AI Operationalization in the Cloud for healthcare systems

Project Description:

CloudMedical is a multinational medical company active in the following areas:

1. Bioinformatics and Personalized medicine with the help of smart bracelets
2. Medical Surveillance and Epidemiology
3. Planning and delivery of medical equipment

The company develops and uses various Artificial Intelligence (AI) based solutions for the different areas mentioned above. These include solutions such as understanding what medicines works well for different individuals based on their medical records and genetic data (personalized medicine), predicting the possible diseases that might occur in the future in a given locality based on the health and disease data of the patients in that locality (epidemiology), etc.

As more and more AI solutions are being developed across these different areas, the company is planning to develop one AI based system which can support and serve all the AI requirements across the different areas of the company.

CloudMedical has decided that the system needs to be built on a public cloud service and the **expected functionalities of the system** are as follows :

1. Train the model ad-hoc, batch or dynamic:

   Figure 1 shows an AI process flow in general. The most important step in any AI process is to provide training data (known data) to an AI algorithm. The algorithm then learns patterns from the training data to generate a trained model. A model is basically a mathematical representation of a real-world process.

   This trained model can then perform predictions/classifications. For example, in the case of weather forecasting, the last 30 days of weather data including parameters like wind speed, day of the week, humidity, temperature etc can be use to train an AI algorithm which then generates a trained model. This trained model can then predict the weather for a given day or the next days based on the day of the week, humidity etc.

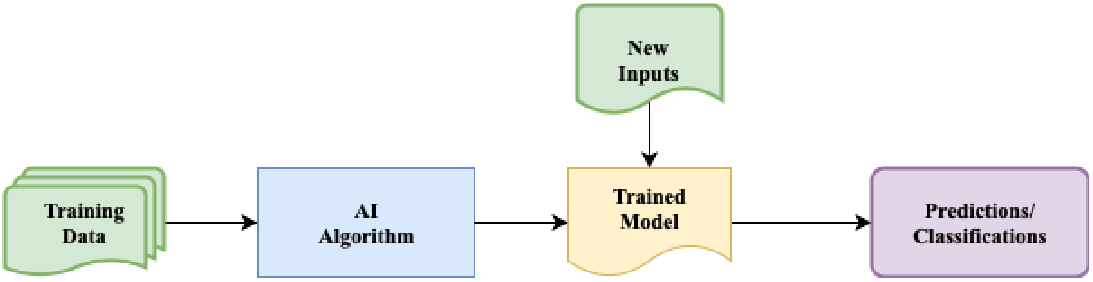

Fig1: AI Process flow

Such training can be performed in three different ways :

1.1 Ad-hoc Training

This method allows a data scientist to train and generate a model as and when required . This model is further pushed to production where it is used to perform some predictions, classifications etc. This is removed from production if the model is not giving accurate results and generated again as and when required.

1.2 Batch Training

Batch training provides a way by which it is possible to have a constantly refreshed version of the model in production. As opposed to the first method, this allows the models to be re-trained at periodic time intervals using the latest data so as to ensure that the model is up-to-date with the latest patterns in data.

1.3 Dynamic Training

There are some AI models which supports online learning where learning can be performed by the algorithms in real-time. This method allows such models to be trained dynamically using real-time data.

The data required for training the model is obtained from different sources depending upon the use case and the area of application. CloudMedical **requires the AI system to support all these three method**s of training a given model.

2. Serve the model either via batch or via REST API

   The second important step in AI process is to ensure that the model, built using the above steps, is available for making predictions/recommendations etc. This means the model that has been created needs to be either provided as a file or in a database, which can then be used by the application(s) to generate predictions or recommendations. For example, in the case of personalized medicines, the app used by the user needs to make predictions on the recommended medicines, possible health conditions etc, based on the health data obtained from the smart bracelet. In this case, the app can use an ml model which is stored locally in phone along with the app. This model is then centrally updated periodically in batches and pushed to the edge during the deployment phase.

Another method is through REST API where the model will be served using a web-service which can be invoked by the applications using a REST call. For example, in the above mentioned example, the app instead of having the model locally in the phone can send a REST call to a web-service by passing the input required for the model as a JSON.

These **both methods for serving the AI models needs to be supported by the envisioned system**.

Key Users of the System :

The end users of the system will be:

1. Data Scientist building the AI models
2. Data Engineer defining data pipelines
3. Developers building AI-powered solution consuming models
4. The end user wearing the device

Traditionally, Data scientists build AI models using Jupyter notebooks. It is basically a cross-language platform which provides a web interface for developing, documenting and interfacing with some back end system to run AI algorithms. Most of the data scientists use Python although some use R for programming.

Models can be trained in ad-hoc fashion by the Data Scientist that can build and tune it. However, data scientists may not be trained engineers and many a times they send the jupyter notebooks to the production engineers who are responsible for handling the deployment. It can happen that the engineer does not understand the code well due to poor documentation practices or the code might be inefficient. It may also happen that once the solution is in production there is no inherent feedback system for improvement and updates. For certain systems, model performance is likely to suffer over time and monitoring isn’t always standard practice. Data scientists also aren’t really trained to do things like writing good test cases, so it can happen that the model updates going right from Jupyter to production occasionally break applications or serve incorrect predictions.

To this end, CloudMedical would like to establish a more operationalized practice in order that the whole process can be held in batch without additional intervention. This implies that

models must be versioned and stored in a specific repository and made it available to the serving layer.
Additionally, the **system must constantly evaluate the accuracy** of the model without the help of the Data Scientist. A more advanced feature **dynamic training** when the model is continuously updated from new data.

<u>What has been Done ?</u>

CloudMedical has already provisioned the following architecture components:
1) Source Code repository for hosting the source code of the model and persisting the structure of the model itself
2) A high-performance message broker (like Kafka) that can handle the events
3) A Stream Processing runtime that can process the events as they arrive
4) An in-memory data store that can facilitate data access for building the model by the Data Scientists
5) An HDFS storage hosting both transactional and log data from CloudMedical where Spark and Hive Jobs can process data
6) A batch computing infrastructure that can run scheduled jobs for training and scoring
7) An API/serving layer where each model is wrapped in a microservices

The following table provide the specifications of the infrastructure already available at CloudMedical's Cloud .
The specifications for the API, when possible, follow the Open API Specification https://github.com/OAI/OpenAPI-Specification

Pre-existing components
Specification of the core part of the system that are already in use by CloudMedical

Component of the system	API Specification	Implementation details	Security access implemented	Intended use	Version
Source Code repository	https://developer.github.com/v3/	Github client for individual workstations, Github Enterprise apis for Azure DevOps builds https://docs.microsoft.com/en-us/azure/devops/boards/github/connect-to-github?view=azure-devops	Oauth **https://docs.microsoft.com/en-us/azure/devops/boards/github/connect-to-github?view=azure-devops#server-github-ent-oauth-register**	Source code versioning and control including Jupyter notebooks + ML Model persistence (serialization)	2.24

High performance message broker (kafka)	There are different application specific semantics of connections for producers, consumers, connect and streams For producers and consumers one can use the specific client or Kafka Proxy for Rest : the details for Kafka proxy are here https://github.com/confluentinc/kafka-rest	Individual clients can produce/consume messages with the following specifications https://kafka.apache.org/documentation/#api For Kafka connect there is a specific Rest Api here https://kafka.apache.org/documentation.html#connect	Possible authentication mechanism are ASL/GSSAPI (Kerberos) - SASL/PLAIN - SASL/SCRAM-SHA-256 and SASL/SCRAM-SHA-512 SASL/OAUTHBEARER https://kafka.apache.org/documentation/#security	Underling infrastructure serving real time data pipeline and streaming applications consuming data	2.3
Stream Processing Runtime	Apache Kafka Streams this provide a DSL abstraction upon the underling messaging system https://kafka.apache.org/23/documentation/streams/developer-guide/dsl-api.html	For creating a stream form a Kafka Topic see https://kafka.apache.org/23/documentation/streams/developer-guide/dsl-api.html#streams-developer-guide-dsl-sources For transforming it https://kafka.apache.org/23/documentation/streams/developer-guide/dsl-api.html#transform-a-stream The	The stream security details **https://kafka.apache.org/10/documentation/streams/developer-guide/security.html**	Used to store , process and distribute events produced by all the data producers.While all data is first injected into a topic, in order to be further transformed, the Streams DSL provides a different abstraction and speaks in terms of **KTable** and **KStreams** and the elementary piece of data is called record or fact instead of message	2.3
In memory Database (An IMDB is a Database System that primarily relies on memory and not on disk storage it resides in the cloud)	Azure Data Explorer (ADX) uses this api specification https://github.com/Azure/azure-rest-api-specs/tree/master/specification/azure-kusto/resource-manager Implementation details are here	The system is intended to query both streaming and historical data https://docs.microsoft.com/en-us/azure/data-explorer/write-queries	X.509v2 certificate installed locally. X.509v2 certificate given to the client library as a byte stream. AAD application ID and an AAD application key (the equivalent of username/password authentication for applications). previously-obtained valid AAD token	The system is intended to query both streaming and historical data for Data Science purposes using Jupyter Notebooks	Cloud service latest version provided

HDFS	Azure storage Data Lake Rest API specification https://github.com/Azure/azure-rest-api-specs/tree/master/specification/storage/data-plane	This service is a Storage layer that offers HDFS access https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-use-hdfs-data-lake-storage The Rest apis for the services https://docs.microsoft.com/en-us/rest/api/storageservices/data-lake-storage-gen2	Access to the resource is granted through a SAS Key https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-access-control#access-control-lists-on-files-and-directories	Managed storage service that can be accessed in an Hadoop compatible fashion and hosts Analytics workloads	Latest provided by the service
Batch computing	Azure batch Rest API specification https://github.com/Azure/azure-rest-api-specs/tree/master/specification/batch/data-plane	This service is used to instantiate and run large scale parallel jobs. For an overview of the service https://docs.microsoft.com/en-us/azure/batch/batch-technical-overview For the APIs of https://docs.microsoft.com/en-us/rest/api/batchservice/	Calls to the Rest APIs need to be authenticates by either an access Key or a AAD account https://docs.microsoft.com/en-us/rest/api/batchservice/authenticate-requests-to-the-azure-batch-service	Provide batch execution of long running jobs needed for training the CloudMedical AI System	Latest provided by the service
API Management	Azure APIM Rest API specification https://github.com/Azure/azure-rest-api-specs/tree/master/specification/apimanagement/resource-manager	Service used to describe and expose a Rest API to end user's applications . https://docs.microsoft.com/en-us/rest/api/apimanagement/	This service can be used to protect APi access https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-protect-backend-with-aad As for the service itself it offers this security layers https://docs.microsoft.com/en-us/azure/api-management/api-management-security-controls	The service is used by CloudMedical for both external and internal exposition of APIs in particular the scoring offered by the ML enabled products	Latest provided by the service

Main Constraints:

The AI system needs to satisfy the following constraints:

1) 8 Data Scientists working 200 h a month each

2) The system will start hosting an initial set up of 10 specific projects each one will have a

model that is built continuously during the working hours
3) Each event (Train/ Serve / Ingest data) is handled through messages that can signal incoming invocations from external sources (like devices) and internal invocations as well
4) Each model is consumed by an API with calls from applications that can hit 200 calls/sec.
5) For the batch processing, data is fed via files in csv format with maximum size 1Tb
6) Data is passed to the API by Json document with the relevant features to perform the predictions/scoring.

Project Requirements and Deliverables:

First Deliverable:

1. Identify a potential target reference architecture for the AI system that takes into account the key functionalities and the constraints of the system
    a. Identify the ASR
    b. Identify the architecture style/styles that can be used in the scenario
2. Evaluate the pros and cons of the following choices for using the model for prediction/ recommendation (Scoring):
    a. Rest API (each scoring of the model is an invocation to an API Call)
    b. In app (the application holds the trained model and performs the scoring when requested)
    c. In database scoring: model is pushed in the database that massively scores it on the relevant dataset
    d. Asynchronous via messaging : Implement a pub sub architecture where a stream of messages will be evaluated on the fly

Second Deliverable:

CloudMedical would like to put the Reference Architecture to the test by deploying it in the following scenario: they would like to offer the feature to their personal smart bracelet with heart rate monitor to alert with a possible heart Disease :

1. A sample of the data from which the model is built is available here https://archive.ics.uci.edu/ml/datasets/Heart+Disease
2. The incoming data is passed from the smart bracelet via an API Call where the feature of the customer is specified in a Json document

3. Model is scored and the Json is augmented in terms of the prediction and accuracy value
4. Each individual value is stored in the DataLake for future reference
5. The model is built on a subset of all the data

Third Deliverable:

CloudMedical see this project as a strategic endeavor and they would like to cast the consideration in the light of the blog entry:

https://martinfowler.com/articles/data-monolith-to-mesh.html

Revisit your considerations and assumption in the light of this article:

1. Is the DataLake component necessary in the overall picture?
2. How one enforces Domain Specific Design in the solution of the second deliverable?
3. How one can achieve the capabilities of Discoverable, Addressable, Trustworthy, Self-described, Inter operable and Secure described in the article?

References:

1. MLOPs https://mlops.org/about/
2. Stanford's dawn https://dawn.cs.stanford.edu/
3. Uber's Michelangelo https://eng.uber.com/michelangelo/
4. Databricks MLFlow https://www.mlflow.org/
5. FBLearner https://code.fb.com/ml-applications/introducing-fblearner-flow-facebook-s-ai-backbone/
6. https://medium.com/analytics-and-data/overview-of-the-different-approaches-to-putting-machinelearning-ml-models-in-production-c699b34abf86