End-to-End ML Model Monitoring using Airflow and Docker

Overview

Let’s consider you have developed a machine learning model with high accuracy by using the raw data. The next step is to put this highly accurate model into production by fetching the data in daily batches. But the model in prediction can fail without any warning. Due to the same keeping, an eye on the model’s prediction power becomes necessary.

Also, in the real world, data can change due to various factors affecting prediction accuracy. In production, taking a major to prevent such changing behavior of model and data is important to serve the business problem over time.

In this project, we will build an end-to-end pipeline to monitor any changes in the model's predictive power or degradation of data, commonly known as Model and Drift Monitoring. The model is built to classify whether or not a loan is to be given or rejected based on data fetched from PostgreSQL. This project also demonstrates the orchestration of ML pipelines using Docker and Airflow.

Aim

1. To fetch the data from the PostgreSQL server
2. To Build a Loan Eligibility classification mode that predicts whether the loan is to be given or refused
3. To monitor Concept, Data, and Model drift
4. Orchestrate monitoring pipeline with Airflow

Tech Stack

- ➔ Language: Python
- ➔ Libraries: pandas, numpy, matplotlib, scikit-learn, deepchecks, sqlalchemy, psycopg2-binary
- ➔ Services: Airflow, Docker, PostgreSQL

Approach

1. Extracting Data from PostgreSQL
2. Data Preprocessing
    a. Train-Test split
    b. Encoding Categorical Variables
    c. Imputing Missing Values

    d. Rescaling the Data
    e. Feature Engineering
3. Model Training and Evaluation
    a. Random Forest
    b. Gradient Boosting
4. Model Monitoring
    a. Concept Drift
    b. Data Drift
    c. Model Drift
5. Orchestration with Airflow

Note:

1. **Kindly follow the readme.md file for running the code**
2. **This project requires the data to be present in postgres server. The data is available in code>main>dags>data>raw location. Kindly upload the data and provide the appropriate credentials in code>main>dags>creds.json file.**

Project Takeaways

1. Connecting Python to PostgresSQL server
2. Conditional extraction of Data from PostgreSQL
3. Missing Value imputation
4. Rescaling the data
5. Feature Engineering
6. Checking Data Sanity
7. Building and Evaluation Random Forest model
8. Building and Evaluating Gradient Boosting model
9. How to generate a model deployment report
10. What is Concept Drift?
11. How to monitor concept drift and generate the report
12. What is Data Drift?
13. How to monitor Data Drift and generate the report
14. What is Model Drift?
15. How to monitor Model Drift and generate the report
16. How to install Docker?
17. How to compose Docker container
18. How to send messages to slack using airflow
19. How to create DAG in Airflow
20. Orchestrating pipeline using Airflow