AEYECATCHER.PY

CS611 - Machine Learning Engineering

Group 9
Christover Abraham Manafe
Loh Kwang Peng Micheal
Low Siang Leng Henry
Yee Jin Kett

Table of Contents

1. Business Problem & Implications

1.1. Problem Background

In today's digital age, social media platforms and websites have become an integral part of our lives, and the amount of content being shared and uploaded online is increasing exponentially. With the rise in popularity of social media platforms like TikTok, Instagram, and Facebook, the need for explicit/NSFW[1] image moderation has become more important than ever. With 3.2 billion images and 720,000 videos being shared daily (T.J. Thomson et.al, 2022), this has given rise to the complexity of content moderation. Content moderation is an industry-wide problem as cited by TikTok CEO Chew Shou Zi, and it is tough to identify and take down objectionable contents[2] such as suggestive content, violence, vices & racial slurs in a timely manner.

While social media giants like Facebook and TikTok have invested heavily in machine learning and human moderators to conduct moderation activity to remove unsafe content from their respective platforms, start-ups and SMEs are unable to employ the similar processes due to budgetary constraints.

1.2. Project Focus

Our project aims to value add to this field by developing a deployable machine learning pipeline for explicit image classification, with a particular focus on explicit nudity detection.

1.3. Project Scope

We plan to use state-of-the-art machine learning algorithms and techniques to develop a model that can accurately detect and filter out explicit images, including, but not limited to nudity and sexual exposure. Through this, businesses can leverage on a powerful yet cost-effective tool to moderate the content on their platforms, enabling users' trust and safety while maintaining brand reputation.

Subsequently, we would develop a cloud-native solution by leveraging on services such as Amazon SageMaker and AWS Lambda that is highly tailored to the business' needs.

1.4. Stakeholders – Users & Administrators

Stakeholders, including both users and administrators, can leverage our machine learning system in various ways to enhance their experience and ensure a safer online environment.

Users of social media platforms will upload images and receive feedback on their contents from the pipeline. This feedback will indicate if the image contains explicit nudity or not. Additionally, users can perform self-labelling by reporting inappropriate images (in situations where the ML system fail to flag out inappropriate images). When a certain threshold of reported images is reached, the system will trigger a model retraining to improve the accuracy of the pipeline's explicit image classification over time.

On the other hand, social media community managers will be the primary administrators of our machine learning system. They will be responsible for maintaining the pipeline's functionality and ensuring the accuracy and reliability of the system. As part of their role, they will monitor the pipeline's performance, fine-tune the system parameters, and carry out periodic updates to the model. By utilizing our ML system, administrators can focus their efforts on managing the platform and creating a seamless user experience, while having confidence in the system's ability to enhance content moderation and foster a safer online community.

Existing use-cases suggest that community managers often face the challenge of moderating user-generated content in real-time. To tackle this challenge, some companies have implemented machine learning systems to help identify inappropriate content and flag them for review. Our machine learning system aims to provide a similar solution that can effectively support social media community managers in monitoring user-generated content for explicit nudity. By leveraging self-labelling features, the system can also promote user engagement and foster a safer online community. Overall, our ML system offers stakeholders a comprehensive solution that facilitates content moderation, empowers user engagement, and ultimately contributes to a more responsible and respectful online environment.

[1] Not safe for work

[2] See Appendix: Figure A for common categories of content moderation on Social Media platforms

2. Data Collection & Project Datasets

2.1. Data Sources

In our data collection process[3], we evaluated different options to obtain the necessary dataset for our explicit image classification project. While one of the feasible options was to use Amazon SageMaker Ground Truth to label web scraped images from known explicit sites and Google safe search images, we ultimately decided to leverage existing pre-labelled datasets, review and consolidate the images, and use Amazon Rekognition's "DetectModerationLabels" method as our labelling tool to generate multiple sub-classes/labels to improve the granularity of our dataset. This approach allowed us to improve the quality of the data we use for training, validation, and testing while minimizing the labelling costs. Moreover, Rekognition uses an existing trained model to classify/label the images, making it a more cost-effective solution compared to Ground Truth, which uses human labellers.

2.2. Data Labelling

For our data labelling process, we leveraged Amazon Rekognition[4], an image and video analysis service provided by AWS. We combined images from multiple sources, including the NudeNet classifier dataset, nsfw data scraper

NSFW images and 50,000 safe/borderline images. Basic preprocessing (removing corrupted images, invalid image format) was also done prior to uploading onto the S3 Bucket. We used Amazon Rekognition's "DetectModerationLabels" function to generate parent labels and child sub-labels for each NSFW image. After reviewing the labels, we selected images based on their sub-labels to balance our dataset. We then created image labels and copied the images into different folders within an Amazon S3 bucket based on their new labels. With the number of sub-labels of each NSFW image, it will be useful to inform community managers and offenders why the images are classified NSFW ("Borderline Nudity") so as opposed to a Safe/NSFW classification. Despite the need for greater training images, the team feels that this will also allow the model to be more resilient against future content drifts.

2.3. Dataset Statistics

Figure D in the Appendix describes the number of labelled datapoints that the team has collected for training.

2.4. Data Imbalance

Based on our sampled data, we have identified that there is a tendency for class imbalance. We will address this in our data preprocessing step.

2.5. Dataset Format

Our dataset will be stored in an AWS S3 bucket with a labelled folder structure for easy data access. We will use Amazon SageMaker to run the entire machine learning workflow, including data pre-processing, feature engineering, model training, tuning, evaluation, deployment, and monitoring.

Amazon SageMaker tools will facilitate data reading, processing, and transformation. Feature engineering will extract meaningful image features for improved model performance. We will explore using transfer learning models such as ResNet50 and ViT models. The trained models will be deployed to an endpoint for prediction on new data. These pipeline architectures ensure effective and efficient explicit nudity detection using machine learning.

2.6. Data Privacy & Ethical Considerations

As our project focuses on explicit nudity detection, we recognize the importance of ensuring the privacy and ethical considerations of the data we collect. To protect the privacy of individuals, we will ensure that all images used in our dataset are appropriately anonymized and stripped of any identifying information. Moreover, we will limit access to the data to only those team members who require it for the project, and we will store the data securely in accordance with AWS security best practices. Ethically, we will ensure that our project is not used for any malicious or harmful purposes and that the project's end goal serves a legitimate purpose in society. We will also follow all relevant laws, regulations, and guidelines related to the use of explicit imagery for research

[3] See Appendix: Figure B for the Data Collection Pipeline
[4] See Appendix: Figure C for the list of categories classified by Amazon Rekognition

purposes. Finally, we will be transparent about our data collection and labelling process and provide clear communication to our stakeholders on how the data will be used and protected.

3. Machine Learning System Architecture

The team has segmented the pipeline into a few components for the purpose of the project[5].

Service	Explanation
Amazon S3 Training Bucket	Stores training images that will be converted into PyTorch Tensor for model training
Amazon S3 Interim Bucket	Stores reported/appealed images for moderators to evaluate and take appropriate action.
Amazon SageMaker – Training	Conduct model training, building and compilation
Amazon SageMaker – Model Registry	Contains a catalogue of models to track and manage
Amazon SageMaker - Endpoint	Deploying model for real-time inference
AWS CodeCommit	Store source code and changes history
AWS CodeBuild	Compiles source code and build model
AWS CodePipeline	Automate pipeline for CI/CD
Amazon CloudWatch	Monitor model performance from logs and to send alarm
AWS Lambda	Serverless computing service to perform inference, update data label, and trigger model training pipeline.
API Gateway	Managed service that facilitates interactions between public requests to AWS services.

3.1. Model Building[6]

3.1.1. Model Building Workflow[7]

In the development of the model, the team implemented a continuous integration approach. The commencement of this process is signaled when the model building code is committed into the repository. This submission sets off a CloudWatch event, which in turn initiates the model training pipeline in CodePipeline.

The model training pipeline engages the SageMaker Pipeline to carry out various stages of training. These stages are listed as follows:

1. The preprocessing of training data.
2. The actual training of the model.
3. The evaluation of the model.
4. The final step, which involves registering the model into the Model Registry.

During the evaluation stage, the trained model must reach a predefined level of accuracy before it is added into the model registry. This requirement is put in place to guarantee that any newly trained model satisfies the baseline performance standards for the model.

3.1.2. Data Preprocessing

In the data preprocessing stage, the team will be extracting up to 1000 images per class and adopting 80/10/10 split of training, validation and test set. This is to ensure that enough data will be used for model training, while addressing the class imbalance issue and cost considerations. The images will then undergo a series of transformation to ensure that the images conform to the requirements of the model (224x224, normalized).

We have also use various data augmentation methods on the training set, such as random horizontal and vertical flips and rotation. These augmentation techniques will help increase the amount of training data, reducing risk of overfitting, and improve model generalization by introducing diverse variations in the augmented images.

[5] See Appendix: Figure E for the final system architecture.
[6] See Appendix: Figure F for the detailed model building architecture.
[7] See Appendix: Figure G for the CodePipeline stages for model building.

3.1.3. Model Training

ResNet50

ResNet 50 is a deep convolutional neural network that employs residual networks. It introduced skip connections to address the vanishing gradient problems, enabling the training of deeper networks. (Kaiming He, et.al, 2015) While at its inception it achieved the state-of-the-art results, other model architectures have since surpassed it. However, it remains as one of the more popular models due to the simplicity of understanding the model.

Vision Transformer (ViT-16)

Vision Transformer is an image classification architecture that employs transformer architecture to process images. It divides the input images into patches and process them through a series of self-attention layers, enabling long-range interaction between image elements. The model also eliminates the need for convolutional layers, allowing us to capture global dependencies in the image.

In the training step, the team has frozen all the feature extraction layers, finetuned the last fully connected classifier layer of the following models:

Model	Accuracy
Resnet50	~20%
ViT-16	~60%

Based on the model performance, we have identified that the ViT-16 will be the most appropriate as it outperforms the rest of the models. While the model can be further fine-tuned to achieve better performance, the team will be utilizing the model after 10 epochs of finetuning due to resource constraints.

3.1.4. Model Quantization

As the model size can get quite substantial, we have introduced post-training quantization to reduce the precision of weights, allowing for compression of models while retaining similar performance. While the compression of model by way of a reduction in precision results in a degradation of model, the team has built in a conditional step, where the quantized model will be benchmarked against the un-quantized model based on accuracy. Ultimately, the un-quantized model was deployed as the deviation was greater than 5% threshold set. The un-quantized model size was also relatively manageable at around 300mb.

3.2. Model Deployment[8]

3.2.1. Model Deployment Workflow[9]

Our project employs an image classification model designed to operate in a real-time inference setting. Given the time-sensitive nature of our task and the potential impact of erroneous classifications, we have chosen a deployment workflow that maximizes both model accuracy and system reliability.

Our workflow is designed as a sequence of steps: Build, Deploy to Staging, Approve Production Deployment, and Deploy to Production. The workflow initiated either when modifications are made to the model deployment source code or when a new model gets approved in the model registry. The workflow then builds a package from the repository, which encompasses both our staging and production deployment CloudFormation template.

Utilizing the template, the workflow updates the stacks in CloudFormation. This either results in the creation or the update of the SageMaker endpoint in the staging environment. Following these updates, we execute an inference test on the staging endpoint.

The `Approve Production Deployment` stage, a manual checkpoint, is the gatekeeper in preventing the workflow to automatically deploy the model into production environment. At this stage, the team could conduct additional testing on the staging endpoint. Based on the results of these tests, the team can decide whether to authorize the deployment into the production environment.

[8] See Appendix: Figure H for the detailed model deployment architecture.

[9] See Appendix: Figure I for the CodePipeline stages for model deployment.

Once the stage approval is given, which happens after successful testing in the staging environment, the workflow proceeds to deploy the model. The deployment strategy used is determined by the specifications within the CloudFormation template included in the package.

3.2.2. Auto Scaling Policy

Given the nature of the business use case, being able to automatically scale the endpoint instance horizontally will be essential to ensure steady performance with the appropriate cost trade-offs. With this, we made use of the “SageMakerVariantInvocationsPerInstance” metric, monitoring a target value of 70 per minute, with a scale out cooldown of 5 mins and scale in cooldown of 10 mins. These are tentative values and will be adjusted accordingly to fit individual communities as they see fit.

3.2.3. Deployment Strategy

The team adopted Canary deployment strategy in the deployment workflow. Canary deployment is a type of incremental rollout process where new versions of a model (or application) are released to a small, controlled subset of users or environment before a full rollout. This deployment strategy allows us to test the model's performance, assess potential risks, and detect issues early without affecting the entire user base or system.

In the context of our project, using the **Canary Deployment strategy** for our explicit image classification model offers several benefits. Firstly, given that our system operates in a real-time environment, it's critical to ensure a seamless experience for end-users. By initially deploying the new model version to a limited subset of traffic, we can monitor its performance, measure prediction accuracy, and identify any unexpected behaviours or anomalies before it affects all users.

Secondly, this approach provides us an opportunity to compare the new model version with the existing one in a live setting. We can evaluate metrics such as model latency, throughput, and resource usage under actual load conditions. Such direct comparison under real-world conditions provides valuable feedback to inform our decision about the full deployment of the new model.

Finally, the incremental rollout reduces the risk associated with deploying new models. If any problems arise during the Canary phase, we can quickly rollback the deployment, minimizing the impact on the overall system and user experience. It also gives us time to diagnose the issue and make necessary adjustments before a broader rollout. Thus, Canary Deployments act as an essential safety net, ensuring high reliability and performance consistency of our image classification system.

The team has added CloudWatch alarms that are used for managing rollback procedures during Canary deployments. For this purpose, the chosen metric is `InvocationModelErrors`. This alarm is available in both staging and production environment.

3.3. Monitoring & Retraining Step

3.3.1. User Feedback to handle Concept & Model Drift

Concept drift arises when the underlying data distribution & statistical properties evolve, rendering the model's assumptions invalid. It can be triggered by factors such as shifting user preferences, market dynamics, or external influences. Detecting and adapting to concept drift is essential for maintaining accurate predictions in dynamic environments, making the model suitable for communities with different social acceptance norms.

On the other hand, model drift refers to the degradation of model performance over time, even without changes in the data distribution. It can be caused by shifts in the operating environment, emerging patterns, or limitations of the model itself. Given our numerous labels, our model will be more susceptible to such shifts through iterations of retraining on predicted images. Monitoring and addressing model drift are crucial to uphold the reliability and effectiveness of the machine learning model.

3.3.2. Implementation of User Feedback – Discord Server Bot

For our project, we decided to implement this user feedback loop in Discord. A popular discord bot template[10] was modified to allow the bot to do the following:

Context	Bot Actions
User uploads image	Send the url containing the image to the ModelUpload Lambda through a POST API, which simultaneously sends the image to Sagemaker endpoint, staging S3 bucket and AWS RDS (to store the metadata), returning the RDS file id and the classification result. These are saved in a local database within the bot along with other metadata (i.e. user, channel, timestamp)
Image result – Safe	Does nothing
Image result - NSFW	Auto-Moderating actions: Timeout the user (10 seconds) and deletes the message. Sends a message in the moderator notification channel with relevant details. Sends a private message to the user with reason for the timeout and gives an option to appeal.
NSFW user – Appeal	Retrieves the RDS file id and send it to the ModelAppeal Lambda through a POST API, updating the appeal status in RDS and returning the appeal id. Sends a message in the moderator notification channel with relevant details.
NSFW user – Accept	Does nothing
User reports image	Sends a message in the moderator notification channel with relevant details.

By incorporating user feedback, involvement in model validation plays a vital role in detecting and mitigating drift. Users' interactions with the model's predictions through community engagement provide valuable insights into its performance. Whenever images are wrongly classified (via the appeal/report loop), moderators will then check/confirm the labels of those images, moving them into the training bucket to form the ground truth. Currently, images that are not reported will also be moved by the moderators/administrator every 24 hours to the training bucket.

Whenever the number of wrongly classified images crosses a pre-defined threshold, the lambda function will trigger the model training pipeline.

This implementation can be applied to any other online community in a similar fashion. Given the modularized nature of our project, the code can be used to build a separate pipeline in another AWS account. Community managers can then update the various API parameters for their own user feedback mechanism implementation.

3.3.3. AWS CloudWatch Alarms

As part of our real-time explicit image classification project, we've identified the necessity to closely monitor certain metrics in our machine learning (ML) system to ensure optimal performance and efficiency. These metrics, accessible through Amazon CloudWatch[11], provide valuable insights into our model's performance and can trigger necessary adjustments in the infrastructure or the model itself if predefined thresholds are breached. The team added a few alarms on SageMaker endpoint in both the staging and production environments, aiming to facilitate the monitoring process.

We also set an “Invocation Model Errors” CloudWatch alarm to monitor the number of failed invocations of our image classification model in the production environment. This alarm is pivotal as it provides immediate notification of spikes in error counts, which could signify serious issues with the model or the input data. This enables us to rapidly investigate and rectify any issues, maintaining a reliable service and ensuring a superior user experience, which is crucial for a real-time inference pipeline like ours. We set the alarm threshold to more than 5 model invocation errors in 5 minutes. Furthermore, this alarm is integrated into our deployment configuration. If the alarm is triggered during deployment, SageMaker will initiate an automatic rollback of the deployment process. By including this alarm in our deployment workflow, we strengthen the robustness and reliability of our machine learning system.

[10] Discord Bot Template from https://github.com/kkrypt0nn/Python-Discord-Bot-Template
[11] See Appendix: Figure J for the list of CloudWatch alarms.

Another essential metric we chose to monitor is CPU Utilization. While we also have autoscaling policy in each of the endpoint, keeping a watchful eye on the CPU usage of our model host can offer valuable insight regarding model computational demands. Such information might necessitate optimization of the model to reduce its computational load or an upgrade to a larger instance type to handle the model's demands better. Despite the autoscaling policy we have, closely monitoring CPU usage provides us with early warnings regarding potential disruptions caused by insufficient computing power. We set the alarm threshold to more than an average of 70% CPU usage (on average) for every 5 minutes interval based on baseline percentages.

In essence, integrating CloudWatch monitoring into our ML system allows us to respond promptly to performance issues, streamline computational resources, and, ultimately, ensure the provision of a high-quality, real-time explicit image classification service.

4. Limitations, Considerations & Future Works

4.1. Technical Limitations

Obtaining Feature Attribution with SageMaker Clarify Model Explainability[12]

An essential future enhancement for our image classification pipeline is the real-time monitoring of model explainability, applied both during training and on live data. This augmentation would substantially increase system transparency and robustness by providing instant insights into the model's decision-making process. This involves not only tracking performance metrics but also analysing the feature attributions given by SageMaker Clarify's SHAP values in real-time.

SHAP (SHapley Additive exPlanations) values offer a powerful tool in understanding feature importance. Originating from cooperative game theory, SHAP assigns each input feature an importance score, offering a breakdown of how each influences the model's prediction. For our image classification model, SageMaker Clarify can indicate the image regions most influential in making specific predictions, assisting us in discerning whether the model focuses on relevant parts or is distracted by background noise. This clarity is represented through a "heatmap" which highlights the most influential areas of the image, thereby providing greater transparency and accountability to the model's decision-making process.

4.2. Data Limitations

Amazon Rekognition was chosen as the data labelling solution as opposed to Amazon GroundTruth primarily due to cost concerns. This subjects the model to the inherent biases/distributions from Amazon Rekognition. Nevertheless, these should be mitigated in the long run through iterations of model retraining with the model adapting to the individual communities' user feedback.

4.3. Model Limitations

Model Accuracy & Experimentation with Proportion of Class Labels

The current model accuracy stands at around 60%. While this shows some capacity for explicit content detection, there is considerable room for improvement. One potential avenue for enhancing the model's precision involves expanding the dataset utilized during training. Having a more diverse, extensive dataset would allow the model to learn and distinguish nuances in images more effectively. Increasing the number of training epochs could also yield benefits; it allows the model additional opportunities to learn from the data. However, it's crucial to balance this with computational resources and the risk of overfitting.

Balancing the proportion of safe to Not Safe for Work (NSFW) images in model training is crucial for our explicit content moderation project. The costs of false negatives, where NSFW images are incorrectly deemed safe, are considerably higher than false positives due to potential user exposure to explicit content and subsequent harm to the platform's reputation. Therefore, we may opt for a model more inclined to predict NSFW, even if it slightly increases false positives. However, we must be cautious to avoid overfitting the model to the training data. To

[12] See Appendix: Figure K for an example of explainability on image classification with SageMaker Clarify .

achieve this balance, we will rigorously experiment with different data proportions to find an optimal performance level in real-world conditions.

4.4. Deployment Infrastructure

Model Latency

Another key metric to focus on is Model Prediction Latency. This measure refers to the duration it takes for our model to generate a prediction upon receiving an input. Latency plays a significant role in the user experience, particularly for real-time applications like ours. For this reason, we plan to monitor the ModelLatency metric under the AWS/SageMaker namespace in CloudWatch. By defining an acceptable threshold for latency based on our application's requirements, we can set up CloudWatch alarms to notify us if this limit is exceeded. This approach allows us to maintain the responsiveness of our service and ensure a seamless user experience.

Setting up a suitable baseline for Model Prediction Latency is essential to adequately monitor and react to potential issues in real-time. As we move towards a staging test with general users, we will begin collecting latency data under real-world conditions. This data will help us understand the typical latency our model exhibits under varying load and user interaction patterns.

In this staging phase, we will observe and analyze the trends and patterns of model latency. We will consider both average latency and peak times, accounting for user behavior patterns that might impact system load. By observing these patterns, we will be able to set a realistic and acceptable threshold for ModelLatency. Our aim is to set a baseline that accounts for typical usage, while also ensuring we can react swiftly if latency starts to exceed expected peaks, ensuring our system continues to deliver timely responses and a seamless user experience.

Adversarial Attacks

The model may be susceptible to adversarial attacks, where users intentionally provide inaccurate feedback or submit images designed to mislead the model. These attacks can degrade the model's performance over time, leading to an increase in misclassifications. Implementing robust verification processes for user feedback and deploying “defences” against adversarial attacks can help to mitigate this risk.

Pipeline Architecture

Our current implementation makes use of a real-time inference. Switching to an asynchronous inference setup may be more justifiable as the use case scales up.

4.5. Ethical & Legal Considerations

Using user images for model training raises significant ethical concerns, primarily revolving around privacy and consent. While the images could significantly improve model performance due to their real-world variability, users might oppose their personal content being used for such purposes, even if the images are anonymized. Additionally, considerations around the handling of potentially explicit images, especially those involving minors or non-consenting individuals, add layers of complexity. Addressing these concerns necessitates stringent data handling and usage policies, with user consent at the forefront.

4.6. Scope Expansion

While the current project focuses on detecting explicit nudity, the reality of content moderation extends to other potentially harmful or inappropriate material such as gore, violence, drug-related content, as well as different media formats like GIFs and videos. Expanding the project scope to handle these elements would increase the system's overall effectiveness but also introduce additional complexities. Each type of content and media format might require different detection techniques and algorithms, which would need to be seamlessly integrated into the existing infrastructure.

5. References

Alex000kim, Nsfw_Data_Scraper, (2022). GitHub repository, https://github.com/alex000kim/nsfw_data_scraper

Amazon Web Services (2020). *Explaining Image Classification with SageMaker Clarify*. Amazon SageMaker Examples. https://sagemaker-examples.readthedocs.io/en/latest/sagemaker-clarify/computer_vision/image_classification/explainability_image_classification.html

Brown, R. (2023, May 9). *Why social media content moderation is important for online platforms & how it works?*. Cogito. https://www.cogitotech.com/blog/why-social-media-content-moderation-is-important-for-online-platforms-how-it-works/

Cogito Tech LLC. (2023, May 9). Why social media content moderation is important for online platforms &amp; how it works?. Cogito. https://www.cogitotech.com/blog/why-social-media-content-moderation-is-important-for-online-platforms-how-it-works/

EBazarov, Nsfw_Data_Source_Urls, (2022). GitHub repository, https://github.com/EBazarov/nsfw_data_source_urls

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015, December 10). Deep Residual Learning for Image Recognition. arXiv:1512.03385. Retrieved from https://arxiv.org/abs/1512.03385

Kkrypton, Python Discord Bot Template (2023). GitHub repository, https://github.com/kkrypt0nn/Python-Discord-Bot-Template

Moderating content (2023). *Amazon Rekognition Developer Guide.* Retrieved from https://docs.aws.amazon.com/rekognition/latest/dg/moderation.html

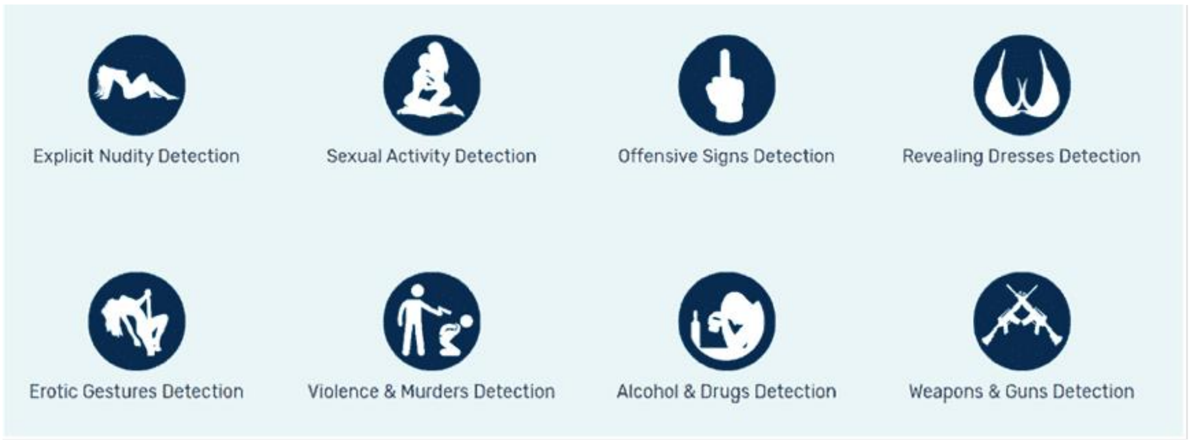
Matheus Oliveira Franca (2021, June 29). *Detection and categorization of suggestive thumbnails.* Retrieved from https://www.diva-portal.org/smash/get/diva2:1595278/FULLTEXT01.pdf

NotAI.tech, Nudenet, (2022). GitHub repository, https://github.com/notAI-tech/NudeNet

T.J. Thomson, Daniel Angus, Paula Dootson. (2022, December 21). *3.2 billion images and 720,000 hours of video are shared online daily. can you sort real from fake?*. The Conversation. https://theconversation.com/3-2-billion-images-and-720-000-hours-of-video-are-shared-online-daily-can-you-sort-real-from-fake-148630

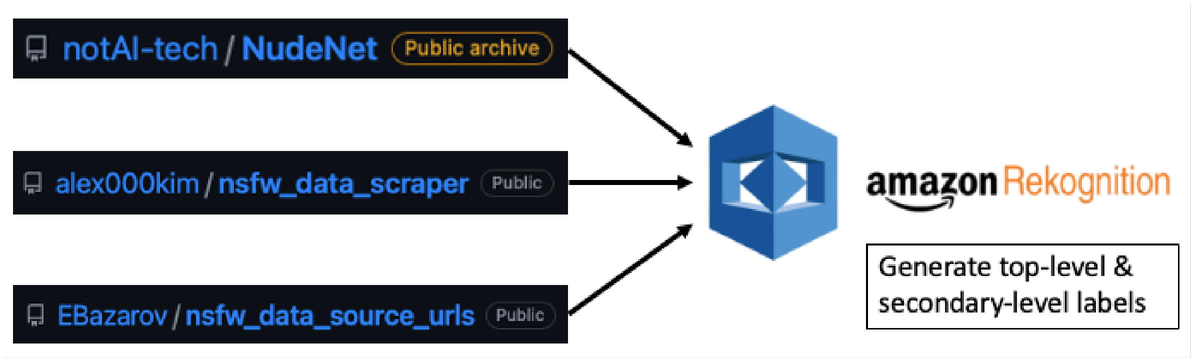
6. Appendix

Figure A: Types of Contents Moderated on Social Media Platforms (from Cognito)



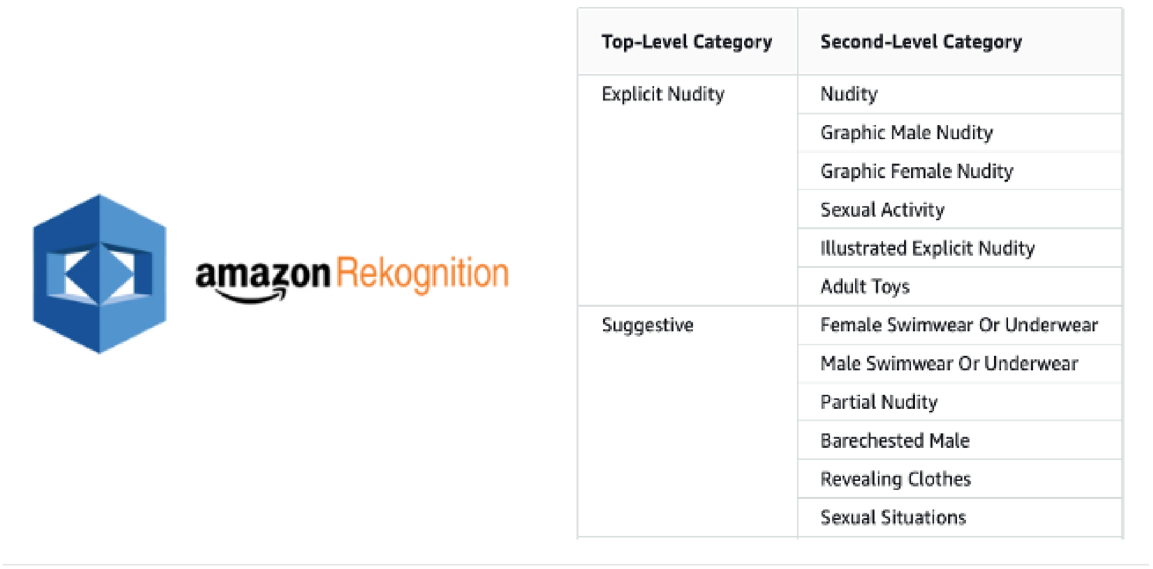
Explicit Nudity Detection
Sexual Activity Detection
Offensive Signs Detection
Revealing Dresses Detection
Erotic Gestures Detection
Violence & Murders Detection
Alcohol & Drugs Detection
Weapons & Guns Detection

Figure B: Data Collection Pipeline



notAI-tech / NudeNet
Public archive
alex000kim / nsfw_data_scraper
Public
EBazarov / nsfw_data_source_urls
Public
amazon Rekognition
Generate top-level & secondary-level labels

Figure C: Amazon Rekognition Categories (from Amazon Developer Guide)



amazon Rekognition

Top-Level Category	Second-Level Category
Explicit Nudity	Nudity
	Graphic Male Nudity
	Graphic Female Nudity
	Sexual Activity
	Illustrated Explicit Nudity
	Adult Toys
Suggestive	Female Swimwear Or Underwear
	Male Swimwear Or Underwear
	Partial Nudity
	Barechested Male
	Revealing Clothes
	Sexual Situations

Figure D: Dataset Statistics

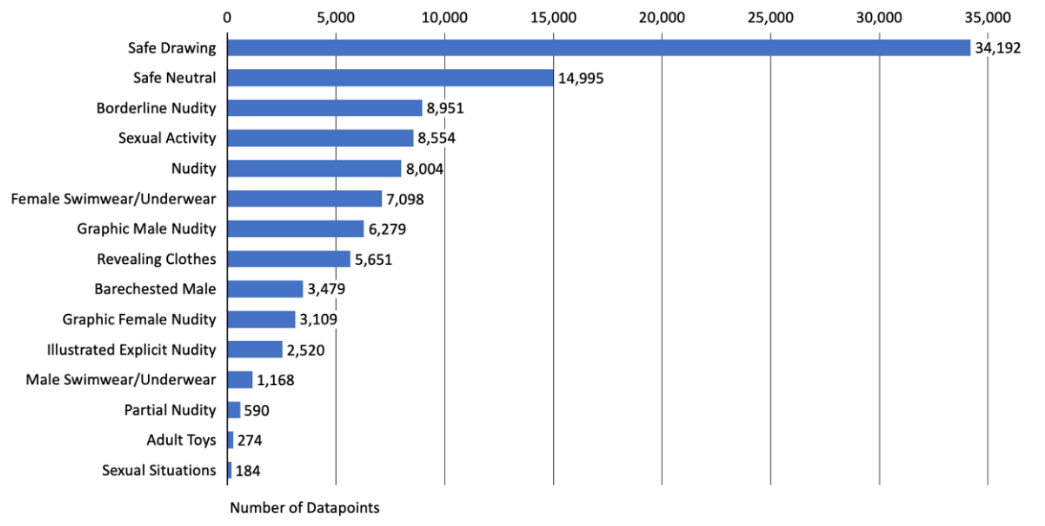


Figure E: Final Overall System Architecture

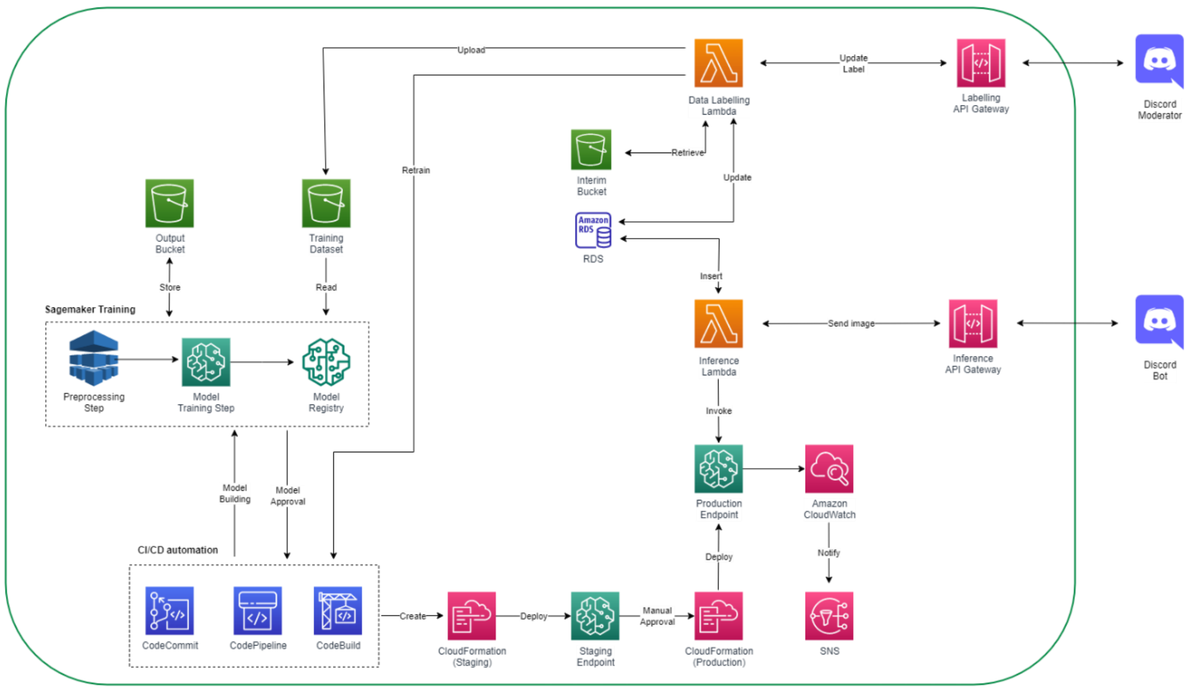


Figure F: Detailed Architecture for Model Building

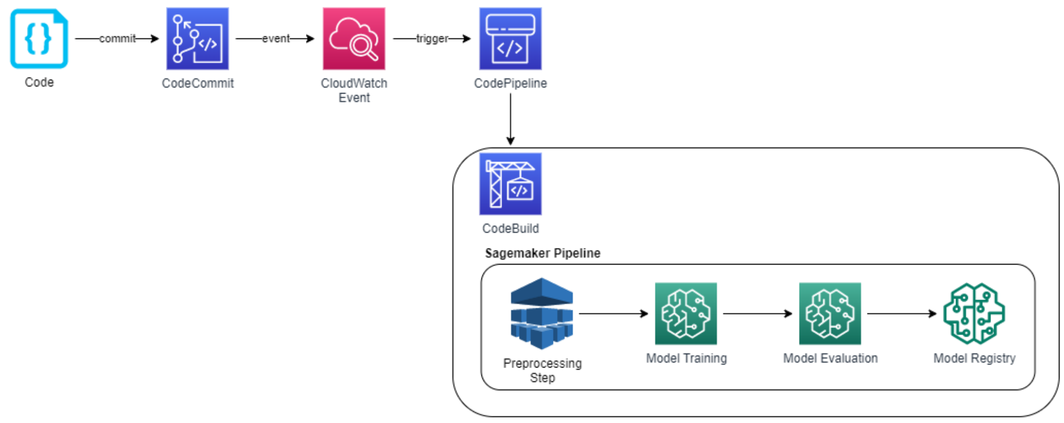
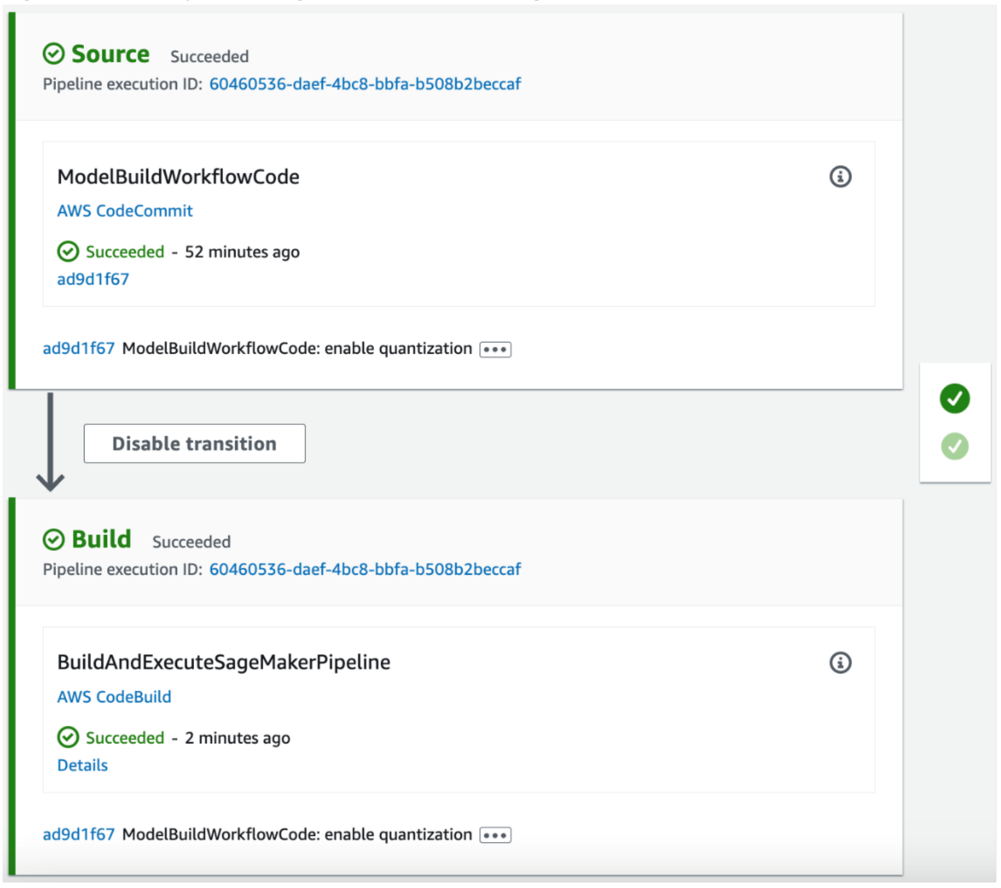


Figure G: CodePipeline Stages for Model Building



Source Succeeded
Pipeline execution ID: 60460536-daef-4bc8-bbfa-b508b2beccaf

ModelBuildWorkflowCode
AWS CodeCommit
Succeeded - 52 minutes ago
ad9d1f67

ad9d1f67 ModelBuildWorkflowCode: enable quantization

Build Succeeded
Pipeline execution ID: 60460536-daef-4bc8-bbfa-b508b2beccaf

BuildAndExecuteSageMakerPipeline
AWS CodeBuild
Succeeded - 2 minutes ago
Details

ad9d1f67 ModelBuildWorkflowCode: enable quantization

Figure H: Detailed Architecture for Model Deployment

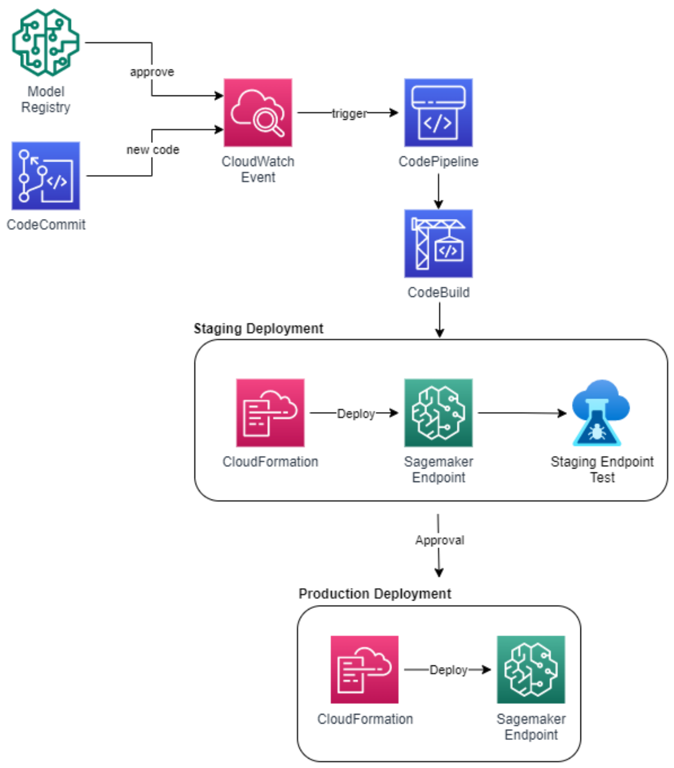
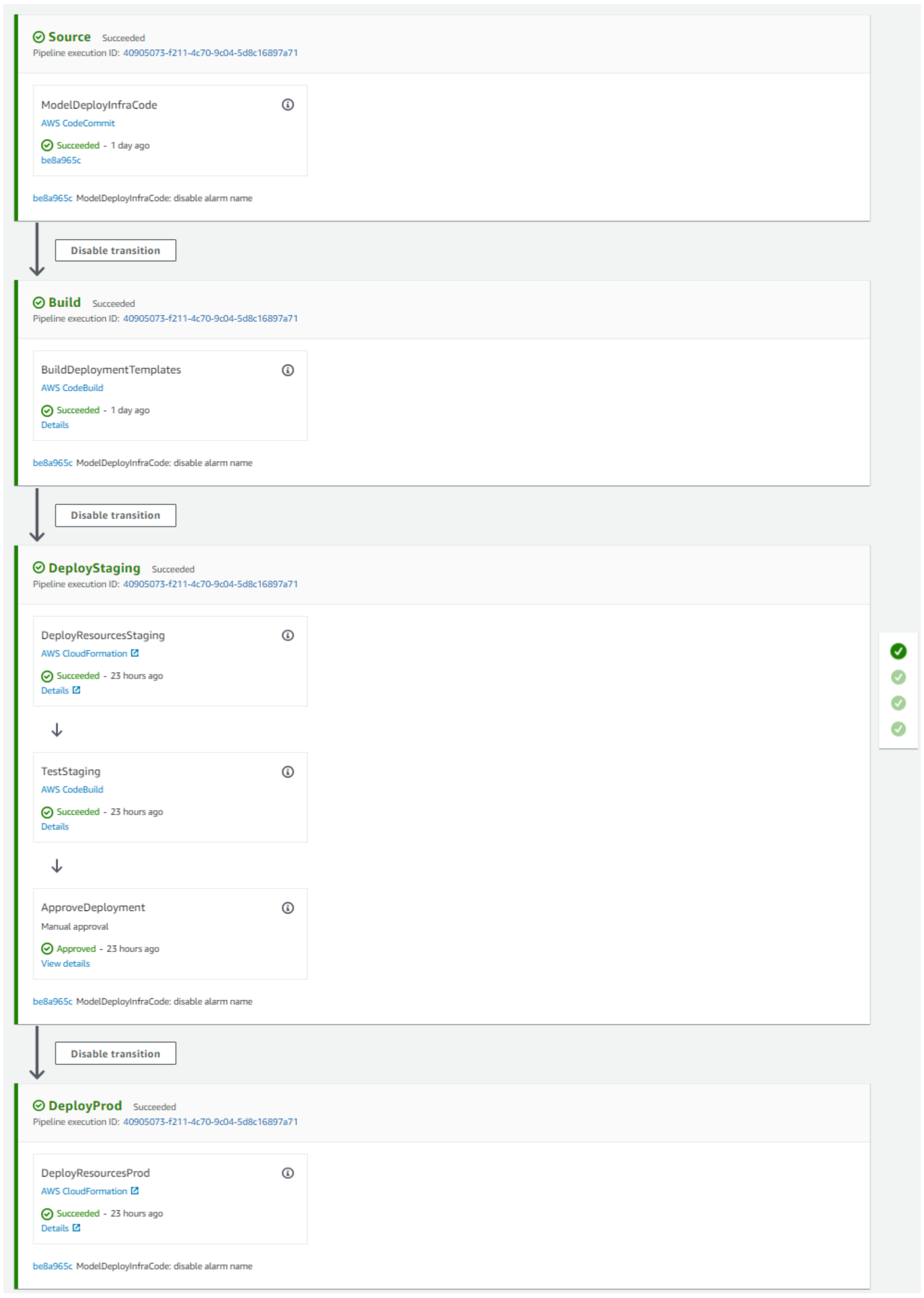


Figure I: CodePipeline Stages for Model Deployment



Source Succeeded
Pipeline execution ID: 40905073-f211-4c70-9c04-5d8c16897a71

ModelDeployInfraCode
AWS CodeCommit
Succeeded - 1 day ago
be8a965c

be8a965c ModelDeployInfraCode: disable alarm name

Build Succeeded
Pipeline execution ID: 40905073-f211-4c70-9c04-5d8c16897a71

BuildDeploymentTemplates
AWS CodeBuild
Succeeded - 1 day ago
Details

be8a965c ModelDeployInfraCode: disable alarm name

DeployStaging Succeeded
Pipeline execution ID: 40905073-f211-4c70-9c04-5d8c16897a71

DeployResourcesStaging
AWS CloudFormation
Succeeded - 23 hours ago
Details

TestStaging
AWS CodeBuild
Succeeded - 23 hours ago
Details

ApproveDeployment
Manual approval
Approved - 23 hours ago
View details

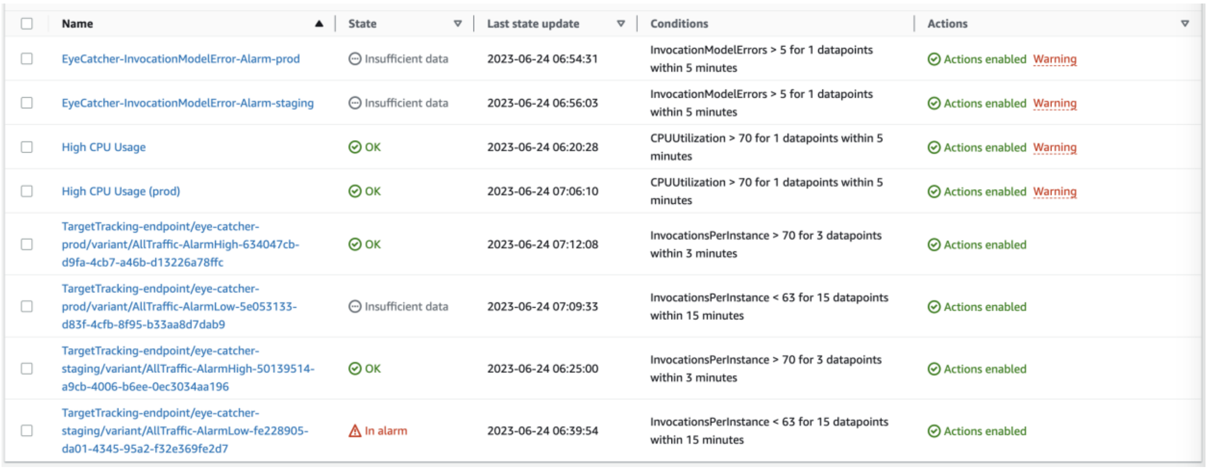
be8a965c ModelDeployInfraCode: disable alarm name

DeployProd Succeeded
Pipeline execution ID: 40905073-f211-4c70-9c04-5d8c16897a71

DeployResourcesProd
AWS CloudFormation
Succeeded - 23 hours ago
Details

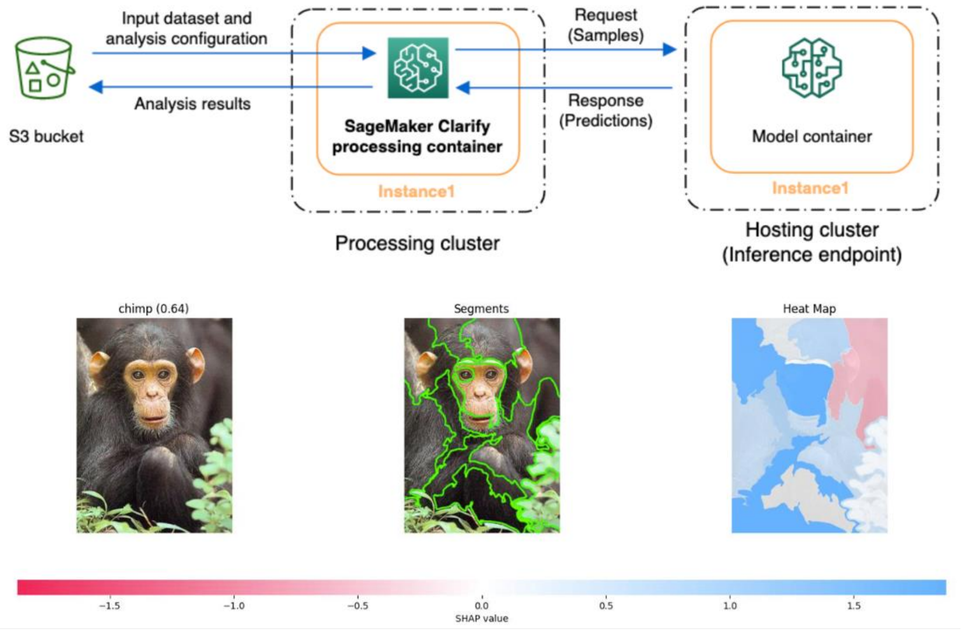
be8a965c ModelDeployInfraCode: disable alarm name

Figure J: Cloudwatch Alarms



Name	State	Last state update	Conditions	Actions
EyeCatcher-InvocationModelError-Alarm-prod	Insufficient data	2023-06-24 06:54:31	InvocationModelErrors > 5 for 1 datapoints within 5 minutes	Actions enabled Warning
EyeCatcher-InvocationModelError-Alarm-staging	Insufficient data	2023-06-24 06:56:03	InvocationModelErrors > 5 for 1 datapoints within 5 minutes	Actions enabled Warning
High CPU Usage	OK	2023-06-24 06:20:28	CPUUtilization > 70 for 1 datapoints within 5 minutes	Actions enabled Warning
High CPU Usage (prod)	OK	2023-06-24 07:06:10	CPUUtilization > 70 for 1 datapoints within 5 minutes	Actions enabled Warning
TargetTracking-endpoint/eye-catcher-prod/variant/AllTraffic-AlarmHigh-634047cb-d9fa-4cb7-a46b-d13226a78ffc	OK	2023-06-24 07:12:08	InvocationsPerInstance > 70 for 3 datapoints within 3 minutes	Actions enabled
TargetTracking-endpoint/eye-catcher-prod/variant/AllTraffic-AlarmLow-5e053133-d83f-4cfb-8f95-b33aa8d7dab9	Insufficient data	2023-06-24 07:09:33	InvocationsPerInstance < 63 for 15 datapoints within 15 minutes	Actions enabled
TargetTracking-endpoint/eye-catcher-staging/variant/AllTraffic-AlarmHigh-50139514-a9cb-4006-b6ee-0ec3034aa196	OK	2023-06-24 06:25:00	InvocationsPerInstance > 70 for 3 datapoints within 3 minutes	Actions enabled
TargetTracking-endpoint/eye-catcher-staging/variant/AllTraffic-AlarmLow-fe228905-da01-4345-95a2-f32e369fe2d7	In alarm	2023-06-24 06:39:54	InvocationsPerInstance < 63 for 15 datapoints within 15 minutes	Actions enabled

Figure K: SageMaker Clarify Example (from Amazon SageMaker Examples)



- **Segments**: Highlights the image segments.
- **Shades of Blue**: Represents positive Shapley values indicating that the corresponding feature increases the overall confidence score.
- **Shades of Red**: Represents negative Shapley values indicating that the corresponding feature decreases the overall confidence score.