**Project Title:**

**LEARN-MATCH: AI-Powered Personalized Course Recommendation System” (L**earning **E**ngagement and **A**ccuracy through **R**ecommendations and **N**avigation with **M**achine **A**ssisted **T**argeted **C**ourse )

**1.Project Overview**

**Objective:**The primary goal of this project is to develop an AI-based course recommendation system that leverages K-MEANS clustering. The system aims to provide personalized course recommendations from various online and in-person platforms, saving users time by aggregating data across different educational resources with a user-friendly website interface. This AI system would take users’ interests and basic information at the very beginning page, therefore the recommendation will combine it with our database to rank the course during their search to realize the customized recommendation. The system included automated data scraping with API connection to guarantee the comprehensiveness of the recommendation.

**Scope:**The AI system will recommend courses based on user interests, previous interactions, and learning goals. It will utilize a webpage interface with a broad recommendation for recent popular courses and a searching bar to define their interested course based on the information they provided. The system will integrate data from multiple platforms, including Coursera, Educative, and EdX. Limitations include ensuring the system remains updated with new courses and managing the vast amount of data from different platforms.

**AI Techniques and Tools:**

The project will employ transfer learning and large language models (LLMs) to analyze user queries and provide relevant course suggestions. Specific tools and frameworks may include TensorFlow, PyTorch, and pre-trained models like GPT-3 or BERT. Therefore, K-MEANS could rapidly group the courses by the context learned by the LLMs.

**Stakeholders**

**Project Team:**

* Project Manager: oversee the entire project development.
* Data Scientist/ML engineer: develop and implemented K-MEANS algorithm for initial recommendation system and fine-tuning large language model for advanced recommendation system.
* Data Engineer: realize the data updating from all data providers and maintain the data safety and privacy.
* Backend Developer: develop and maintain the backend infrastructure of the recommendation system and ensure the scalability, safety and efficiency of server-side components.
* Frontend developer: design and develop a user-friendly interface for the course recommendation system and collaborate with the backend developer to integrate APIs.
* Education Expert: assist in identifying the key features for course recommendation and interpreting the users’ information.
* User experience test engineer:
* Attorney: Handle the possible ethical and legal risks.

**End Users:**

* students
* Lifelong learners
* Academic advisors
* College administrators
* Teacher and course instructor

**Other Stakeholders:**

* Educational intuitions: school, university…
* EdTech Company: potential founder
* Government and Education policy maker
* Educational investors and funding agency: resource founder
* IT company
* Online-course provider

**2. Computer Infrastructure Consideration**

**1. Project Needs Assessment**

* Define Objectives: Identify the primary goals of the recommendation system, such as improving course selection for users, enhancing user engagement, or increasing course completion rates.
* Identify Stakeholders: Determine who will use the system (students, instructors, course providers) and gather requirements from each group.
* User Research: Conduct surveys or interviews to understand user needs, preferences, and pain points regarding online learning platforms.
* Data Requirements: Assess the types of data needed, including user profiles, course descriptions, user interactions, ratings, and reviews.

**Resources:**

* User surveys and feedback forms
* Project management tools (e.g., Jira, Trello)
* Collaboration tools for stakeholder meetings (e.g., Zoom, Microsoft Teams)

**2. Hardware Requirements Planning**

* Identify Compute Needs: Determine the processing power required based on the algorithms used (K-MEANS and LLMs), data volume, and expected user load.
* Select Appropriate Hardware: Consider options such as GPUs for LLM training and inference and CPUs for K-MEANS calculations.
* Redundancy and Backup: Ensure redundancy in hardware to avoid system downtime and plan for data backups.

**Resources:**

* High-performance GPUs (e.g., NVIDIA RTX, A100) for training LLMs
* CPUs with high multi-core performance for K-MEANS
* On-premise servers or cloud-based virtual machines (AWS, Google Cloud, Azure)

**3. Software Environment Planning**

* Choose Programming Languages and Frameworks: Use Python for model development due to its rich ecosystem for AI and ML (libraries like scikit-learn for K-MEANS, TensorFlow/PyTorch for LLMs).
* Set Up Development and Production Environments: Use containerization (e.g., Docker) to create consistent environments for development, testing, and production.
* Version Control: Implement a version control system (e.g., Git) for managing code changes and collaboration.

**Resources:**

* Python and relevant libraries (scikit-learn, TensorFlow, PyTorch, pandas)
* Integrated Development Environment (IDE) like google Colab and Github.
* Docker for containerization

**4. Cloud Resources Planning**

* Evaluate Cloud Providers: Assess cloud providers (AWS, Google Cloud, Azure) based on pricing, scalability, and available services for ML.
* Select Services for Hosting and Processing: Use managed services like AWS S3 for data storage, AWS SageMaker for ML model training, and cloud databases (e.g., AWS RDS, Firestore).

**Cost Management**: Google Cloud Pricing Calculator.

**Resources:**

* AWS, Google Cloud, or Azure account for cloud resources
* S3 or Google Cloud Storage for data storage
* Managed ML services (e.g., AWS SageMaker, Google AI Platform)

**5. Scalability and Performance Planning**

**Strategies:**

* **Load Testing**: Simulate user load to test the system's performance and identify bottlenecks.
* **Optimize Algorithms**: Use techniques like dimensionality reduction or approximate nearest neighbor algorithms for K-MEANS to enhance performance.
* **Horizontal and Vertical Scaling**: Plan for scaling the infrastructure by adding more instances (horizontal) or upgrading existing instances (vertical) based on user demand.
* **Monitoring and Metrics**: Set up monitoring tools to track system performance, user interactions, and recommendation accuracy.

**Resources:**

* Load testing tools (e.g., Apache JMeter, Locust)
* Monitoring tools (e.g., Prometheus, Grafana)
* Infrastructure as Code (IaC) tools (e.g., Terraform, Ansible) for managing cloud resources

**Additional Considerations (might need):**

* **Caching**: Use caching mechanisms (like **Redis**) for frequently accessed data and precomputed recommendations.
* **Security**: Implement data encryption, secure API gateways, and user authentication mechanisms (e.g., **OAuth 2.0**) to protect sensitive user data.
* **Data Privacy Compliance**: Ensure adherence to GDPR, CCPA, and other relevant data privacy regulations, especially when dealing with personalized user data.

**3. Security, Privacy, and Ethics (Trustworthiness)**

**1. Problem Definition**

**Strategy: Inclusive Stakeholder Engagement**

* **Implementation**: Involve diverse stakeholders, including students, educators, and subject matter experts, during the problem definition stage. This can be done through surveys, interviews, or focus groups to ensure the system addresses a wide range of needs and avoids biases.
* **Example**: Conduct user interviews to gather insights on what factors students consider important for course recommendations (e.g., skill level, course length, learning style).

**2. Data Collection**

**Strategy: Data augmentation**

* **Technical Implementation**: Augment the under represented data in each column to prevent the potential clustering biasing**.**

**3. Model Development**

**Strategy: Bias Detection and Mitigation**

* **Technical Implementation**: Utilize tools like **AI Fairness 360** or **Fairlearn** to assess and mitigate bias in the K-MEANS model. This can involve analyzing the distribution of recommendations across different demographic groups and adjusting the model to reduce bias.
* **Example**: Evaluate the model's recommendations to ensure that underrepresented groups receive equitable course suggestions. If a particular demographic is disproportionately underrepresented in the recommendations, consider reweighting the input data accordingly.

**Strategy: Hyperparameter tunning with different method**

### Since the K is the hyperparameter that will control the clustering result. Too much cluster will dramatically increase the complexity of model. We use the matched result from inertia method and silhoutte score to make the optimal number of cluster is consistent.

**Strategy: Clustering result visualization**

Visualize the clustering result so see if the cluster are well-separated

**4. Deployment**

**Strategy: Bias detecting**

Detecting the model bias by get the number of sample belongs to each cluster so make sure the number id not high imbalanced.

**Strategy : Comparing the execution time and best K value from different model**

Use silhoutte score as the metric to quantify the execution time for K-means and Mini Batch K-means to make sure out choice is optimal and efficient.

**5. Monitoring/Maintenance**

**Strategy: Continuous Performance and Ethical Auditing**

* **Implementation**: Regularly audit the system for performance metrics (accuracy, user satisfaction) and ethical considerations (bias, transparency). This can include user feedback loops and monitoring for unusual patterns in recommendations.
* **Example**: Implement feedback mechanisms within the application where users can report unsatisfactory recommendations. Use this feedback to adjust the model periodically and conduct audits for potential biases or ethical concerns.

**4.Human-Computer Interaction (HCI) Considerations**:

**1. Understanding User Requirements**

**Strategy**: Conduct short surveys at the very first page to collect what users are looking for in a course recommendation system, their previous experience and expected outcome after learning. For example, preferred learning styles, desired skills, budget, time commitment).

**Tools:**

Google Forms or Typeform for conducting surveys.

Data Analytics Tools (e.g., Pandas, Matplotlib) for analyzing survey results.

**2. Creating Personas and Scenarios**

**Strategy**: Build detailed user personas and scenarios for diverse end-users to achieve the complexity of the AI system.

**Persona 1: New Learner - Sarah, 22**

* **Background**: Sarah is a junior college graduate looking to develop new skills to enhance her resume and career prospects. She is unfamiliar with online learning platforms but is eager to learn.
* **Goals**:
  + Find beginner-friendly courses in digital marketing.
  + Build skills that can be applied immediately in entry-level jobs.
  + Look for affordable and free courses as she’s on a limited budget.
* **Challenges**:
  + Navigating multiple platforms (Edx, Educative, Coursera) to find courses.
  + Overwhelmed by the variety of course options and doesn't know where to start.
* **Scenario**: Sarah opens the course recommendation system and inputs her goal of learning digital marketing. The AI system recommends a curated list of beginner-level courses with affordable pricing, emphasizing those with short completion times and job-relevant content. The system highlights her preferred learning style based on quick feedback questions.

**Tools**:

* Survey forms to understand her preferences (budget, skills, time availability).

**Persona 2: Mid-Career Professional - John, 35**

* **Background**: John is a software engineer with 10 years of experience. He wants to transition into a managerial role, which requires learning project management and leadership skills.
* **Goals**:
  + Find intermediate to advanced courses on leadership, management, and soft skills.
  + Learn part-time while managing a full-time job and family.
  + Gain certifications to validate new skills for a promotion.
* **Challenges**:
  + Balancing work, family, and study time.
  + Finding courses that fit his intermediate skill level and aren't too basic.
* **Scenario**: John logs into the course recommendation system, specifies his current role and future goals, and requests courses that match his schedule (short, flexible, part-time). The system offers a mix of self-paced and instructor-led courses that can be completed on weekends, highlighting certifications and career transition paths.

**Tools**:

* Scheduling optimization algorithms to suggest courses that fit into busy schedules.
* LinkedIn API integration to suggest courses that align with common promotion paths in his field.

**Persona 3: Lifelong Learner - Maria, 45**

* **Background**: Maria is a curious learner who enjoys expanding her knowledge in various subjects like philosophy, history, and literature. She takes online courses purely for personal enrichment.
* **Goals**:
  + Explore different topics of interest, such as art history and modern philosophy.
  + Engage in learning without a strict schedule or certification requirements.
  + Discover niche topics across different platforms.
* **Challenges**:
  + Finding non-career-focused, niche courses.
  + Keeping track of completed courses and discovering new ones in diverse fields.
* **Scenario**: Maria inputs her general interests into the system and selects topics from various fields. The system recommends courses based on her interests, offering suggestions for both long and short courses. Maria is also given options to follow specific educators or institutions she prefers.

**Tools**:

* Interest-based recommendation algorithms.
* Recommendation system filters for "personal interest" and "non-certification" courses.

**Persona 4: Career Changer - Ahmed, 29**

* **Background**: Ahmed is a mechanical engineer who wants to switch to data science. He has no background in programming but is highly motivated to make the switch.
* **Goals**:
  + Find structured, sequential learning paths to build foundational skills in data science.
  + Seek courses with hands-on projects and career services for mentorship.
  + Look for job placement support after course completion.
* **Challenges**:
  + Overcoming the steep learning curve in programming and data analysis.
  + Finding a clear path from beginner to employable skills.
* **Scenario**: Ahmed selects "Career Change to Data Science" in the system, which recommends a step-by-step learning path, starting from Python programming to advanced machine learning. The system suggests courses with project-based learning and career coaching.

**Tools**:

* Learning path design algorithms to suggest structured course sequences.
* Skill assessments to gauge current knowledge and recommend appropriate starting points.

**Persona 5: Student - Emily, 19**

* **Background**: Emily is a college student pursuing a degree in business. She wants to take additional online courses to supplement her academic studies and broaden her skill set.
* **Goals**:
  + Find courses that complement her business degree (e.g., finance, entrepreneurship).
  + Earn certifications that can help her stand out to potential employers.
  + Balance online learning with her full-time studies.
* **Challenges**:
  + Limited time to take online courses.
  + Aligning course content with her current academic curriculum.
* **Scenario**: Emily uses the system to find courses that align with her degree. The system suggests business-related courses that offer certifications and short-term courses to fit her study schedule. It also integrates study reminders to keep her on track during the semester.

**Tools**:

* Curriculum alignment algorithms to match courses with academic programs.
* Learning reminders and scheduling integration to help balance time.

**Persona 6: Instructor/Trainer - David, 40**

* **Background**: David is an experienced instructor in web development and wants to recommend the best online courses to his students. He also looks for courses to stay updated in his field.
* **Goals**:
  + Find high-quality courses to recommend to his students for extra practice.
  + Stay up-to-date with the latest trends and technologies in web development.
* **Challenges**:
  + Filtering through courses to find the most relevant ones for different skill levels.
  + Identifying up-to-date courses in a rapidly changing field.
* **Scenario**: David searches for web development courses and receives a list tailored by student level (beginner, intermediate, advanced). The system flags newly updated courses and provides reviews by other instructors to help him choose.

**Tools**:

* Course filtering algorithms by skill level and update recency.
* Instructor reviews and peer feedback system to improve course selection.

**3. Conducting Task Analysis**

**Strategy**: Break down the tasks that users need to accomplish, such as searching for courses, filtering results, and saving preferences. Analyze how users interact with each task to design more intuitive interfaces and processes.

* **Tools**:
  + Figma or Adobe XD for task-flow diagram creation.
  + Hotjar or FullStory to collect user interaction data.

**4. Identifying Accessibility Requirements**

**Strategy**: Conduct an accessibility audit based on WCAG guidelines to ensure that the course recommendation system is usable by people with disabilities, such as those with visual impairments or motor skill challenges.

* **Tools**:
  + WAVE Accessibility Tool or axe DevTools for accessibility testing.
  + Screen readers like NVDA for testing screen reader compatibility.

**5. Outlining Usability Goals**

**Strategy**: Define specific usability goals like ease of navigation, time to complete tasks, and user satisfaction levels, based on user requirements and expected interaction patterns.

* **Tools**:
  + UsabilityHub or Optimal Workshop for usability testing and gathering user feedback.
  + Google Analytics to track metrics like time on page or drop-off points.

**5. Risk Management Strategy**

**Data Collection**

**Strategy:** Data augmentation and enumerating

**Technical Implementation:**

Apply the data augmentation strategies to better represent the imbalanced data which might cause the potential model biasing .

Different to other feature, the enumerate of course title need to maintain the semantic relevancy after enumerating, so we identify the text similarity using TF-IDF with K-means clustering to assign a unique value for each title.

**Outlier and Centroid initialization**

Since K-means itself might be sensitive to the outlier and the initialization of centroid. We need to detect the outlier first and test the stability of model by different initialization and monitor the standard deviation or the difference of cluster assigned.

**Strategy:** run the model with different centroid initialization

Detect outlier with Z score

**Model Development**

**Strategy:** Bias Detection

**Technical Implementation:** count the number of samples in each cluster to see if the sample in each cluster is highly imbalanced.

**Deployment**

**Strategy:** evaluate the model performance from the user feedback

**Technical Implementation:** After we display the top3 recommended course for the user. We take the users feedback to evaluated the effectiveness of our model and record their response. If the record shows we keep receiving the unsatisfied feedback. We need interruption and recheck for the model

**Monitoring/Maintenance**

**Strategy:** Continuous Performance and Ethical Auditing

We established a comprehensive system for continuous monitoring of performance metrics and ethical considerations to maintain the integrity and effectiveness of LEARN-MATCH.

**Technical Implementation:** We developed a custom monitoring dashboard using Grafana and Prometheus to track key performance indicators (KPIs) such as recommendation accuracy, user engagement, and system response times. We implemented automated alerts using PagerDuty for anomalies in these metrics. Our application includes a feedback mechanism where users can report unsatisfactory recommendations, which feeds into our periodic model adjustment process. We use Apache Airflow to schedule regular audits for potential biases and ethical concerns, with results automatically compiled into reports for review by our ethics board.

**Residual Risk Assessment**

To conduct a residual risk assessment for the LEARN-MATCH project:

1. K-means is assume the data is sphere-distributed, which might not be true for this dataset
2. Assess the likelihood and impact of each hazard:

|  |  |  |  |
| --- | --- | --- | --- |
| **Hazard** | **Likelihood** | **Impact** | **Initial Risk Level** |
| Data breaches | Medium | High | High |
| Biased recommendations | Medium | Medium | Medium |
| System downtime | Low | Medium | Low |
| Inaccurate course information | Medium | Medium | Medium |
| User privacy violations | Low | High | Medium |

Additional mitigation strategies:

* Conduct regular penetration testing to identify potential security vulnerabilities.
* Implement a bug bounty program to encourage the reporting of security issues.
* Develop a comprehensive incident response plan for potential data breaches.
* Establish a diverse ethics board to regularly review the system's recommendations and decision-making processes.

**6. Data Collection Management and Report**

**1. Data Type**

LEARN-MATCH collects various types of data, including:

* User profiles (interests, learning goals, previous courses)
* Course information (descriptions, ratings, reviews)
* User interactions (clicks, time spent, completion rates)

**2. Data Collection Methods**

We employ multiple methods to collect comprehensive data:

* API integrations with course platforms (Coursera, Educative, EdX)
* User input through our web interface
* Automated web scraping for course updates

**3. Compliance with Legal Frameworks**

We ensure compliance with GDPR, CCPA, and other relevant data protection regulations. We have implemented data minimization principles and obtain explicit user consent for data collection.

**4. Data Ownership**

Our terms of service clearly define data ownership. Users retain ownership of their personal data, while we may use aggregated and anonymized data for system improvement.

**5. Metadata**

We have implemented a metadata management system to track:

* Data source
* Collection date
* Data quality metrics
* Usage restrictions

**6. Versioning**

We use Git for code version control and a data versioning tool for datasets to track changes and enable rollbacks if needed.

**7. Data Preprocessing, Augmentation, and Synthesis**

Our data pipeline includes:

* Preprocessing: Cleaning and normalizing data, handling missing values
* Augmentation: Enriching course data with additional information (e.g., skill tags)
* Synthesis: Generating synthetic user profiles for testing and development

**8. Report on Risk Management in Data Collection**

We have implemented several measures to manage risks in data collection:

* Encryption for data in transit and at rest
* Regular audits of data access logs
* Periodic vulnerability assessments
* Established data breach response plan

**9. Report on Trustworthiness in Data Collection**

To ensure trustworthiness in our data collection practices, we:

* Maintain transparency in our data collection methods
* Provide users with control over their data (access, rectification, deletion)
* Regularly update our privacy policies and obtain renewed consent
* Implement data quality checks to ensure accuracy and relevance of collected information
* By implementing these strategies and considerations, LEARN-MATCH has established a robust framework for risk management and data collection, ensuring the development of a trustworthy and effective AI-powered course recommendation system.

By implementing these strategies and considerations, LEARN-MATCH has established a robust framework for risk management and data collection, ensuring the development of a trustworthy and effective AI-powered course recommendation system.

**7. Model Development and Evaluation**

**1. Model development**

**Algorithm selection:** since the recommendation system is a dataset without any label. We use K-means as the clustering algorithm to identified the similarity between different course and recommend user the course based on the Euclidian distance of other data point to use input data points. We also compare the K-means with the mini batch k-means to address the efficiency of K-means.

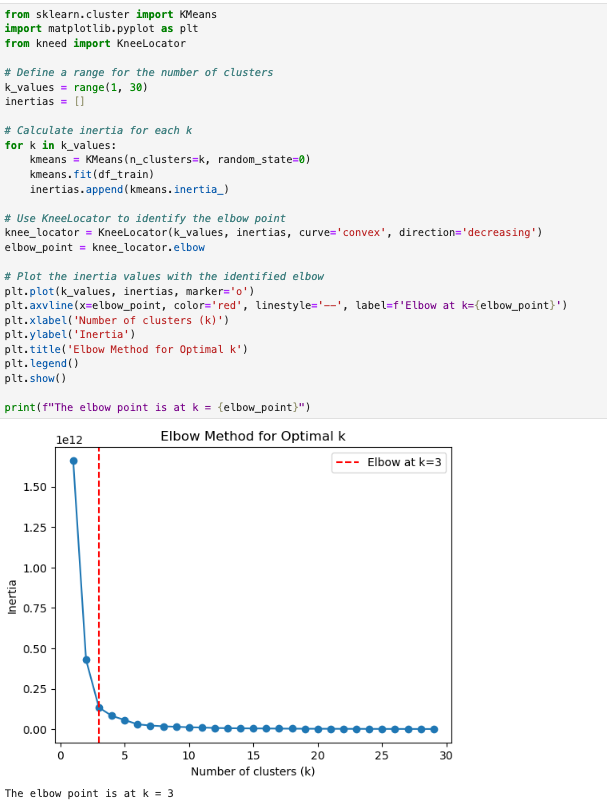
**Feature engineering**: from all the features provided in the data resources, we will select the most relevant feature such as “title”, ‘language’, “topic”, “difficulty level”, and discard “link” kind of features which is invalid user input and can’t provide useful information for the recommendation.

**Model complexity and architecture:** K-means clustering complexity comes from three part: the number of data point n, the number of clusters k and the dimensionality of each data point d. So the overall complexity is O(n\*k\*d). Therefore, the cost of clustering will increase with increase of datapoint, too much cluster or too much features.

**2 and 3. Model Training and evaluation**

**Since K-means is a unsupervised learning model. The model training is accompany with the model evaluation. We adapt the clustering performance to select the best hyperparameter to build the model.**

### **Hyperparameter tuning an training: The hyperparameter tuning is conducted with two methods to ensure the result comes from each of them are consistent and valid. We use inertia method and** silhoutte score to find the optimal K:

A screen shot of a graph

Description automatically generated

Both method gives 3 as the optimal K values.

**Evaluation:** Besides evaluate with inertia method and silhoutte score. We also visualize the clustering result to make sure K=3 could give a well-separated cluster. Cross-validation is not applicable here.

A diagram of a colorful circle

Description automatically generated with medium confidence

**4. Implement of trustworthiness and Risk management**

Since there is overlap of trustworthiness and risk management, we report them below together:

**1. sensitivity to initial centroids: report the score with different initialization**[**¶**](http://localhost:8888/notebooks/Playground_Risk_Trustworthy_HCI.ipynb#1.-sensitivity-to-initial-centroids:-report-the-score-with-different-initialization)

* Risk: The performance of K-Means can be highly sensitive to the initial placement of centroids. Different initializations can lead to different clustering results, especially if the clusters have varying sizes or densities.
* Solution: Use K-Means++ for smarter initialization or run the algorithm multiple times and select the best result.

A screen shot of a computer program

Description automatically generated

**2. Sensitive to outlier (use Z score)**

A screenshot of a computer code

Description automatically generated

**3. Label stability**

Adjusted Rand Index (ARI), which measures the similarity between two clustering result:

A screenshot of a computer program

Description automatically generated

### **4.** **Choice of K (same as hyperparameter tuning)**

**5.** **Compring the performance and excution time of different model**

(K-means and mini batch K-means)

A graph of a graph with blue and orange lines

Description automatically generated

### **6. Bias detecting**

Check the number of datapoints in each cluster:

**A screenshot of a computer code

Description automatically generated**

**7. Transparency**

User friend interface with a display of each process step and process progress:

**A screenshot of a computer

Description automatically generatedA screenshot of a video

Description automatically generated**

**5. Apply HCI Principles in AI Model Development**

Develop Interactive Prototypes

* Use Streamlit to give a constraint on use input: such the data type. Using button to constraint the valid input option to avoid error occurrence.
* Save the use input to our database and perform clustering use saved model
* Showing user the each steps our app and display the processing progress on the interface.
* Display the top3 recommended course use the Euclidian distance after clustering.
* Delete the use input from clustering database to make sure the consistency.
* Get user feedback like unsatisfied or satisfied as feedback.

Design Transparent Interfaces

**A screenshot of a computer

Description automatically generatedA screenshot of a video

Description automatically generated**

**User** **input section and data saving:**

A computer screen shot of a program

Description automatically generatedA computer screen shot of text

Description automatically generated

Data processing:

A screenshot of a computer program

Description automatically generated

Clustering and result display:

A screenshot of a computer program

Description automatically generated

**Create Feedback Mechanisms**

After display the recommendation result, we will offer a further button for user to select: Satisfied result or Unsatisfied result. The feedback will saved as a time sensitive data to a csv file for further performance monitoring

A screenshot of a computer program

Description automatically generated

**8. Deployment and Testing Management Plan**

**1. Deployment environment selection**

**Local deployment**: since K-means clustering is not a super complicated model, we choose to deploy the model use streamlit in local machine. Since we only take one user input at a time, the system is not required to have great scalability. Therefore, a python script with using streamlit interface could quickly deliver the AI system and collect the user feedback and monitoring the system as well.

**2. Deployment strategy**

**Tools: docker** (Ensure consistent deployment environments and simplify scaling. )

**Framework:** The docker is chose as the containerization tools for our recommendation system which ensure the stable running, logging and performance monitoring by integrate it with Grafana and Prometheus. Grafana will be used for the visualization of our designed system performance metric and Promethous could provide monitoring and alerting from collected metrics.

**3. Security and Compliance in Deployment (Trustworthiness and Risk Management)**

**Reproducible result:** To ensure the model is not drifting and biasing with more and more user input to the recommendation system. We delete the user input data point from the clustering dataset to keep the original dataset clean. Therefore, every time the clustering will only be altered due the one user input respect the to original dataset.

**A screen shot of a computer code

Description automatically generated**

**User input restriction:** By using streamlit, we provide several button for each of option to collect the use input to avoid the invalid use input and potential system error for the deployment stage.

A screen shot of a computer program

Description automatically generated

**9. Evaluation, Monitoring and Maintenance Plan**

**1. System evaluation and monitoring**

**Tool:** Grafana and Promethous

**Monitoring Metrics**: a. immediate monitoring: total runtime and clustering runtime

b: delayed monitoring: collect time sensitive user feedback to model drifting detection.

**Framework**: Within the docker container, the Grafana will visualize the metric as we defind above and Promethous could provide monitoring and alerting based on the metrics. For example, since we only take one user input as a time, the total runtime and clustering runtime could be very stable. If any suspicious runtime has been detected, it means our AI system is under issue which Promethous could alert the developer for further checking. The delayed monitoring is based on user feedback with recorded time. The time could help us to analyze in a certain time range how many “satisfied” and “unsatisfied” feedback has been collect. If at certain time range, we keep getting “unsatisfied” feedback, it means we either need to improve our database or there is wrong clustering result that displayed to the user.

**2. Feedback collection and continuous improvement**

**Feedback mechanism:** At the end of each clustering cycle, we will use streamlit to collect the user feedback using the button “satisfied” or “unsatisfied” with the recorded time into a .csv file, as the screenshot below.

A screenshot of a computer program

Description automatically generated

**Continuous improvement:** we plan to use a dataset with more features for a advanced clustering by taking more detailed user input if the user select the “unsatisfied” button. By this end, user could provide the system with more information, but the computational cost will also be increased. We will considering this improved part as a non-free option.

**3. Maintenance and Compliance audit**

**Potential model biasing:** monitoring the assigned cluster with use input to see if any highly imbalanced outcome has appeared.

**Data protection:** delete the user input after the clustering to prevent the potential data leaking and privacy risk.

**Code availability:** [**https://github.com/zhiyuwang0911/AIS\_project/tree/main/Zhiyu**](https://github.com/zhiyuwang0911/AIS_project/tree/main/Zhiyu)