BEHealthy healthtech company Case Study

Business goal:

Currently, if you need to extract diseases and treatments from free text, a trained person with clinical knowledge must manually look at the clinical notes and then pull this information.
A data entry team would manually look at the clinical text and extract information about diseases and their treatment data. A data validation team would validate the extracted information. This process is prone to errors, and as the data increases, the data-entry team's size would need to be scaled up.

Automating this process would result in reduced man-hours. The data-entry team would not be required. The data validation team would still need to perform validation on extracted data. It would also significantly reduce manual errors.

1. **System design:** *Based on the above information, describe the KPI that the business should track.*
    - While extracting diseases and treatments from free text the most important KPIs would be using MLOps are:
        a. Must Have: There should be a reduction of men hours.
        b. Must Have: We should be able to see significantly reduction in manual errors
        c. Good to Have: We should able to get structured information for analytics purposes
        d. Good to have: The model should be able to adapt to new input methods

2. **System Design:** *Your company has decided to build an MLOps system. What advantages would you get by opting to build an MLOps system?*
    - MLOps is a method based on adapting DevOps practices to machine learning development processes. MLOps makes data science more productive by reduce defects, improve delivery time. Below are Few key points on how MLOps can benefit us:

Productivity

- MLOps increases the productivity of all processes within the ML lifecycle by:
- **Creating automated pipelines**
    - There are many labor-intensive and repetitive tasks within the ML lifecycle. For instance, data scientists spent nearly half of their time getting the data ready for the model.
    - Manual data collection and preparation are inefficient and can lead to suboptimal results.
    - MLOps stands for automating the entire workflow of the ML model. This covers all the actions from data collection to model development, testing, retraining, and deployment.
    - MLOps practices save time for teams and prevent human-induced errors. In this way, teams can engage in more value-added efforts rather than repetitive tasks.
- **Standardizing ML workflows for efficient collaboration**
    - Company-wide adoption of ML models requires collaboration between not just data scientists and engineers but also IT and business professionals.
    - MLOps practices enable businesses to standardize the ML workflows and create a common language for all stakeholders.
    - This minimizes incompatibility issues and speeds up the entire process from creation to deployment of models.

Reproducibility

- Automating ML workflows provides reproducibility and repeatability in many aspects, including how ML models are trained, evaluated, and deployed. This makes continuously trained models dynamic and integrated into change:
- **Data versioning**: MLOps ensures storing different versions of data that were created or changed at specific points in time and saving snapshots of different versions of data sets.
- **Model versioning**: MLOps practices involve creating feature stores for different types of model features and versioning the model with different hyperparameters and model types.

Reliability

- By incorporating CI/CD principles from DevOps into the machine learning processes, MLOps makes ML pipelines more reliable. Automated ML lifecycle minimizes human errors and companies gain realistic data and insights.
- One of the biggest challenges of ML development is scaling from a small model to a large production system. MLOps streamlines model management processes to enable reliable scaling.

Monitorability

- Monitoring the behavior and performance of ML models is essential because models drift over time as the environment changes. MLOps enable businesses to monitor and get insights about model performance systematically by:
- **Retraining the model continuously**: ML models are monitored and automatically retrained periodically or after a certain event. The purpose of retraining a model is to ensure that it consistently provides the most accurate output.
- **Sending automated alerts to staff in case of model drift**: MLOps gives the business real-time status of your data and model and alerts the relevant employees if the model performance degrades below a certain threshold. Thus, it enables you to take quick actions against model degradation.

Cost Reduction

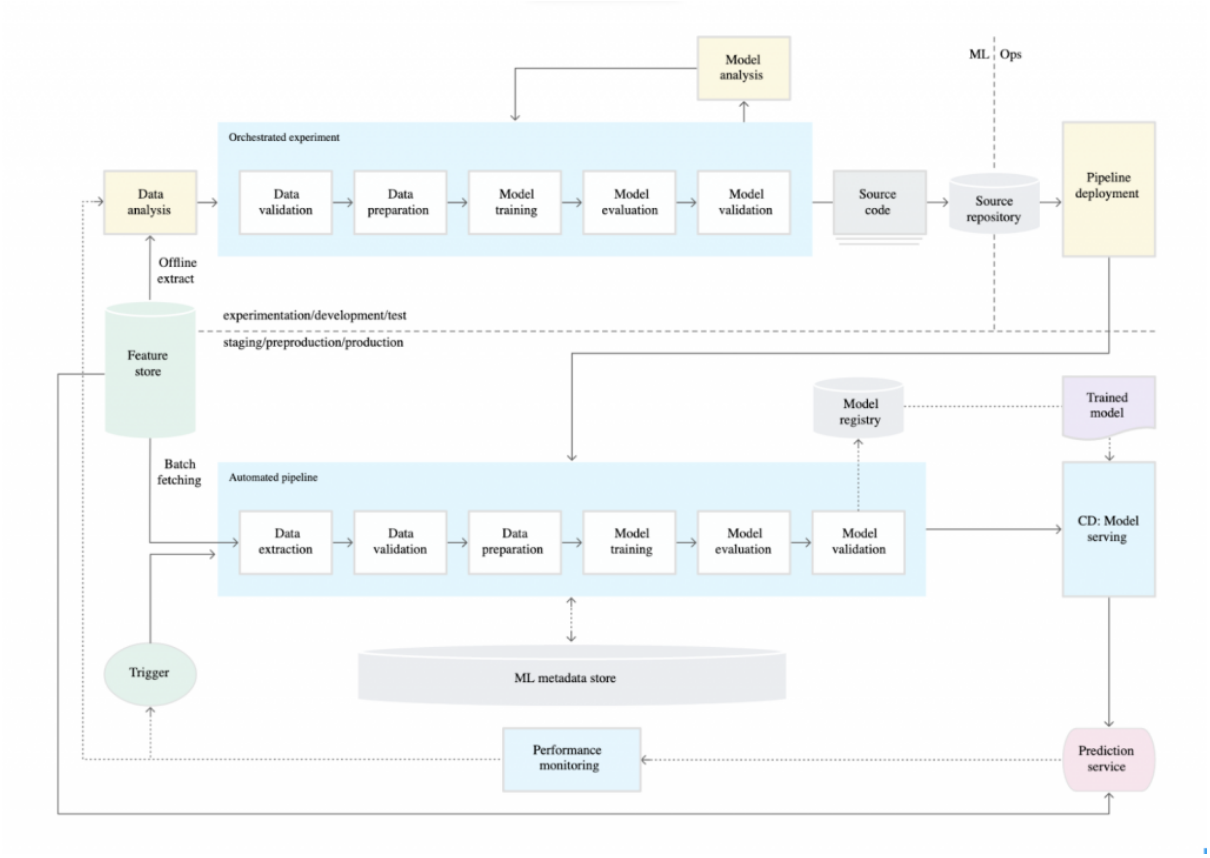
- MLOps can significantly reduce costs over the entire machine learning lifecycle:
- Automation minimizes the manual efforts to manage machine learning models. This will free up employee time which can be used for more productive tasks.
- It enables you to detect and reduce errors more systematically. Decreased errors during model management will also translate to reduced costs.

3. **System design:** *You must create an ML system that has the features of a complete production stack, from experiment tracking to automated*

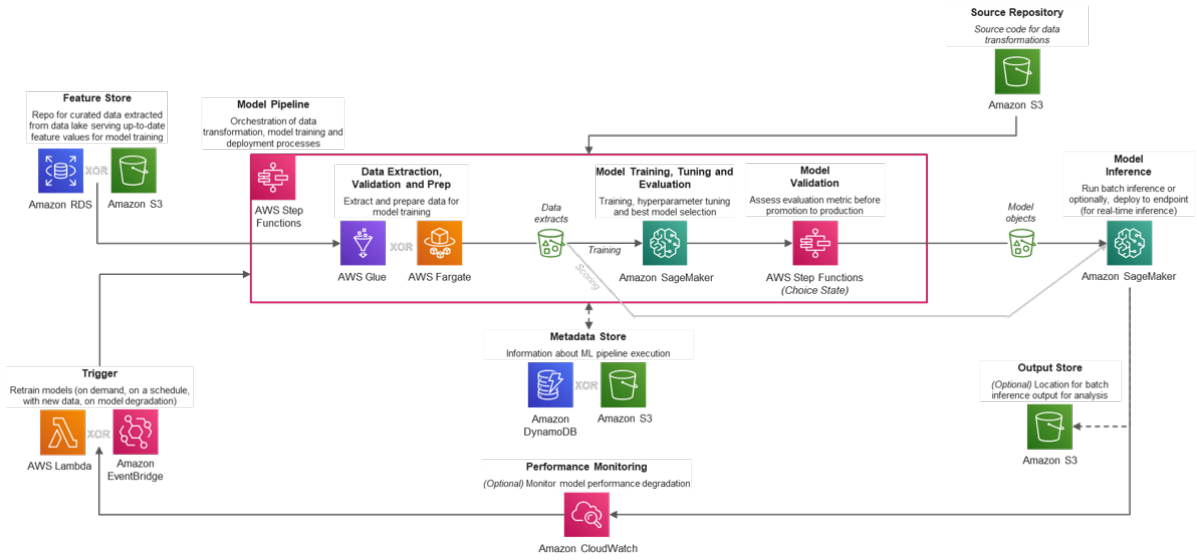
model deployment and monitoring. For the given problem, create an ML system design (diagram).

The MLOps tools that you want to use are up to your judgement. You can use open-source tools, managed service tools or a hybrid.
You can use draw.io or any other convenient tool to create the architecture. You can refer to the module on "Designing Machine Learning Systems" for understanding how to create an MLOps Architecture.
You can upload the design/diagram as an image in the PDF.

Standard Architecture:



Architecture Including Tools:



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

- The diagram shown above is an example of what an automated MLOps pipeline could look like with MLOps tools in AWS. It uses a serverless architecture which has benefits in terms of cost-efficiency and ease of development. The key with MLOps tools in architecture are:

- **AWS Step Functions** for orchestrating the various jobs within the pipeline and incorporating logic for model validation
- **Amazon S3** for initial data storage/data lakes, storing flat-file data extracts, source code, model objects, inference output, and metadata storage (the initial feature store might also be a relational database e.g. Amazon RDS and alternatively, Amazon DynamoDB could be used to store metadata for some use-cases)
- **Amazon SageMaker** for model training, hyperparameter tuning and model inference (batch or real-time endpoint)
- **AWS Glue or ECS/Fargate** for extracting, validating and preparing data for training jobs

- **AWS Lambda** for executing functions and acting as a trigger for retraining models
- **Amazon CloudWatch** for monitoring SageMaker tuning and training jobs
- **Amazon Athena** to the inference output (for connecting Tableau for data visualisation)

5. 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.

a. *Data and model experimentation*
b. *Automation of data pipeline*
c. *Automation of the training pipeline*
d. *Automation of inference pipeline*
e. *Continuous monitoring pipeline*

Data and model experimentation –

- First step of any machine learning system is Data Preparation and create baseline solution. We start with data collection in the form of csv files then use Data analysis libraries like
- We can use **AWS Glue or ECS/Fargate** to read the data from the csv files and then clean the dataset to drop duplicates and null values. Internally we are doing data profiling here for Exploratory data analysis for understanding the data by creating and visualization. Then we are also using some feature engineering like one-hot encoding or label encoding, count vectorizer to convert the categorical features into numerical data as machine learning models can't understand categorical data. Here We are using AWS Glue libraries where we can run the dataset on multiple machine learning algorithms at once and find the best performing model out of it. We can use **Amazon**

SageMaker for hyperparameter tuning and apply on the best performing baseline model to improve the performance of the model such as tuning for accuracy, recall, F1-score, or AUC. Once the tuning is completed, we save the hyperparameters and model for further use down the line.

Automation of data pipeline

- The different blocks in the data pipeline constitutes data collection, data cleaning, feature extraction is automated using **Amazon S3**. Each block in the pipeline is created as python scripts that are sequentially connected with **Amazon S3.** A block in a pipeline is a collection of functions that does specific tasks such as loading the data, cleaning the data, converting categorical features using one-hot encoding, vectorization and storing features to feature store. We can use Amazon event bridge, **AWS Lambda** to trigger and monitor the pipeline.

Automation of Model pipeline

This pipeline is continuation of data pipeline where we load the data from future store and then split the data into train, validation, and test sets. We use the fine-tuned hyperparameters and train the model. We use **Amazon SageMaker** experiment tracking endpoints to store the artifacts, hyperparameters and metrics. Once the training is completed, we send the model to staging and perform model versioning. The model that performs better than existing model are promoted to production.

Automation of Inference Pipeline

- Once the model training is completed the next step is to create API using **Amazon SageMaker** to server the model for predictions. In this pipeline we load the model from model registry with production stage and use batch or streaming data for predictions. We deploy the service dags to airflow for monitoring the Inference pipeline.

Continuous monitoring pipeline

- In this pipeline we continuously monitor the performance of the model on real world data as the ml models are prone to degrade over time. In this pipeline we monitor data drift and model drift by calculating the distribution of data and other metrics of model prediction. We can use **Amazon CloudWatch** to monitor the Computing resources and use alerting system whenever there is a drift in the data. Based on some conditions or thresholds retraining of the pipeline will be triggered

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?*
    - If the drift is not significant then No action is needed.
    - The data and model training pipelines are triggered for retraining the model with the new data if the drift is moderate.
    - We must go back to the development environment to perform data and model experimentation if the drift detected is beyond an acceptable threshold

- *What component/pipeline will be triggered if you have additional annotated data?*
    - The continuous learning/training pipeline can re-train our model whenever there is an additional in the data. Therefore, this helps create a robust system that can account for dynamic user behaviors.
    - But if our retrained model is not good enough; Well, in that case we can go back to the development environment and perform rapid experimentation to come up with the best model.
- *How will you ensure the new data you are getting is in the correct format that the inference pipeline takes?*
    - We can apply data preparation using operations like cleaning, pre-processing and feature engineering on the

new data. when the new data comes in we generally compare the data using schema validation and do the data sanity and validation using the cleaning, pre-processing and feature engineering steps .