Week 12: Model Monitoring

Yujia Cao, Wendi Yuan, Yuhan Zhao

Based on last week's research, we identified the model's explanation and conducted a potential risk analysis. This week, we focused on preparing the model for deployment and developing a monitoring plan tailored to the model's requirements, aligned with our research objectives.

We successfully serialized our trained model using Pickle, saving it as a .pkl file for future use. Additionally, we used Pickle to save and load our datasets, ensuring consistency and reusability in our workflow. All team members use macOS systems and deploy our Python environment via Anaconda, running Jupyter Notebook (Appx. 3). A comprehensive list of the packages used in this project is also provided (Appx. 4).

Based on our research and the characteristics of our topic, YouTube video performance, we plan to deploy our model primarily in batch mode, with optional consideration for real-time inference. Batch mode aligns best with our dataset, which spans 2017–2018 and includes video performance metrics and audience engagement data collected over time, making it naturally suited for periodic bulk processing. Batch processing allows for the efficient handling of large datasets, accommodating high-latency workflows where results are processed collectively after all data is analyzed. Features such as sentiment and engagement metrics often require substantial preprocessing, which is more resource-efficient and easier to manage in bulk. Additionally, batch mode enables the use of comprehensive datasets for predictions, enhancing accuracy, especially since our model was trained on historical data without access to real-time inputs. This approach is also simpler to implement and maintain, making it ideal for periodic evaluations and analysis. Given our objective of evaluating video performance over a period of time to provide content creators with actionable insights for improving engagement, batch processing is more effective than real-time inference, which would focus on instantaneous feedback that is neither necessary for this use case nor conducive to thoughtful video production. While real-time inference could be useful for immediate optimizations, such as suggesting improvements during video uploads, it would require additional infrastructure for real-time data processing and low-latency predictions, which are beyond the scope of our current research objectives and less relevant to content creators' workflows.

When deploying our YouTube engagement prediction model, it is essential to select and monitor appropriate performance metrics to ensure the model performs effectively in production and aligns with business objectives. These metrics evaluate the accuracy, quality, and reliability of the model's predictions

over time. For instance, Root Mean Squared Error (RMSE) measures the average error between predicted and actual engagement metrics, providing a clear assessment of overall prediction accuracy. A rising RMSE would indicate declining model performance, signaling the need for investigation or retraining. This metric is particularly critical for tracking outcomes such as *days_to_trend*, where a drop in accuracy suggests the model's predictions no longer reflect reality. Similarly, Mean Absolute Error (MAE) quantifies the average deviation of predictions from actual values, offering a straightforward evaluation of prediction precision. A low MAE indicates that the model closely aligns with the true outcomes, making it a valuable metric for monitoring how well *days_to_trend* predictions approximate the real values. Together, these metrics provide a robust framework for maintaining model integrity and ensuring that predictions remain actionable and relevant in production.

Business metrics serve as a critical link between model performance and real-world business outcomes, assessing how well the model's outputs align with objectives such as increasing engagement and optimizing content strategy. These metrics validate whether the model adds value by positively influencing creator strategies and audience interactions. For instance, engagement uplift monitors whether predictions and recommendations drive improvements in key metrics such as views, likes, and comment counts. These indicators are directly tied to a video's success, as they reflect how well the model's predictions match actual user behavior. Discrepancies, such as high predicted likes but low real-time engagement, may signal issues with the model's accuracy or relevance. Tracking metrics like rank and trending score are equally important, as these scores determine how a video is promoted on the platform, such as being featured on the trending page. Misalignment between predicted and actual rankings indicates that the model may not fully capture the factors driving video visibility and reach, which are key to business goals. Additionally, gathering feedback from content creators on the model's insights—such as their actionability, usefulness, and timeliness—helps improve adoption rates and ensures creators benefit from the system, ultimately enhancing user satisfaction. Another critical metric is the Conversion Rate or Click-Through Rate (CTR), which measures how well the model predicts a video's ability to generate views. A higher CTR indicates that the model effectively identifies content likely to attract user engagement, aligning with the platform's core goal of maximizing audience interaction. Together, these business metrics provide a comprehensive framework for monitoring the model's impact on YouTube's strategic objectives and ensuring its relevance and value in production.

Operational metrics are equally important to ensure that the infrastructure and processes supporting the model are efficient, reliable, and well-suited for a production environment. These metrics monitor deployment performance and help maintain a smooth and responsive system. One critical metric is latency, which measures the time taken to generate predictions. This is particularly vital in real-time

applications, where high latency can disrupt business operations or negatively impact the user experience. Even in batch processing, excessive delays could hinder timely content creation and promotion decisions. For instance, predicting video trends quickly is essential for creators to adapt and optimize their strategies, and any lag in prediction could compromise these efforts. Throughput is another key operational metric, tracking the number of predictions the model can process within a given timeframe. This ensures that the model can handle the anticipated volume of video data in production. For a platform like YouTube, where sudden surges in video uploads are common, insufficient throughput could overwhelm the system, leading to delays or dropped predictions. Lastly, system availability and uptime measure the percentage of time the system is functional and accessible. High availability ensures that predictions are consistently delivered to stakeholders without interruptions, supporting reliable decision-making. These operational metrics collectively help maintain the performance and scalability of the model in a production environment, safeguarding its effectiveness and user satisfaction.

Monitoring all performance, business, and operational metrics is crucial to ensuring the success of the model in production and maintaining alignment with strategic goals. Tracking model performance metrics, such as RMSE, R^2, and drift monitoring, ensures the model continues to perform reliably as data or user behavior evolves. Poor performance in these areas can directly impact the accuracy of predictions, such as when a video might trend, leading to suboptimal content recommendations and misjudgments about video popularity. Monitoring business outcomes connects model performance to tangible results, such as engagement or revenue. A model that fails to align with user behavior or platform trends risks generating inaccurate recommendations, leading to poor engagement and missed business opportunities. Additionally, metrics like content diversity help ensure inclusivity and fairness, preventing unintended bias against specific content types or user groups and maintaining YouTube's broad appeal. Tracking operational metrics is equally essential to ensuring the system's efficiency and scalability. Metrics such as latency and throughput confirm that the model runs effectively in production, minimizing delays and ensuring it can handle large volumes of data. System uptime further guarantees reliable and consistent predictions, enabling smooth user experiences. Together, these metrics ensure that the model delivers measurable ROI, aligns with business objectives, and provides value to both content creators and platform users.

Before we address the problem of identifying three thresholds that correspond to the green, yellow, and red, it's necessary to define them clearly and connect the threshold definition to our model's selected metrics.

Thresholds help us to classify the model's performance on different categories, and here, these three colors represent different levels of how the model's health was measured. Green shows the model performs as expected with no intervention to revise the framework of the model; Yellow represents the model showing signs of degradation that after the model deploys, close monitoring is required; and finally, red means the model's performance has deteriorated to an unacceptable level; it must be retired or retrained. These three colors are not really vivid colors or images that need to be represented; they are divided to make it easier to define what judgments we should make based on the model's metrics and to suggest ways to reduce bias and mitigate risks. Then we go back to the model's selected metrics to see how these metrics were calculated and classify thresholds based on the primitive metrics.

There are two types of metrics we utilized in our model: RMSE and R^2. RMSE was defined as lower values indicating better performance (measures the difference between predicted and actual values). R^2, the coefficient of determination, are values closer to 1 that indicate better performance (measures the percentage of variance explained by the model). The goal of setting the threshold is to ensure that model monitoring is aligned with how the model is organized when there is new input data and new variables applied when the model is deployed in real life. In this case, there are certain considerations in setting the value for thresholds: the model performance, the business requirements, and the baseline for model stabilization. The baseline of our model is: Train RMSE: 2.12M, the Test RMSE is about 2.91, and Train R^2 is 0.978, the Test R^2 is 0.961 (Appx. 2), and this is the value we set for green, yellow, and red thresholds (also in Appx. 1) based on considerations and performance baseline.

To explain the threshold value under the consideration of model performance, we basically capture the model's characteristics and want to maintain the high performance. The green thresholds are closely aligned with the historical performance. For example, if the RMSE (Train $\leq$ 2.15M and Test $\leq$ 3.0M) indicates no degradation. For R^2, when Train $\geq$ 0.975 and Test $\geq$ 0.95 reflect metrics slightly above or at historical levels, signaling model reliability. The yellow thresholds are designed to capture early signs of performance degradation while remaining within acceptable variability. When the RMSE for the train dataset is between 2.15M and 2.5M, the test data is between 3M and 3.5M, reflecting minor increases in error beyond historical metrics but still within tolerable limits. These thresholds provide an early warning system to closely track the model's performance and initiate mitigation strategies if needed. Lastly, the red thresholds represent a significant deviation from the model, and it's a signal to drop the model maybe. If the R-square is less than 0.95 for Train and Test is less than 0.90 when there are new input variables embedded in the original model, red flags indicate the accuracy has dropped below the level and the YouTube trending model may lose the predictive power. The model must be pulled from production as it no longer meets performance standards.

In the business application perspective, these thresholds also transfer important information about the model deployment. The green thresholds (Train RMSE ≤ 2.15M, Test RMSE ≤ 3.0M, Train $R^2 \geq$ 0.975, Test $R^2 \geq 0.95$) align with business tolerance for prediction errors under normal conditions. These thresholds reflect scenarios where model predictions are sufficiently accurate to support decision-making without influential risks, and business operations could take it as an indicator that the model is working properly, such as deciding whether or not to give a YouTube channel more volume or increase commercial time. The yellow thresholds reflect performance that approaches critical business boundaries. When RMSE is between 3.0M and 3.5M, it may result in suboptimal decisions but are still manageable, and accuracy between 0.90 and 0.95 may indicate a weaker ability to explain variance but it could still predict. These thresholds signal that the model requires monitoring and potential recalibration to avoid business disruptions. The Red Flags (Train RMSE > 2.5M, Test RMSE > 3.5M, Train $R^2 < 0.950$, Test R^2 < 0.90) exceed business tolerances, where prediction errors or inaccuracies result in unacceptable risks if media businesses cannot control certain contents that are deviated from the ethical issues, highly invested videos yield little to no revenue, content is boycotted by viewers, etc. In these cases, the model must be removed from production and retrained or replaced. Business requirements ensure the thresholds reflect the real-world impact of model performance, and it's an alert for stakeholders.

Green flag indicates optimal model performance against historical benchmarks such as Training RMSE ≤ 2.15M, Test RMSE ≤ 3.0M, Training $R^2 \geq 0.975$, and Test $R^2 \geq 0.95$. This indicates that the model reliably predicts YouTube video trends, enabling organizations to make accurate decisions such as increasing ad spend or prioritizing the promotion of high-potential videos. To maintain this level of performance, regular evaluations should be conducted to ensure the model remains stable in the face of changing data patterns and is proactively updated as needed.

Yellow flags point to early signs of performance degradation, where metrics such as Train RMSE between 2.15M-2.5M or Test R^2 between 0.90-0.95 deviate slightly from historical benchmarks. These thresholds indicate that while the model's predictions are still usable, their reliability has begun to decline. For example, incorrect predictions of trends may become more frequent, which could lead to poor ad placement or video prioritization errors. Close monitoring, error analysis, and possible hyperparameter recalibration can mitigate further degradation and keep the model within acceptable business limits.

Red flags indicate critical performance deterioration, with metrics like Test RMSE exceeding 3.5M or Test R^2 dropping below 0.90. Such significant deviations suggest the model is no longer capable of accurately predicting YouTube trends, which could lead to costly business errors, such as investing heavily in content unlikely to trend or overlooking videos with strong potential. In these cases, the model must be removed from production immediately. Retraining the model on updated datasets or replacing it

with a more robust alternative ensures performance standards are restored. These thresholds not only safeguard model reliability but also prevent business risks associated with poor predictions, reinforcing their importance in decision-making processes.

In our project, one key challenge we addressed was determining the frequency of retraining our model to maintain high performance and relevance while considering computational efficiency. Given the dynamic nature of YouTube's platform, where viewer preferences and trending topics shift rapidly, we explored different retraining options.

Our group considered several retraining frequency options, each with distinct advantages and limitations. First are weekly updates and quarterly rebuilds; this strategy involves weekly retraining on the most recent data to capture short-term trends, along with quarterly full retraining on the entire dataset to address broader structural shifts. Next, we explored monthly retraining possibilities; this approach updates the model once a month, striking a balance between frequent updates and computational costs. Finally, we considered daily retraining, which is also crucial in some contexts. Although appealing for immediate adaptability, this option was deemed impractical due to excessive resource requirements and the risk of overfitting to day-to-day noise in the data. After careful consideration, our group chose Weekly Updates with Quarterly Rebuilds for its ability to balance responsiveness to short-term changes and stability over time. Given the transient nature of daily trends, our model would struggle to generalize if retrained too frequently. By focusing on weekly updates, we avoided these pitfalls while ensuring our model remains responsive to new trends.

Our project was to develop a model that predicts and analyzes trending content effectively. Weekly updates ensure our model adapts to short-term shifts, such as viral challenges or sudden spikes in viewer interest, while quarterly retraining mitigates long-term degradation caused by evolving platform algorithms and user behavior. For example, YouTube’s recommendation algorithm may prioritize different factors over time, such as engagement metrics (*likes*, *comment_count*) or video metadata (*tags*, *titles*). Without regular updates, our model would gradually lose its predictive power. By retraining weekly, we kept our performance scores high, ensuring that our project met its goals of timely and accurate predictions.

For our team, data drift refers to changes in the statistical properties of the input data over time. These changes might affect the model's performance because the patterns the model was trained on no longer match the incoming data. Concept drift happens when the relationship between the input data and the target variable evolves over time. This is more critical because it affects the underlying logic of the model. Our team needs to detect and adapt to concept drift to ensure predictions and recommendations remain accurate. We also recognized that data drift and concept drift could significantly impact our model. Viewer demographics and preferences evolve constantly; for instance, the rising popularity of

short-form videos could shift the statistical properties of our input features. We implemented drift detection mechanisms to monitor feature distributions and trigger retraining when significant changes were detected. For concept drift, the relationship between input features (*tags*, *titles*, *likes*) can change as YouTube modifies its recommendation system. Another crucial approach was regular retraining and ensemble modeling, which allowed us to adapt to new patterns while retaining the predictive power of historical data. These strategies helped ensure the model stayed relevant and robust despite the evolving YouTube ecosystem. After evaluating the options, our group determined that weekly updates paired with quarterly full retraining was the best approach for our project. Weekly updates enabled us to stay aligned with rapidly changing trends, while quarterly rebuilding ensured the model incorporated broader shifts in user behavior and platform policies. This combination addressed the needs of a highly dynamic environment like YouTube while maintaining computational efficiency.

All in all, our group prioritized a retraining strategy that balances responsiveness and stability. By choosing weekly updates and quarterly retraining, we ensured our model stayed relevant to short-term trends while remaining robust against long-term data shifts. This approach aligns with our goal of delivering a high-performing and adaptable model for YouTube trending analysis. Through proactive strategies to address data drift and concept drift, we safeguarded the model's accuracy and ensured it met the dynamic needs of the platform.

Appendix

FLAG	RMSE THRESHOLD (TRAIN/TEST)	R^2 THRESHOLD (TRAIN/TEST)
Green	Train ⩽ 2.15M, Test ⩽ 3.0M	Train ⩾ 0.975, Test ⩾ 0.95
Yellow	2.15M < Train ⩽ 2.5M, 3.0M < Test ⩽ 3.5M	0.950 ⩽ Train < 0.975, 0.90 ⩽ Test < 0.95
Red	Train > 2.5M, Test > 3.5M	Train < 0.950, Test < 0.90

Appx.1

```
Best Model Variation: 2
Hyperparameters: {'learning_rate': 0.05, 'n_estimators': 200, 'max_depth': 6}
Variation        Variation 2
Train RMSE    2120971.173631
Test RMSE     2909245.576456
Train R^2           0.978047
Test R^2            0.960791
Name: 1, dtype: object
```

Appx.2

```
OS: Darwin Darwin Kernel Version 23.3.0: Wed Dec 20 21:28:58 PST 2023; root:xnu-10002.81.5~7/RELEASE_X86_64
Operating System: Darwin 23.3.0
Python Version: 3.11.5 (main, Sep 11 2023, 08:19:27) [Clang 14.0.6 ]
```

Appx. 3

```
Package Versions:
matplotlib.pyplot: 3.7.2
nltk: 3.8.1
numpy: 1.24.3
os: Version not found
palettable.colorbrewer.qualitative: 3.3.3
pandas: 2.0.3
re: 2.2.1
seaborn: 0.12.2
shap: 0.46.0
sklearn.decomposition: 1.3.0
sklearn.feature_extraction.text: 1.3.0
sklearn.metrics: 1.3.0
sklearn.model_selection: 1.3.0
sklearn.preprocessing: 1.3.0
textblob: 0.18.0.post0
wordcloud: 1.9.3
xgboost: 2.0.3
zipfile: Version not found
```

Appx. 4