Project Name: AI-Powered Movie Sentiment Rating System

Project Description: This project aims to convert user-submitted movie reviews into rating scores using sentiment analysis from a Natural Language Processor. Movie ratings and reviews can be sold to online movie platforms, such as Netflix, to provide movie recommendations.

1. What problem is the product aimed at solving?
    1. This product aims to address the challenges of generating accurate rating scores for movie review text.
2. Who is the product geared towards (targeted audience)?
    1. The primary target audience is online movie platforms, such as Netflix or Amazon Prime, to use these ratings in their recommender system.
3. How is the product unique?
    1. This product is unique in its application of transfer learning, where a pre-trained NLP model is fine-tuned for the domain of movie reviews.

Project Components:

1. **Frontend Server**: A Ktor web application to provide an HTML frontend to allow end-users to submit movie reviews and view sentiment analysis reports.
2. **Movie Review Data Collector**: A Ktor service responsible for downloading the raw movie review dataset online, ingesting the data into a simple JSON format, and then persisting it in a NoSQL document store. The persisted data will be used in both fine-tuning training and the evaluation of the prediction from the data analyzer model.
3. **Sentiment Data Analyzer**: A ktor web service that loads a pre-trained Huggingface NLP model (such as DistilBERT) to perform sentiment classification. The sentiment analysis result will be persisted in a collection in the document store. Data analyzer components accept inputs from both interactive and batch jobs. Priority will be given to the /analyze RESTful API endpoints over the pub/sub batch job.
4. **Data persistence:** A NoSQL Document store will be used within the entire system. It will store the ingested dataset collected from Kaggle, as well as all the sentiment analysis results for reporting purposes.

Product environment: Google Cloud

Major Project Workflow:

1. **Fine-tuning (Offline)**: Before the NLP model in the data analyzer components goes live in production, it will undergo fine-tuning training using the collected online dataset via a Vertex AI script. The trained NLP model will then be registered in the artifact registry and hosted in the data analyzer container.
2. **Interactive (Real-Time):** Once the data analyzer is live in production. Users can submit movie reviews via the web UI. The web UI will internally call the RESTful API for the

sentiment analyzer component. The NLP model will then analyze the review text and return the result ratings to be displayed on the webpage.
3. **Evaluation (Weekly)**: A scheduled job will be executed periodically using the collected dataset to randomly evaluate 10% of the dataset and assess the accuracy of the sentiment analysis. If the model prediction is lower than a specific threshold limit, the model should be retrained to adjust the parameter weight.

System design consideration: (CAP Theorem)

1. The data analyzer should give higher priority to interactive movie reviews over the batch evaluation jobs. User ratings should be returned to the web UI within a reasonable latency limit.
2. Interactive workflow is end-user oriented. Interactive workflow prioritizes availability (low latency) over consistency. Evaluation workflow prioritizes consistency (accurate metrics) over availability.
3. The overall health of the Production system would be monitored with tools such as Prometheus and Grafana. Since this system is designed with big data in mind, components can be scaled horizontally if needed.

Software engineering plan:

1. The project would be developed using the Ktor basic service skeleton template and adapted to fit project requirements.
2. Components will be individually developed. After each component is locally unit-tested, the component jar will be published to the Google Cloud for production hosting.
3. For Continuous development and integration, GitHub Actions and Google Cloud Build will be utilized to facilitate seamless integration.

Steps:

1. Develop components using Ktor skeleton

2. Unit test locally

3. Build Docker images

4. Deploy to Google Cloud Run

5. Automate with GitHub Actions

Tech Stack: Ktor, Kotlin, Docker, Firestore, Vertex AI

Monitoring: Prometheus + Grafana

CI/CD: GitHub Actions, Cloud Build