MLOps - NLP - Case Study: Sreedhar K

System Design

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

Ans 1) BeHealthy aims to convert the unstructured clinical notes into structured data for analytical purposes. The key performance indicator (KPI) that BeHealthy should monitor is the accuracy of mapping diseases to prescribed treatments, with both precision and recall being crucial factors. To quantify this, the F1 metric is utilized to assess the machine learning model's ability to detect disease (D) and treatment (T) tags within a sentence. It is important to focus on the model's failures in order to identify areas where it may be inaccurately predicting the treatment for a specific disease. By doing so, efforts can be made to improve the recall score. Additionally, there should be a balanced representation of all diseases and treatments to ensure a comprehensive evaluation. This approach provides a healthy way to quantify the KPI. BeHealthy has developed a Conditional Random Field (CRF) model to identify named entities in the medical dataset. Based on these considerations, the following KPIs are recommended for this business.

1. **F1-score:** The F1 score is a suitable KPI as it combines precision and recall, making it applicable even in cases where datasets are imbalanced. This metric ensures that both precision and recall need to have reasonable values in order for the F1 score to be high. In the given scenario, where neither precision nor recall alone would be sufficient, the F1 score serves as an appropriate KPI.

2. **Training Speed and Performance**: Training machine learning models with large datasets can be time-consuming, often taking several hours. Therefore, it is crucial to optimize the training process to minimize the time required.

3. **Data Capacity:** Data Capacity is a significant KPI that indicates the number of samples a machine learning platform can effectively process based on the given number of variables. It is crucial to consider Data Capacity as an important metric, as exceeding the system's data processing capabilities can lead to crashes or performance issues in many ML platforms. Understanding and working within the Data Capacity of the system ensures smooth and efficient data processing without overwhelming the platform.

4. **Inference Speed**: Inference speed is a highly significant KPI as it directly influences the response time of a model, making it a crucial factor in real-time applications. Inference speed refers to the time it takes for a model to process and generate predictions or outputs based on new input data. In real-time scenarios where timely decision-making is essential, a fast inference speed is vital to ensure quick and efficient processing. By optimizing the inference speed, the model can deliver near-instantaneous results, enhancing its practical usability and effectiveness in real-time applications.

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

Ans 2) Manual Pre Processing , Exploratory Data Analytics , Training , Measuring the model performance , Tracking the drift, Deploying the models could be done once but once it becomes a repetitive task ,it would not only involve lot of time , but cost as well since we require huge resources , and this would affect the business continuity in future. Since performing changes manually would also be a time-consuming process and continuous integration won't be possible . There is additionally a huge chance of Manual error which cannot be avoided, and might become a bottleneck to the team .

Implementation of an end-to-end MLOps system not only addresses the challenges of scalability, efficiency, and manual errors but also promotes collaboration, enables continuous development, and provides robust monitoring mechanisms for maintaining model performance over time.

Advantages of implementing MLOps Systems :

1. **Productivity**: Providing self-service environments with curated data sets empowers data professionals to work efficiently, optimize their productivity, and focus on the core tasks of analysis and model development.

2. **Repeatability**: Automation of all the steps in the pipeline ensures a repeatable process, including how the model is trained, evaluated, versioned, and deployed.

3. **Reliability**: Incorporation of CI/CD practices allows for the ability to not only deploy quickly , but also with increased consistency and quality.

4. **Auditability**: Versioning of all inputs and outputs, from data science experiments to source data to trained model, implies that we can demonstrate exactly how the model was built and where it was deployed.

5. **Data and model quality:** MLOps provides the necessary framework and tools to enforce policies that guard against model bias and enable continuous monitoring of data statistical properties and model quality. By incorporating these practices, organizations can build responsible and reliable machine learning systems that address biases, adapt to changing data dynamics, and maintain high-performance standards.

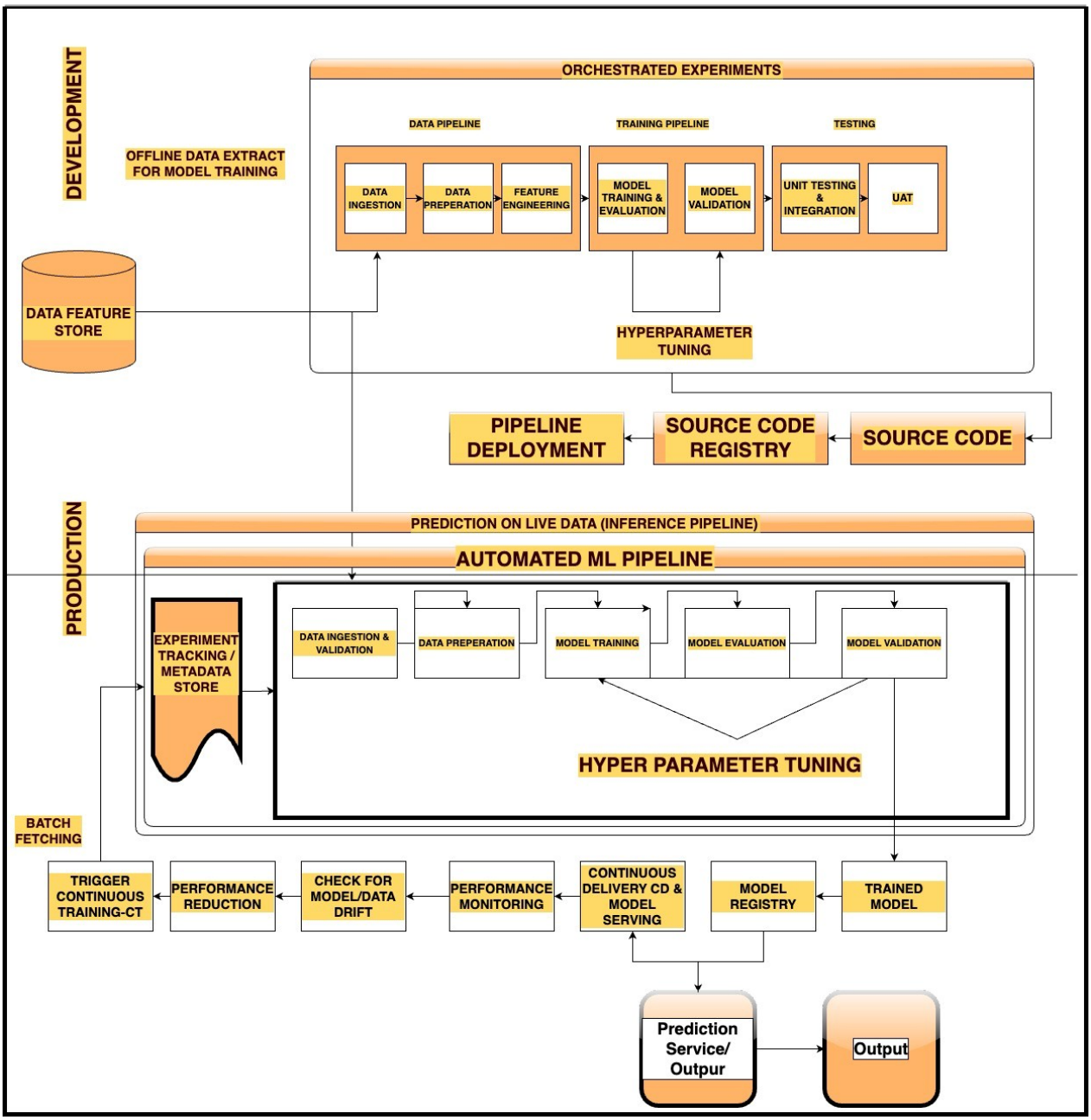
Q3. System design: You must create an ML system that has the features of a complete production stack, from experiment tracking to automated model deployment and monitoring. For the given problem, create an ML system design (diagram).

The MLOps tools that you want to use are up to your judgement. You can use opensource 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.

Ans 3) The system design involves development stage, Production stage and CI/CD stages.



Development stage includes:

1. Data Ingestion
2. Exploratory Data Analysis
3. Data Cleaning and Validation

4. Data Preparation and Feature Engineering
5. Model Training
6. Model Evaluation
7. Model Validation
8. Hyper-parameter Tuning
9. Source Code Repository

After doing Model experimentation, and once we got the right model, we will move to the automated production pipeline.

Data Pipeline

1. Data Validation
2. Data Preparation

Training Pipeline

1. Model Training
2. Model Evaluation
3. Model Validation
4. Hyper-parameter Tuning

Inference Pipeline

1. Predictive Service (Prediction) using API
2. Experiment tracking/ Metadata Storage

The Continuous Integration and Continuous Deployment includes

1. Model Registry
2. Continuous Delivery – Model Serving
3. Performance Monitoring
4. Continuous Training

Tools used for various stages in development and production are as follows:

1. **Pandas Profiling** – EDA
2. **PyCaret** – Model training, Model Evaluation, Model Validation and Hyperparameter Tuning.
3. **Git Repository** – Source Code Repository.
4. **Airflow** – Data Pipeline, Training Pipeline, Inference Pipeline.
5. **Pytest** – Unit Testing and Integration testing.
6. **MLFlow** – Experiment tracking and Model Registry and Staging.
7. **Evidently** – Data and Model Monitoring.
8. **MLFlow Serving** – Predictive Service and Model Inference.

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

Ans4) We have considered open source tools for the BeHealthy Use Case considering below factors :

1. Open source is generally much more cost-effective than a proprietary solution. Not only are open source solutions typically much more inexpensive in an enterprise environment for equivalent or superior capability, but they also give enterprises the ability to start small and scale. The cost-effectiveness of open source solutions, along with their scalability, grants enterprises the ability to leverage powerful software capabilities while optimizing their investments and avoiding unnecessary expenses.

2. BeHealthy is a late stage Start-Up . Hence, we can expect the budget constraints. So, open source becomes a better choice instead of Cloud based packaged tools which are expensive.

3. Open source empowers BeHealthy to start small and explore community versions, allowing them to assess software options and gradually scale up. The flexibility to transition to a commercially-supported solution when needed ensures that BeHealthy can align their software choices with their business requirements and growth trajectory.

Below are the tools which are used in the above use case:

1. **MLflow** is a platform to streamline machine learning development, including tracking of experiments, packaging of code into reproducible runs, and sharing and deploying models. MLflow provides a set of lightweight APIs which can be used with any existing machine learning application or library (TensorFlow, PyTorch, XGBoost, etc), wherever you currently run ML code (e.g. in notebooks, standalone applications or the cloud). MLflow's current components are:

   - MLflow Tracking: An API to log parameters, code, and results in machine learning experiments and compare them using an interactive UI.

   - MLflow Projects: A code packaging format for reproducible runs using Conda and Docker, so you can share your ML code with others.

   - MLflow Models: A model packaging format and tools that let you easily deploy the same model (from any ML library) to batch and real-time scoring on platforms such as Docker, Apache Spark, Azure ML and AWS SageMaker.

   - MLflow Model Registry: A centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of MLflow Models.

2. **Apache Airflow** (or simply Airflow) is a platform to programmatically author, schedule, and monitor workflows.

When workflows are defined as code, they become more maintainable, versionable, testable, and collaborative.

Airflow is utilised to author workflows as directed acyclic graphs (DAGs) of tasks. The Airflow scheduler executes your tasks on an array of workers while following the specified dependencies. Rich command line utilities simplify the performance of complex surgeries on DAGs. The rich user interface makes it easy to visualize pipelines running in production, monitor progress, and troubleshoot issues when needed

3. **Evidently** is an open-source Python library for data scientists and ML engineers. It helps to evaluate, test, and monitor the performance of ML models from validation to production. Evidently has a modular approach with 3 interfaces on top of the shared metrics functionality.
   - Tests: batch model checks
   - Reports: interactive dashboards
   - Real-time ML monitoring

4. **Pandas profiling** is an open source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. Besides, if this is not enough to convince us to use this tool, it also generates interactive reports in web format that can be presented to any person, even if they don't know programming.

5. **PyCaret** is an open-source, low-code machine learning library in Python that automates machine learning workflows. It is an end-to-end machine learning and model management tool that speeds up the experiment cycle exponentially and makes you more productive. In comparison with the other open-source machine learning libraries, PyCaret is an alternate low-code library that can be used to replace hundreds of lines of code with few lines only. This makes experiments exponentially fast and efficient. PyCaret is basically a Python wrapper around several machine learning libraries and frameworks such as scikit-learn, XGBoost, LightGBM, CatBoost, spaCy, Optuna, Hyperopt, Ray, and few more.

6. **Pytest** is a Python testing framework that originated from the PyPy project. It can be used to write various types of software tests, including unit tests, integration tests, end-to-end tests, and functional tests. Its features include parametrized testing, fixtures, and assert re-writing.

.

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.

Ans5 : For the required use case BeHealthy would need a proper workflow defined based on which each of the components will function.

Automation of Data Pipeline:

The Data Pipeline will be the entry point of the MLOps design.

This involves 3 different tasks, which are executed sequentially. Data is cleaned and validated for duplicate data ,and for removal of unwanted data. Spelling and Vocabulary errors are corrected and outliers are managed.

Once the data is cleaned, data is prepared for processing, we can use these contexts as features and feed them to our model, then the model will be able to understand the sentence better. Some of the common features that we can extract from a sentence are the number of words, number of capital words, number of punctuation, number of unique words, number of stop-words, average sentence length, etc. We can define these features based on our data set we are using. The data pipeline is created in **Airflow**.

Data Cleaning >> Data Preparation >> Feature Engineering

Automation of Training Pipeline:

Once we have successfully executed the Data Pipeline, Training pipeline will be triggered. This pipeline includes model training, Model evaluation and model validation. We can utilise **PyCaret** for model training. Once the model is trained, model is evaluated and then the model is validated on the unseen test data. The training pipeline is created in **Airflow**.

Model training >> Model Evaluation >> Model Validation

Data and Model Experimentation:

A lot of experimentation and Hyper-tuning is required to select the best performing model. Once done the model will be trained and evaluated and validated on unseen data and the results are captured and stored as metadata using **MLFlow**.

For tuning models we are using PyCaret and each such experiments are tracked and stored in MLFlow. Once we have sufficient experimentation completed and stored, and the model has achieved required defined metrics, we can select corresponding model and make it ready for staging.

Model Registry:

Once we select the best performing model it has to be unit tested prior to staging.

The selected model will be loaded for the unit testing this will done using **PyTest**.

Once all the unit tests are successfully executed and passed, the model is promoted to Staging and loaded for Integration testing.
On successful completion of Integration testing, the model is Promoted to production. **MLFlow** provides an UI for promoting the Model to Production.

Automation of Inference Pipeline:

The Inference Pipeline is created in **AirFlow**.

Once the model is pushed to Production, **MLFlow** Server is utilised to expose the model as endpoint and can be now utilised as Inference Service or Predictive Service.

The Live production data is sent to data pipeline for data cleaning and data preparation tasks so as to ensure that the new live production data is in the correct format as required in the Inference pipeline.

The predictive Service generates the prediction as output. The service logs are generated and are sent to Data and Model Monitoring.

Continuous Monitoring Pipeline:

Data and Model Monitoring:

The service logs from the predictive service in the Inference Pipeline is sent for Performance monitoring of the deployed Model. The data and model monitoring is done using **Evidently**. Evidently keeps a track of model performance and monitors for any sudden or gradual drift in the model performance. It also logs the performance and issues alerts in case if the drift exceeds the defined threshold limit.

Handling Drift in Model Performance:

When talking about drift, a number of different closely related terms may come up: covariate shift, concept drift, data drift, model drift, model decay, dataset shift, distribution shift.

Once drift is detected, the model scores would have shifted. It's important to understand what the root causes are: which feature accounts for the drift. Techniques for measuring feature importance are invaluable in gauging root causes; drift in important features is more likely to cause degradation in performance than drift in less important features. However, standard feature importance metrics need to be adjusted for drifts and not just prediction importance. Features causing drift can be different from the most important model features.

With passing time, the model performance is expected to degrade and therefore if the drift is detected, the model is retrained on the new data sets and the drift is managed.

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

Even after continuous training of the model, the drift exceeds the defined threshold limit. In such a scenario, the training pipeline is executed again and the best performing model is promoted to production. The best performing model is identified by experimentation and hyperparameter tuning. Generally, for the given use case, the defined threshold in performance drift is 10%.

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

In case there are new sets of annotated data, the data pipeline is triggered which will perform data cleaning and data preparation steps. Once the data pipeline is successfully completed, Training pipeline is triggered ,which trains the model and evaluates the model. The model is validated on the newer unseen test data and its performance is captured. After the number of experiments, the best model is selected and model registry process starts.

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

The new data is cleaned and prepared by triggering the data pipeline prior to executing the Inference Pipeline, so as to make sure the new annotated data that is sent to inference pipeline is in the correct format for Inference Pipeline tasks.
