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

**The overall requirements,** can be broken down into three different categories:

|  |  |  |
| --- | --- | --- |
| Must have | Good to have | Out of scope |
| An ML based solution with atleast **50% accuracy** that analyses the clinical notes, extract disease and corresponding treatments and create a disease-treatment mapping. | A ML based solution that can confidently provide correct disease-treatment mapping with minimum **75% accuracy.** | A ML based solution with 100% accuracy that eliminates the need of data validation team altogether. |

**KPIs**

|  |  |  |
| --- | --- | --- |
| KPI | Current position | Expected position |
| Need of trained doctor to manually review the disease-treatment mappings | 1000 doctors | 500 doctors (initial) |
| Scaling up of data entry team | Yes | Scaling down / status-quo |
| Man-hour effort need per day for data entry team(one person) | 8 hr/day | 1 hr/day |
| Chances of human-prone errors | High | Very low |
| Reduce Costs | High | Decrease by atleast 50% |

Success Criteria: System that can

* Extract diseases and corresponding treatments to create create a disease-treatment mapping from the clinical notes
* Consistently deliver atleast 50% accuracy
* Automate the data processing, model building and evaluation, testing and inference processes
* Monitor drifts in data or concepts and learn continuously when required
* Automatically trigger data, training, testing pipelines as required

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

* Reduction in human errors
* Reduction in man-hours spent on building and training the system
* Reduce the cost required to build and maintain such a model
* Building a model takes a lot of effort. An MLOps system will bridge the gap and also

reduce lag in model development and deployment

* The system will have tools that help teams to collaborate during the model building

stage.

* There will be a good view on what experiments were conducted and their metrics so the best model can be identified.
* Experiment tracking in MLOps system will track all the modelling exercises that were conducted. It will give a good view what worked and what did not work and pick the best model.
* Results of experiments can also be shared between various teams for evaluation and

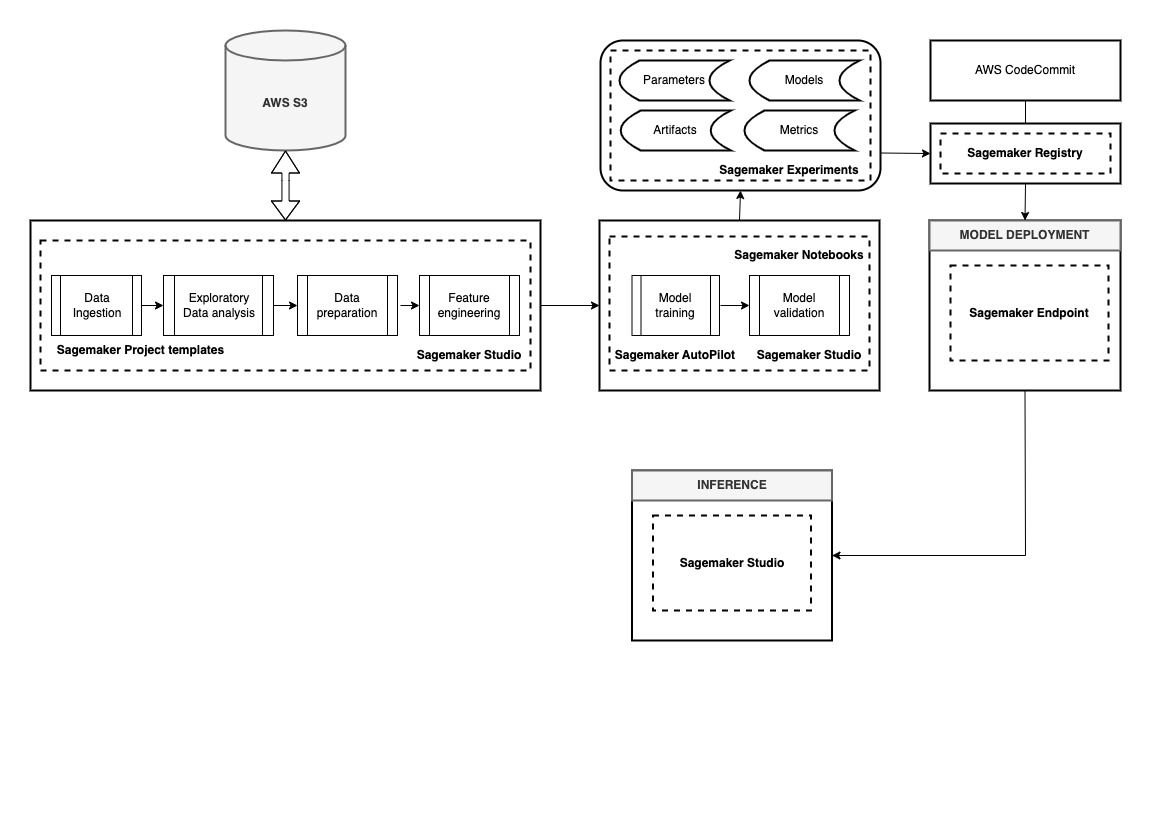
validation.

* Symmetry can me maintained between development and production environments to ensure that the results got during development are the ones that are got in production.
* ML models tend to become stale over time. MLOps systems have monitoring systems to detect data drift or model bias automatically. This will ensure that prediction accuracy is maintained.
* The pipeline can be triggered, and models can be continuously trained as and when a drift is detected, or new data arrives. It will help avoid model decay.
* API endpoints can be created for real time inference
* The whole pipeline can be automated with minimal manual intervention

This system would eliminate the need to have a trained doctor analysing each and every clinical note to identify the mapping as well as reduce the need of a data entry team. This automation would result in reduced human errors and reduced man-hours effort

* Increase productivity of the team
* Ensure high quality model production

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.

Fig : Development Environment in MLOps

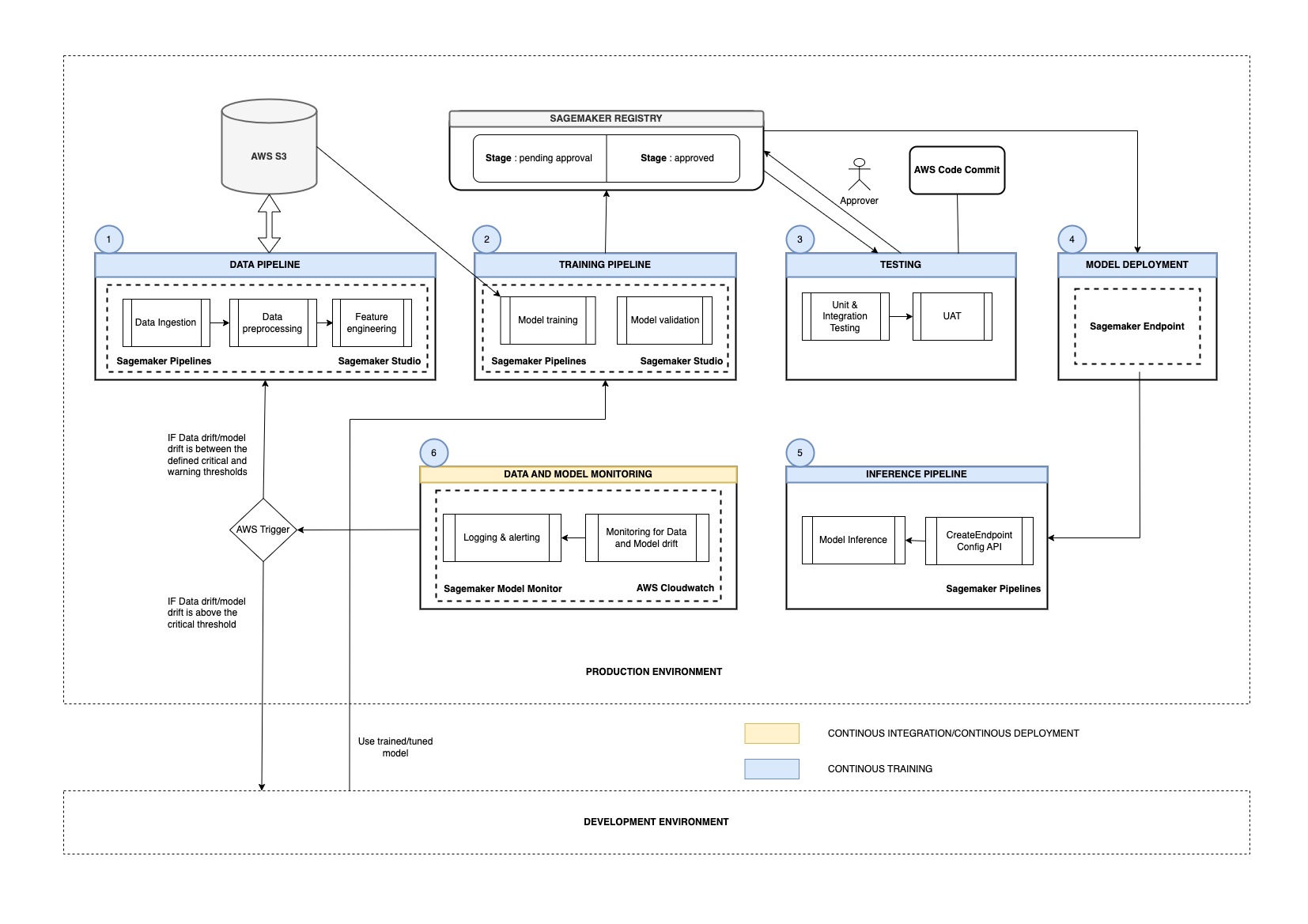


Fig : Production Environment in MLOps

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

As BeHealthy is a late stage startup, we have opted for leveraging managed(Cloud) services for MLOps because of below benefits/advantages :

|  |  |
| --- | --- |
| **Service** | **Benefit using AWS Sagemaker** |
| End2End MLOps | Integrated |
| Time to set up | Less |
| Maintenance of infrastructure | Low |
| Ease of deployment | Low |
| Learning curve | Low |
| Pre-configured MLOps template | Available |

Looking at the current stage of the startup, we want to leverage vendor solutions to get started as quickly as possible so that we can focus your limited resources on the core offerings of your product. As the use cases grow, however, vendor costs might become exorbitant, and it might be cheaper for them to invest in their own solution.

The data does not seem to have any sensitive user information so it can be safely sent to managed services on the cloud

Also cloud services offer the scalability that will be required as the business grows.

**Benefits/Characteristics of Sagemaker :**

* Amazon SageMaker is a fully managed machine learning service.
* With SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment.
* It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so we do not have to manage servers.
* It also provides common machine learning algorithms that are optimised to run efficiently against extremely large data in a distributed environment. With native support for bring-your-own-algorithms and frameworks, SageMaker offers flexible distributed training options that adjust to your specific workflows. Deploy a model into a secure and scalable environment by launching it with a few clicks from SageMaker Studio or the SageMaker console.
* Training and hosting are billed by minutes of usage, with no minimum fees and no upfront commitments.

**Tools within AWS SageMaker:**

* Amazon SageMaker Studio - First fully integrated development environment (IDE) for machine learning.
* Amazon SageMaker Notebooks - Enhanced notebook experience with quick-start and easy collaboration
* Amazon SageMaker Experiments - Experiment management system to organise, track and compare thousands of experiments.
* Amazon SageMaker Debugger - Automatic debugging analysis and alerting
* Amazon SageMaker Monitor - Model monitoring to detect deviation in quality and take corrective actions.
* Amazon SageMaker Autopilot - Automatic generation of machine learning models with full visibility and control.

**Q5. Workflow of the solution:**

**We 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.**

The system architecture comprises multiple layers for the different stages of the ML life cycle. The ML system is designed considering all the components, tools, methods and procedures to help deliver the overall requirements drafted in the PRD.

the system architecture is divided into two environments:

1.**Development environment:** This environment is structured to help we understand which model will perform the best for the given problem statement and data. This is an environment where we can perform rapid experimentation around data and models.

2.**Production environment:** This environment is where we operationalise the best model identified from the development environment/stage.

Model development in the development environment is an interactive process and broadly consists of the following steps:

1**.Exploratory data analysis:** In this step, we understand the patterns and infer insights from the given data using data visualizations. Some of the tools used in this stage will be Matplotlib, Seaborn etc. We can leverage the differen available Sagemaker Project templates to start with our MLOps journey.

**2.Data preparation and feature engineering:** Once we understand the data and its distribution, we move to the other operations to perform cleaning, pre-processing and feature engineering on the given data. This step is crucial for performing experimentation using the features extracted from the data.

3**.Model training and tuning:** We start building the model using the features extracted from the previous step. Rapid experimentation is essential to selecting the baseline model in this stage. Here, we experiment with multiple models/algorithms; however, it is essential to strategize which models we want to select.

To accelerate the development, we will be using tools like Sagemaker AutoPilot and Sagemaker Experiments and Trials for rapid prototyping and experiment tracking.

• **Sagemaker Autopilot** allows automatic generation of ML models with full visibility and control. Once we provide the independent and target variables along with dataset, Autopiot will automatically explore different slutions to find the best model that can be used.

• **Sagemaker Experiments and Trials is** an Experiments management system to organise, track and compare multiple experiments.

**4.Model validation:** In this step, based on the performance of all the candidates (models used for experimentation), we select the best one that meets the evaluation criteria If it doesn’t meet the standards, we must go back and experiment until we reach acceptable numbers.

**5.Model serving/endpoint:** Once we have finalized the best-performing model, we create an endpoint using the endpoint configuration that is created using the **CreateEndpointConfig API. SageMaker Endpoint** is used to provide resources and deploy models. The endpoint name must be unique within an AWS region in your AWS account. This Endpoint in Dev Environment will act as a simulator of Production-environment.

Broadly, in the production environment, we will work with multiple pipelines. Here, the concept of the pipeline is introduced to split up your machine learning workflows into independent, reusable, modular parts that can then be integrated together to serve any task like feature creation, model training, prediction serving, etc. An ML pipeline is made up of multiple components which need to be reusable, composable, and potentially shareable across the entire ML pipeline.

**Capabilities of Sagemaker Pipelines :**

• **Build ML workflows:** Using Python SDK, we can build ML workflows comprising parameters, different steps and data dependencies. We can also orchestrate SageMaker jobs such as the processing job and the training job and can also trigger the execution of these pipelines.

• **Troubleshoot ML workflows:** We can visualise the execution of the pipeline and the status of each step in the pipeline in real time in SageMaker Studio. We can also view additional information about each of the steps in SageMaker Studio.

• Manage models: We can manage different versions of models using the Model Registry. We also have the capability 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**: We can create a project in SageMaker Studio and get a code repository, seed code and the MLOps infrastructure set up for we. We are provided with MLOps templates published by SageMaker for building, deploying and establishing end-to-end workflows.

• **Track lineage:** With in-built lineage tracking for SageMaker pipelines, we can track data, models and artefacts. Also, support is provided for tracking custom entities.

**Automated Data and training pipeline:** This component focuses on automating model training by converting the code developed in notebooks to Python scripts. With automation coupled with using a feature store, the data and training pipelines can run whenever there is any change in the live data. This helps in the continuous delivery of existing deployed models after it is re-trained on the newly transformed data stored in the feature store. The tool used for automation in this stage is Sagemaker Studio and Sagemaker Pipelines.

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

**Inference pipeline:** In this stage, once the model/code passes all the tests, we will go ahead and deploy the model for serving predictions. The tool used in this stage is Sagemaker Endpoint for deployment and providing end-service.

**Data and model monitoring:** Keeping a continuous check on 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 ‘trigger’ connected to this component decides what action to take: model experimentation or model retraining. 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

We can set alerts using AWS CloudWatch when in the case of deviations in the model quality.

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

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

If the drift is between the defined warning threshold and the critical threshold, then it implies the inference data is not similar to the data model was trained on. This would not give accurate results. Hence, in this case, model should be retrained with the new data, so that we can get optimized model and its parameters.

If the data drift is above the defined critical threshold, then model retraining will not help and in this case, we need to go back to the development environment, perform the process of experimentation again to find the best ML model, which needs to again undergo UAT and then should be good for inference purpose.

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

The complete pipeline starting with the data pipeline should be triggered so that the additional data is ingested and the model is trained, validated, tested and then deployed again. This will make sure that the new patterns in the additional data is captured.

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

The new data/test data will have to undergo the data pipeline, which ensures the data is preprocessed and is available in the format that is expected by the model. Hence,most of the times, data pipeline is triggered on the test dataset and inference pipeline will help to get predictions on the processed test dataset/dataframe.