Q1. System design: Based on the above information, describe the KPI that the business should track.

Key Performance Indicators (KPIs) for System Design:

Time Efficiency (Time-to-Extract):

Definition: Measure the time taken to extract diseases and treatments from a given set of clinical notes.

Rationale: The primary goal is to reduce the overall time spent on the extraction process, contributing to increased efficiency.

Accuracy of Extraction:

Definition: Evaluate the correctness of the automated extraction system by comparing its results to those obtained by trained personnel.

Rationale: Accuracy is crucial to ensure that the automated system provides reliable information, reducing manual errors.

Scalability Metrics:

Definition: Assess the system's performance and resource utilization under varying workloads, particularly as the volume of clinical notes increases.

Rationale: Scalability is vital to accommodate the growing user base and increasing amounts of clinical data.

Cost Reduction and Efficiency:

Definition: Quantify the reduction in costs associated with the elimination of the data-entry team.

Rationale: Cost reduction is a significant business driver, and efficiency gains contribute to achieving this goal.

User Satisfaction and Feedback:

Definition: Gather feedback from users involved in the validation process to assess their satisfaction with the accuracy and efficiency of the automated system.

Rationale: User satisfaction is critical for the successful adoption of the automated system. User feedback provides insights into the system's real-world performance.

Model Performance Metrics:

Definition: Track metrics related to the performance of the NLP model, such as precision, recall, and F1 score.

Rationale: Understanding the model's performance is essential for continuous improvement and ensuring that it effectively captures diseases and treatments in clinical notes.

Adaptability and Continuous Improvement:

Definition: Implement mechanisms for continuous improvement of the NLP model based on feedback and evolving clinical language.

Rationale: The system should be adaptable to changes in clinical language and capable of evolving to handle new terms and expressions.

Integration Efficiency:

Definition: Assess the seamless integration of the NLP model into BeHealthy's workflow without disruptions.

Rationale: The efficient integration of the automated extraction system ensures a smooth workflow and reduces friction in the overall process.

Tracking these KPIs will provide BeHealthy with comprehensive insights into the performance and impact of the automated system for extracting structured information from clinical notes.

Q2. System Design: Your company has decided to build an MLOps system. What advantages would you get by opting to build an MLOps system?

Building an MLOps system in-house offers several advantages:

Customization:

Advantage: Building an MLOps system allows for tailored customization to meet the specific needs and requirements of the company. The system can be designed to align closely with the organization's unique workflows, processes, and infrastructure.

Flexibility:

Advantage: In-house development provides flexibility to adapt and modify the MLOps system as the organization evolves. It allows for rapid adjustments to changing business requirements, technology stacks, and industry standards.

Integration with Existing Infrastructure:

Advantage: An internally built MLOps system can be seamlessly integrated with the company's existing infrastructure, tools, and systems. This integration facilitates a cohesive and efficient workflow across the entire organization.

Security and Compliance:

Advantage: Building an MLOps system in-house enables the organization to have direct control over security measures and compliance. It allows for the implementation of specific security protocols and measures to safeguard sensitive data and ensure regulatory compliance.

Ownership and Control:

Advantage: By building the MLOps system internally, the organization retains ownership and control over its development, maintenance, and updates. This level of control is crucial for strategic decision-making and long-term sustainability.

Learning and Skill Development:

Advantage: The process of building an MLOps system provides valuable opportunities for the internal team to learn and develop expertise in MLOps practices, tools, and technologies. This knowledge can be applied to future projects.

Cost Efficiency (Over the Long Term):

Advantage: While the initial development phase may incur costs, in the long term, an internally built MLOps system may be cost-effective. This is especially true if the organization plans to scale its machine learning operations over time, as the costs associated with vendor solutions may increase with usage.

Alignment with Business Goals:

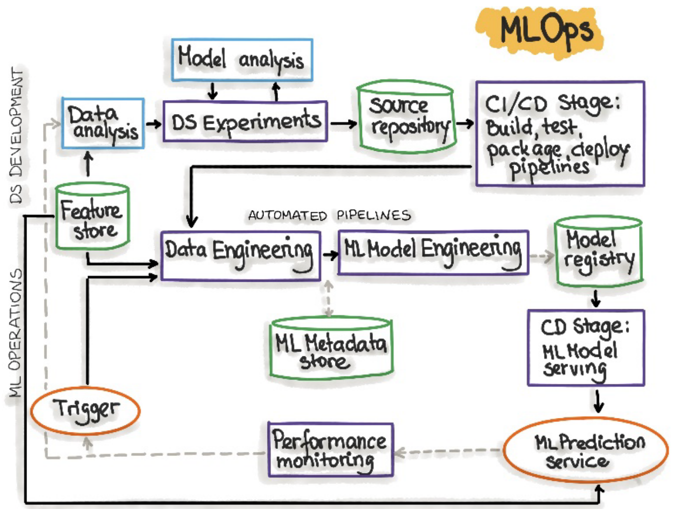
Advantage: Building an MLOps system ensures that the system is aligned with the specific business goals and strategies of the organization. It allows for a more direct translation of business objectives into technical solutions.

Continuous Improvement:

Advantage: In-house development facilitates continuous improvement and optimization of the MLOps system based on real-time feedback and evolving business needs. This agility is valuable for staying competitive and innovative.

While building an MLOps system in-house offers these advantages, it's essential to carefully weigh them against potential challenges, such as the initial development timeline and resource requirements. Additionally, organizations should consider factors like the availability of skilled personnel and the opportunity cost of dedicating resources to in-house development.

Q3. System design: You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring. For the given problem, create an ML system design (diagram).



Components and Stages:

Data Ingestion:

Raw data from various sources is ingested into the system. This could include clinical notes, patient data, and other relevant information.

Data Preprocessing:

Data preprocessing involves cleaning, transforming, and preparing the raw data for model training. This stage ensures that the data is in a suitable format for the ML models.

Experiment Tracking:

Experiment tracking tools, such as MLflow or TensorBoard, are used to log and monitor experiments. This includes recording hyperparameters, metrics, and model artifacts during the model training process.

Feature Engineering:

Feature engineering involves selecting and transforming features to improve model performance. It is an essential step in preparing the data for machine learning algorithms.

Model Training:

ML models, such as CRF models for Named Entity Recognition, are trained using the preprocessed data. The trained models are then saved as artifacts.

Model Evaluation:

The performance of the trained models is evaluated using validation datasets. Metrics such as precision, recall, and F1 score are calculated to assess the model's accuracy.

Model Deployment:

Deployed models are exposed as endpoints accessible by other parts of the system. Tools like AWS SageMaker or TensorFlow Serving can be used for model deployment.

API Gateway:

An API gateway manages the communication between different components. It serves as an interface for external services to interact with the deployed ML models.

User Interface (UI):

A user interface allows authorized users (e.g., data scientists, medical professionals) to interact with the system, input clinical notes, and view results.

Monitoring:

Model and system monitoring tools, such as Prometheus or Grafana, are employed to track the health and performance of the deployed models. This includes monitoring model drift, data drift, and overall system stability.

Logging and Logging Aggregation:

Logs generated during model deployment and monitoring are collected and aggregated using logging tools like ELK Stack (Elasticsearch, Logstash, Kibana).

Alerting System:

An alerting system is configured to notify relevant stakeholders in case of anomalies, errors, or performance issues detected during monitoring.

Feedback Loop:

Feedback from the monitoring system and user interactions is used to continuously improve the models. This feedback loop contributes to ongoing model optimization.

It's important to note that the specific tools and technologies used in each stage may vary based on the organization's preferences, infrastructure, and the nature of the problem being addressed. The diagram provides a high-level overview of the ML system's architecture, encompassing key stages from data ingestion to model deployment and monitoring.

Q4. System design: After creating the architecture, please specify your reasons for choosing the specific tools you chose for the use case.

Certainly, the choice of tools in an ML system design depends on various factors, including the specific requirements of the use case, organizational preferences, scalability needs, and the expertise of the team. Here are some considerations for the tools chosen in the ML system architecture described earlier:

MLflow for Experiment Tracking:

Reasons:

Open Source and Vendor Agnostic: MLflow is an open-source platform that supports a variety of machine learning libraries and platforms, providing flexibility and avoiding vendor lock-in.

Experiment Tracking: MLflow's tracking capabilities allow for easy logging and monitoring of experiments, which is crucial for managing and reproducing machine learning workflows.

Model Registry: MLflow includes a model registry for managing and versioning models, enhancing collaboration and model governance.

TensorFlow Serving for Model Deployment:

Reasons:

TensorFlow Integration: If the ML models are developed using TensorFlow, TensorFlow Serving is a natural choice for deploying TensorFlow models, ensuring compatibility and optimized performance.

Scalability: TensorFlow Serving is designed for serving machine learning models in production with a focus on scalability and low-latency inference.

API Gateway (e.g., AWS API Gateway):

Reasons:

Seamless Integration: An API gateway provides a unified entry point for managing and securing API interactions. AWS API Gateway, for example, integrates seamlessly with other AWS services, facilitating a cohesive architecture.

Authorization and Authentication: API gateways offer features for controlling access, handling authentication, and ensuring the security of API endpoints.

Elasticsearch, Logstash, Kibana (ELK Stack) for Logging and Logging Aggregation:

Reasons:

Comprehensive Logging Solution: ELK Stack provides a comprehensive solution for log management, offering Elasticsearch for storage, Logstash for log processing, and Kibana for visualization.

Scalability and Flexibility: ELK Stack is scalable and flexible, accommodating varying logging needs and facilitating centralized log analysis.

Prometheus and Grafana for Monitoring:

Reasons:

Prometheus for Metrics Collection: Prometheus is designed for efficient metrics collection, making it well-suited for monitoring the health and performance of systems.

Grafana for Visualization: Grafana complements Prometheus by providing powerful visualization and alerting capabilities, enhancing the monitoring experience.

These tool choices are just examples, and the actual tools used would depend on the specific requirements, existing infrastructure, and organizational preferences. Additionally, considerations such as cost, ease of use, and community support play a role in tool selection. The key is to ensure that the chosen tools align with the goals of the ML system, promote efficiency, and facilitate scalability and maintainability.

Q5. Workflow of the solution:

You must specify the steps to be taken to build such a system end to end.

The steps should mention the tools used in each component and how they are connected with one another to solve the problem.

Broadly, the workflow should include the following. Be more comprehensive of each step that is involved here.

Data and model experimentation

Automation of data pipeline

Automation of the training pipeline

Automation of inference pipeline

Continuous monitoring pipeline

The workflow should ALSO explain the actions to be taken under the following conditions.

After you deployed the model, you noticed that there was a sudden increase in the drift due to a shift in data.

What component/pipeline will be triggered if there is any drift detected? What if the drift detected is beyond an acceptable threshold?

What component/pipeline will be triggered if you have additional annotated data?

How will you ensure the new data you are getting is in the correct format that the inference pipeline takes?

Workflow for Building an End-to-End ML System:

Data and Model Experimentation:

Tools Used:

Data Ingestion: Custom scripts, AWS S3, or other data storage services.

Data Preprocessing: Pandas, Scikit-learn for initial preprocessing.

Experiment Tracking: MLflow or TensorBoard.

Workflow:

Ingest raw clinical notes data.

Preprocess data for model training.

Use MLflow for experiment tracking, recording hyperparameters, metrics, and model artifacts during model training.

Iterate on feature engineering and model development.

Automation of Data Pipeline:

Tools Used:

Apache Airflow, AWS Glue, or similar ETL tools.

Workflow:

Define ETL tasks for data preprocessing and feature engineering.

Schedule periodic data updates or real-time ingestion.

Ensure data is cleansed, transformed, and available for model training.

Automation of Training Pipeline:

Tools Used:

MLflow for experiment tracking.

TensorFlow or other ML libraries for model training.

Workflow:

Trigger model training based on new data.

Log experiments and track model performance using MLflow.

Save trained models as artifacts.

Automation of Inference Pipeline:

Tools Used:

TensorFlow Serving for model deployment.

API Gateway (e.g., AWS API Gateway) for managing endpoints.

Workflow:

Deploy the trained model using TensorFlow Serving.

Expose the model through an API Gateway for real-time inference.

Ensure proper authorization and authentication.

Continuous Monitoring Pipeline:

Tools Used:

Prometheus and Grafana for monitoring.

ELK Stack for logging and logging aggregation.

Workflow:

Set up monitoring for model and system metrics using Prometheus and Grafana.

Log events and errors using ELK Stack for centralized log analysis.

Configure alerting for anomalies and performance issues.

Handling Drift:

Conditions:

Drift Detected:

Trigger the continuous monitoring pipeline.

If drift is within an acceptable threshold, monitor and log the event.

If drift exceeds the threshold, trigger an alert and initiate the retraining pipeline.

Handling Additional Annotated Data:

Conditions:

Additional Annotated Data Received:

Ingest the new annotated data into the data pipeline.

Re-run the training pipeline with the augmented dataset.

Update the deployed model with the retrained version.

Ensuring Data Format Compatibility:

Conditions:

New Data Format Verification:

Implement data validation checks in the data pipeline.

Use schema validation or data profiling to ensure incoming data conforms to the expected format.

Log and alert if data format mismatches are detected.

This comprehensive workflow covers the entire lifecycle of an ML system, from data and model experimentation to continuous monitoring. It incorporates automation at various stages to ensure efficiency, scalability, and adaptability to changes in data and model performance. The handling of drift, additional annotated data, and data format compatibility issues is integrated into the workflow for proactive and adaptive management.