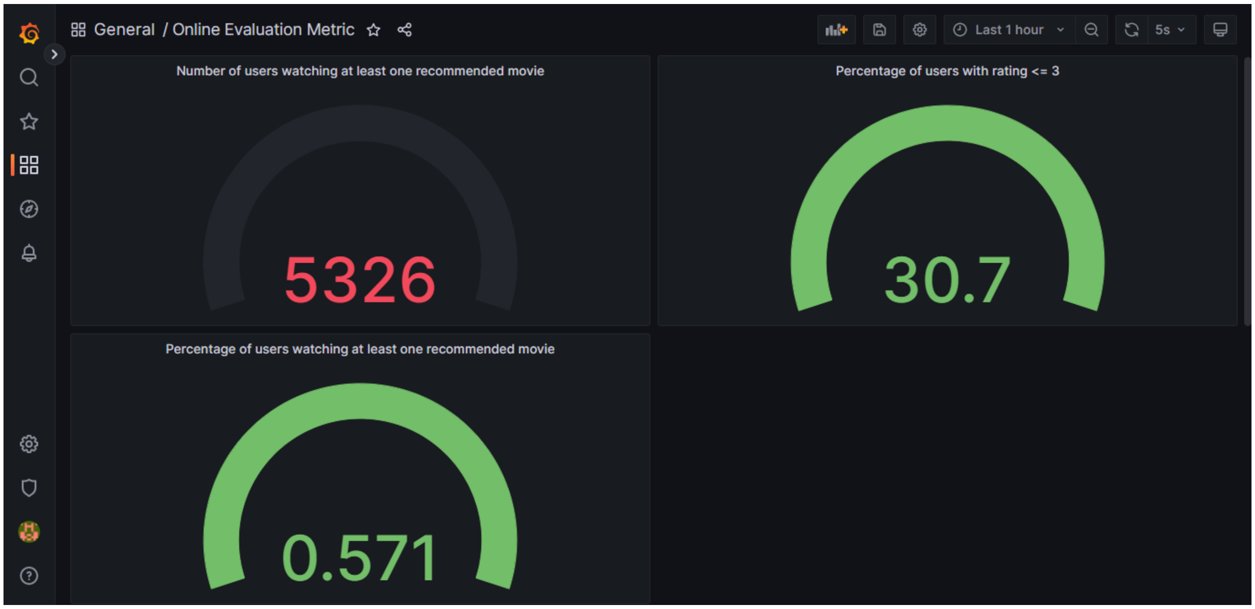
  c. Continuous evaluation and monitoring: Regularly evaluate the recommendation system's performance across different user groups and conditions. Monitor for any emerging biases or underperformance for specific user segments, and update the recommendation model accordingly to address any observed issues.

  By incorporating these changes, the system can mitigate the negative effects of both anticipated feedback loops, ensuring a better user experience and improving the overall performance of the movie recommendation system.

Analysis of a feedback loop

We analyze whether the feedback loop of the echo chamber exists. To detect this, we first add a table in our timescale database to store user ratings from kafka stream (https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/kafka_telemetry_rate.py), and create a new grafana panel that monitors how many percent of users always give low rating (<=3). We find that around 30% of the users gave scores that are less than 4. These users are worth exploring to see what leads to the low score.

The top right grafana panel is the newly added one showing low rating:



Number of users watching at least one recommended movie
5326

Percentage of users with rating <= 3
30.7

Percentage of users watching at least one recommended movie
0.571

By querying the timescale db, we got the user ids of those users that continue to give low ratings more than 5 times within the last three days. Next, we query the recommendation table, to see how many of the recommended movies to these users are duplicated (the query code https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/duplicate_recommendation.sql).

We find that the percentage is 87%, which shows that a large percentage of movies are recommended more than once even when the users give low ratings to it. This showed the echo chamber effect, where there is a lack of diversity in our recommendation, and users are giving low scores because they get tired of it.

Security

Security Issues and mitigation strategies

To determine security issues in the movie recommendation system, we carried out a more in-depth analysis during the threat modeling process. Here's a deeper look at the steps we followed:

- Identify assets: We began by identifying critical assets that need protection within the system. This included user data (personal information, viewing history, preferences), the recommendation model itself, and the underlying infrastructure (servers, databases, network connections).

- Create a system model: To understand how the different components interact and exchange data, we mapped out the system architecture. This included creating diagrams that show data flows between components, such as user interfaces, APIs, databases, and the recommendation engine. This step allowed us to identify potential points of entry for attackers and better understand the attack surface.

- Identify threats: We brainstormed potential attack vectors, adversaries, and their capabilities. We considered internal and external threat actors, such as cybercriminals, competitors, or malicious insiders. We examined how they could exploit vulnerabilities to achieve their objectives, such as stealing user data, manipulating recommendations, or disrupting the service.

- Assess vulnerabilities: We evaluated the existing security measures and controls in the system to identify potential weaknesses that could be exploited by threats. This involved reviewing the authentication and authorization mechanisms, input validation, encryption, and secure coding practices, among others.

- Prioritize threats: After identifying the threats and vulnerabilities, we ranked them based on their likelihood of occurrence and potential impact on the system. This helped us focus on the most pressing security issues and allocate resources accordingly.

During the threat modeling process, we conducted a thorough examination of the movie recommendation system and held discussions with key stakeholders, such as developers, security experts, and system administrators. We also reviewed relevant documentation, such as system architecture diagrams, data flow diagrams, and security policies, to gain a comprehensive understanding of the system.

By following this structured approach, we were able to identify and prioritize security issues in the movie recommendation system. Two of the most plausible security issues we found were:

- Model poisoning attack on the recommendation model: Attackers might try to manipulate the recommendation model by injecting biased or malicious data into the training dataset, causing the recommendation engine to provide inaccurate or inappropriate suggestions.

- Unauthorized access to user data: Attackers could exploit vulnerabilities in the system to gain unauthorized access to sensitive user data, such as personal information, viewing history, or preferences, leading to privacy violations and potential misuse of user data.

By thoroughly analyzing the system and following the threat modeling process, we were able to determine the most pressing security issues, develop mitigation strategies, and enhance the overall security posture of the movie recommendation system.

Security Issue 1 - Model Poisoning Attack on the Recommendation Model

An attacker might attempt to manipulate the recommendation model by injecting biased or malicious data into the training dataset. This can result in the recommendation engine providing inaccurate or inappropriate suggestions, thereby degrading the system's performance and user experience. Evidence for this analysis includes examining the data ingestion process and evaluating the possibility of an attacker tampering with the training data.

Mitigation Strategies:

- Input validation and data sanitization: Implement strict validation and sanitization processes for data ingested into the system. This can help detect and reject malicious or biased data before it affects the recommendation model.

- Anomaly detection and monitoring: Continuously monitor the training data and system performance for anomalies or unexpected patterns, which may indicate manipulation attempts. Use machine learning techniques or statistical analysis to detect deviations from normal behavior.

- Secure data sources: Ensure that the sources from which you obtain training data are trustworthy and secure. Verify the integrity and authenticity of data obtained from external sources.

- Access control: Implement strong access control mechanisms to prevent unauthorized users from tampering with the training data or the recommendation model. Limit the number of users with write access to the data and restrict access to the model itself.

- Differential privacy: Incorporate differential privacy techniques into the data processing pipeline to protect user data while still allowing it to be used for training purposes. This can help prevent attackers from inferring sensitive information about individual users from the aggregated data.

- Model robustness and adversarial training: Train the recommendation model to be more robust against adversarial examples or poisoning attacks by including adversarial samples in the training data or using techniques like adversarial training or defensive distillation.

Security Issue 2 - Unauthorized Access to User Data

An attacker could exploit a vulnerability in the system to gain unauthorized access to sensitive user data, such as personal information, viewing history, or preferences. This can lead to privacy violations and potential misuse of user data. Evidence for this analysis includes reviewing the system's authentication and access control mechanisms, as well as evaluating the security of APIs and data storage.

Mitigation Strategies:

- Strong authentication: Implement robust authentication mechanisms, such as multi-factor authentication (MFA), to ensure that only authorized users can access the system and user data.

- Access control: Use role-based access control (RBAC) or attribute-based access control (ABAC) to limit access to sensitive user data based on a user's role, attributes, or responsibilities within the organization.

- Encryption: Encrypt sensitive user data both at rest (in storage) and in transit (during communication between system components) using strong encryption algorithms and key management practices.

- API security: Secure the APIs used for accessing user data by implementing proper authentication, authorization, input validation, and rate limiting. Regularly audit and monitor API usage for signs of abuse or unauthorized access.

- Secure coding practices: Follow secure coding practices and guidelines, such as the OWASP Top Ten Project, to minimize the introduction of vulnerabilities in the system during development.

- Regular security audits and vulnerability assessments: Conduct periodic security audits, vulnerability assessments, and penetration tests to identify and fix potential weaknesses in the system. Keep up-to-date with security patches and updates for all software components and dependencies.

Analysis of a security issue

To analyze whether there were any attacks against our movie recommendation system with regard to one of the previously discussed security issues, we chose the model poisoning attack as our focus. We used telemetry data collected from the system to assess if there were any signs of such an attack.

The analysis process included the following steps:

- Define indicators of compromise (IoCs): We identified patterns or behaviors that could indicate a model poisoning attack, such as unusual spikes in data ingestion, unexpected changes in model performance, or anomalous user activity.

- Collect and preprocess telemetry data: We gathered telemetry data from various sources, such as data ingestion logs, model training logs, system performance metrics, and user activity logs. We preprocessed the data to filter out noise and irrelevant information.

- Analyze telemetry data: We used statistical methods, machine learning techniques, and data visualization tools to analyze the telemetry data, looking for patterns or anomalies that could indicate a model poisoning attack.

- Investigate anomalies: When we identified potential anomalies, we investigated them further to determine whether they were false positives or actual signs of an attack. This involved cross-referencing the anomalies with other data sources and assessing their potential impact on the system.

Key findings:

- Negative results: Our analysis did not reveal any clear evidence of a model poisoning attack against our recommendation system. We observed some minor anomalies in the data ingestion logs and user activity logs, but further investigation indicated that these were likely due to normal fluctuations and not indicative of an attack.

- System performance: We found that the overall performance of the recommendation model remained stable, without any significant or unexpected changes that could be attributed to a model poisoning attack.

- Monitoring effectiveness: Our analysis demonstrated the effectiveness of using telemetry data to monitor the system for potential security issues. The combination of data sources and analysis techniques allowed us to gain insights into the system's behavior and identify areas for improvement in our monitoring and alerting processes.

In conclusion, our analysis of the telemetry data did not reveal any observable attacks against the movie recommendation system related to the model poisoning issue. However, this exercise helped us understand the value of continuously monitoring and analyzing telemetry data to proactively detect and mitigate potential security threats. It also highlighted the importance of refining our monitoring and alerting mechanisms to minimize false positives and improve the overall security posture of our system.

CODE:
https://github.com/cmu-seai/group-project-s23-The-hangover-Part-ML/blob/main/security_analysis.py