MLOPS_ NLP_CASE STUDY

Bharathveer KS

MLOps - NLP - Case Study

Problem Statement

You are working as a data scientist for a healthtech company called BeHealthy. It is a late-stage startup.

Be Healthy provides a web-based platform for doctors to list their services and manage patient interactions. It provides various services for patients, such as booking interactions with doctors and ordering medicines online. (Similar to 1mg, PharmEasy)

Here, doctors can easily manage appointments, track past medical records and reports and give e-prescriptions.

We can assume that BeHealthy has enrolled thousands of doctors across the major cities in India. You can also assume that millions of patients are using BeHealthy's services; hence, a lot of free-text clinical notes would be present in the e-prescriptions.

BeHealthy would want to convert these free-text clinical notes to structured information for analytics purposes. You have seen such a problem in your previous courses on NLP.

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

The Key Performance Indicator (KPI) that the business should track in the context of the BeHealthy problem statement is the accuracy of the automated process in extracting diseases and treatments from free text. This KPI would measure how well the automated process is performing in correctly identifying the diseases and treatments from the clinical notes compared to manually extracted information.

The accuracy of the automated process should be high enough to ensure that there are no errors in the extracted data, and it should be continually monitored and improved. By tracking this KPI, BeHealthy can ensure that their automated process is meeting the desired business goals of reducing man-hours, eliminating the need for a data-entry team, reducing manual errors, and scaling up the size of the data without increasing the size of the data-entry team.

To summarize, the KPI to be tracked is the accuracy of disease and treatment extraction from free text, ensuring the effectiveness and efficiency of the automated process in achieving the business objectives.

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 (Machine Learning Operations) system offers several advantages for companies developing and deploying machine learning models:

1. **Improved model performance:** MLOps systems ensure that machine learning models are built and deployed efficiently and effectively, leading to enhanced model performance. By streamlining the development and deployment processes, MLOps helps optimize models for better accuracy and effectiveness.

2. **Faster model deployment:** With MLOps practices in place, models can be deployed more rapidly and efficiently. Automation and standardized processes reduce the time it takes to transition a model from development to production, enabling quicker time-to-market and responsiveness to business needs.

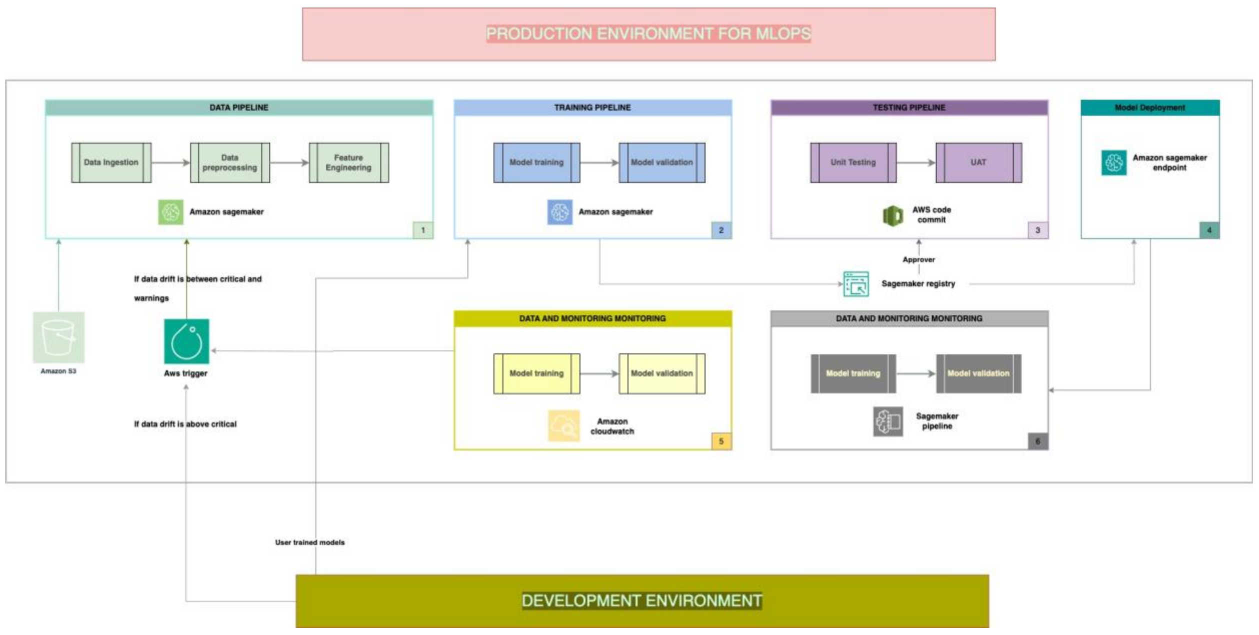
3. **Better model governance and version control:** MLOps systems provide robust version control and governance mechanisms, ensuring that models are developed, deployed, and maintained in a structured and compliant manner. This helps in tracking changes, managing model versions, and ensuring regulatory compliance and data privacy.

4. **Increased collaboration:** MLOps fosters better collaboration among data scientists, developers, and IT operations teams. By implementing standardized workflows, tools, and communication channels, MLOps promotes collaboration and teamwork across different functions involved in the machine learning lifecycle. This leads to improved communication, knowledge sharing, and synergy, ultimately enhancing the quality and efficiency of model development and deployment processes.

Overall, investing in an MLOps system can significantly improve the effectiveness, efficiency, and governance of machine learning initiatives within an organization, enabling faster innovation and better alignment with business objectives.

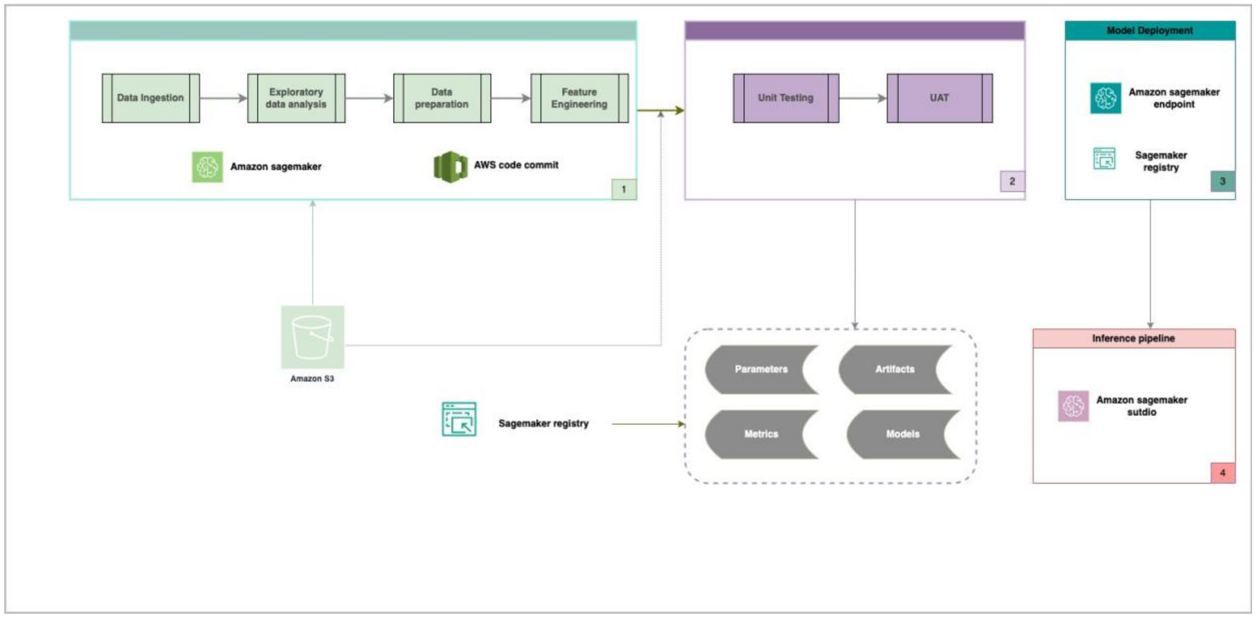
Q3. System design: You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring. For the given problem, create an ML system design (diagram).

Production environment



Development environment

DEVELOPMENT ENVIRONMENT



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

Choosing managed cloud services like AWS SageMaker for the BeHealthy system architecture provides several advantages aligned with the startup's late-stage status and the specific use case requirements:

1. **Easy deployment and management:** Managed cloud services simplify the deployment and management of the entire system. This is crucial for a startup like BeHealthy, as it reduces the need for dedicated DevOps personnel and saves time and resources, allowing the team to focus on core business objectives rather than infrastructure management.

2. **Cost-effectiveness:** Managed cloud services offer a pay-as-you-go pricing model, which can be more cost-effective than traditional on-premises infrastructure. This is particularly important for a startup like BeHealthy, which may have limited resources and needs to optimize costs while scaling its operations.

3**. Scalability:** Managed cloud services provide scalability, allowing the system to scale up or down based on the volume of data and user traffic. This ensures that BeHealthy can handle an increase in demand for its services without experiencing downtime or performance issues, enabling seamless growth as the business expands.

4. **Availability and reliability:** Managed cloud services offer high availability and reliability, ensuring that the system is always up and running. This is essential for a healthcare platform like BeHealthy, which needs to provide uninterrupted services to its users, especially considering the critical nature of healthcare data.

5. **Security:** Managed cloud services like AWS SageMaker offer robust security features, including encryption, access control, and compliance with industry standards. This ensures that user data is secure and that BeHealthy meets its regulatory requirements, which is particularly crucial in the healthcare industry where data privacy and security are paramount.

Moreover, AWS SageMaker provides a comprehensive set of tools and services tailored to machine learning workflows, further enhancing the development and deployment of machine learning models within the BeHealthy system:

- **Jupyter Notebook:** SageMaker offers Jupyter Notebook, which is essential for exploratory data analysis and rapid prototyping of machine learning models, enabling data scientists to collaborate efficiently and iterate on model development.

- **Data Wrangler:** This tool simplifies data preparation tasks such as cleaning, transforming, and aggregating data, accelerating the machine learning pipeline and improving data quality.

- **Amazon SageMaker Studio:** As an integrated development environment (IDE), SageMaker Studio provides a web-based interface for building, training, and deploying machine learning models. It offers a range of tools and services, including Jupyter Notebook, AWS Glue DataBrew, and built-in machine learning algorithms, facilitating end-to-end model development workflows.

- **Built-in and custom algorithms:** SageMaker offers a variety of built-in machine learning algorithms for common tasks like classification, regression, and clustering. Additionally, it supports the development and deployment of custom machine learning algorithms using frameworks such as TensorFlow and PyTorch, providing flexibility and customization options for model development.

- **Automatic Model Tuning:** This feature automates the process of hyperparameter tuning, optimizing model performance by running multiple training jobs in parallel and selecting the best-performing model based on user-specified objective metrics.

- **Model hosting and deployment:** SageMaker simplifies the deployment of trained machine learning models as web services, allowing other applications to access them through a REST API, ensuring seamless integration and scalability of machine learning capabilities within the BeHealthy system.

Overall, leveraging managed cloud services like AWS SageMaker aligns with BeHealthy's requirements for ease of deployment, cost-effectiveness, scalability, availability, reliability, security, and tailored support for machine learning workflows, making it a suitable choice for building the system architecture.

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 system architecture for this case study is structured into multiple layers to accommodate the various stages of the machine learning system lifecycle, with a focus on maintaining high Key Performance Indicators (KPIs) throughout the entire lifecycle while leveraging the benefits of MLOps. The architecture is divided into two environments to separate development and client-facing systems:

1. **Development Environment:**

   - The Development environment is designed to facilitate the evaluation of various models and determine the optimal solution for the given problem statement and dataset.

   - It provides an ideal platform for swift experimentation with both the data and models at hand.

   - This environment is where data scientists and developers can iteratively explore different algorithms, feature engineering techniques, and model architectures to identify the best-performing models.

2. **Production Environment:**

   - The Production environment is where the best model identified during development is deployed after thorough testing.

   - It contains pipelines for training, testing, monitoring, and inference, enabling the system to operate efficiently in a production setting.

   - This environment ensures that the deployed model meets performance, scalability, and reliability requirements, and it includes mechanisms for ongoing monitoring and optimization.

   - Additionally, it may incorporate features such as A/B testing, canary deployments, and automated scaling to maintain high KPIs and respond effectively to changes in workload or data distribution.

By separating development and production environments, the architecture promotes a structured approach to model development and deployment, reducing the risk of introducing errors or performance issues into the client-facing system. It also allows for continuous improvement and refinement of models in the development environment without disrupting production services. Overall, this architecture supports the goals of maintaining high KPIs throughout the machine learning system lifecycle and leveraging MLOps practices for efficient and reliable operation.

1. **Data and Model Experimentation:**

This phase is conducted entirely within the Development environment as intended. Here, data scientists collaborate and perform experiments and trials to identify the best model with the most suitable features, which can also be utilized in the future in case of any drift.

In this structure, feature engineering is executed within Jupyter Notebooks available in SageMaker Studio. These notebooks contain templates that expedite the process. Here, data scientists conduct experimentation and prepare final data, which is then utilized by SageMaker Autopilot for further experimentation across optimized models and algorithms to select the best-performing model based on a metric. All these processes are tracked by SageMaker experiments and trials, where artifacts and results are stored. Once approved, the results can be pushed to an endpoint from the SageMaker model registry for testing purposes using SageMaker endpoint. Once the desired result is obtained as per the threshold, the model is moved to production.

2. **Capabilities of SageMaker Pipelines:**

- **Build ML Workflows:** Using the Python SDK, ML workflows comprising parameters, different steps, and data dependencies can be built. SageMaker Pipelines allow orchestration of SageMaker jobs such as processing jobs and training jobs, and triggering the execution of these pipelines.

- **Troubleshoot ML Workflows:** SageMaker Studio enables the visualization of the execution of the pipeline and the status of each step in real-time. Additional information about each step can also be viewed in SageMaker Studio.

- **Manage Models:** Different versions of models can be managed using the Model Registry. Capability is provided to approve/reject models in the model registry. The model registry consists of different model packages, and each model package consists of multiple versions of the model.

- **Scaling MLOps:** By creating a project in SageMaker Studio, users get a code repository, seed code, and the MLOps infrastructure set up for them. MLOps templates published by SageMaker are provided for building, deploying, and establishing end-to-end workflows.

- **Track Lineage:** SageMaker pipelines offer built-in lineage tracking for tracking data, models, and artifacts, with support for tracking custom entities.

3. **Automated Data Pipeline:**

This component focuses on automating model training by converting code developed in notebooks to Python scripts. With automation coupled with a feature store, data and training pipelines can run whenever there is any change in the live data, enabling continuous delivery of existing deployed models after re-training on the newly transformed data stored in the feature store. The tools used for automation in this stage are SageMaker Studio and SageMaker Pipelines.

4. **Automation of Training Pipeline:**

In this step, various methods used in data preparation, feature extraction, and model validation are tested to effectively track whether all the components are working in the desired manner. The tests applied here include unit tests, integration tests, and user acceptance testing (UAT). If the model passes all these tests, it can be moved to production, i.e., it can be used for making inferences/predictions. Testing helps in the continuous integration of models trained on new data.

5. **Inference Pipeline:**

In this stage, once the model/code passes all the tests, it is deployed for serving predictions using SageMaker Endpoint for deployment and providing end-service.

6. **Continuous Monitoring Pipeline:**

Continuous monitoring of the deployed model is essential for tracking the model performance and ensuring that the model doesn't go stale. It signals what action needs to be performed based on any changes in the live data. The tool used in this stage for monitoring any data drifts is called SageMaker Model Monitor and AWS CloudWatch. The Amazon SageMaker Model Monitor continuously monitors the quality of Amazon SageMaker ML models in production. The Model Monitor provides the following types of monitoring:

- **Monitor data quality**: Monitor drift in data quality.

- **Monitor model quality:** Monitor drift in model quality metric, such as accuracy.

- **Monitor bias drift for models in production:** Monitor bias in your model's predictions. Alerts can be set using AWS CloudWatch in case of deviations in the model quality.

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

Drift Detection and Response:

- Component/ Pipeline Triggered for Drift Detection:
  - If there is any drift detected, the model retraining pipeline will be triggered.

- Response to Drift Beyond Acceptable Threshold:
  - If the drift falls within the warning and critical thresholds, indicating a discrepancy between the inference data and the training data model, the model should undergo retraining using new data to optimize its parameters.

- However, if the data drift exceeds the critical threshold, model retraining becomes ineffective. In such cases, it is advisable to revisit the development environment, conduct experimentation to identify the optimal ML model, subject it to User Acceptance Testing (UAT), and finally deploy it for inference purposes.

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

Triggering Pipeline for Additional Annotated Data:

- Component/ Pipeline Triggered:

- To ensure that the new patterns in the additional annotated data are captured, the entire pipeline starting from the data ingestion stage must be triggered.

- Process Description:

- This involves ingesting the new data into the pipeline, followed by training, validating, testing, and deploying the model again.

- By executing this process, the model will be updated to incorporate the new patterns in the additional data, resulting in more accurate outcomes.

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

Ensuring Correct Data Format for Inference Pipeline:

To ensure that the new data you are receiving is in the correct format that the inference pipeline takes, the following steps can be taken:

1. Data Pipeline Preparation:

- The new or test data needs to undergo the data pipeline, which includes preprocessing and formatting the data ***to match the requirements of the model.***

2. Triggering the Pipeline:

- Typically, the data pipeline is triggered on the test dataset containing the new data.

3. Data Processing:

- Once the data has been processed through the pipeline, it will be formatted appropriately for the inference pipeline.

4. Inference Pipeline Usage:

- The processed data can then be fed into the inference pipeline to generate predictions on the processed test dataset or dataframe.

By following these steps, you can ensure that the new data is prepared in the correct format for accurate predictions by the ***model within the inference pipeline.***