Mlops Final Project

Ali Musharaf Baig · Follow
6 min read · 2 days ago

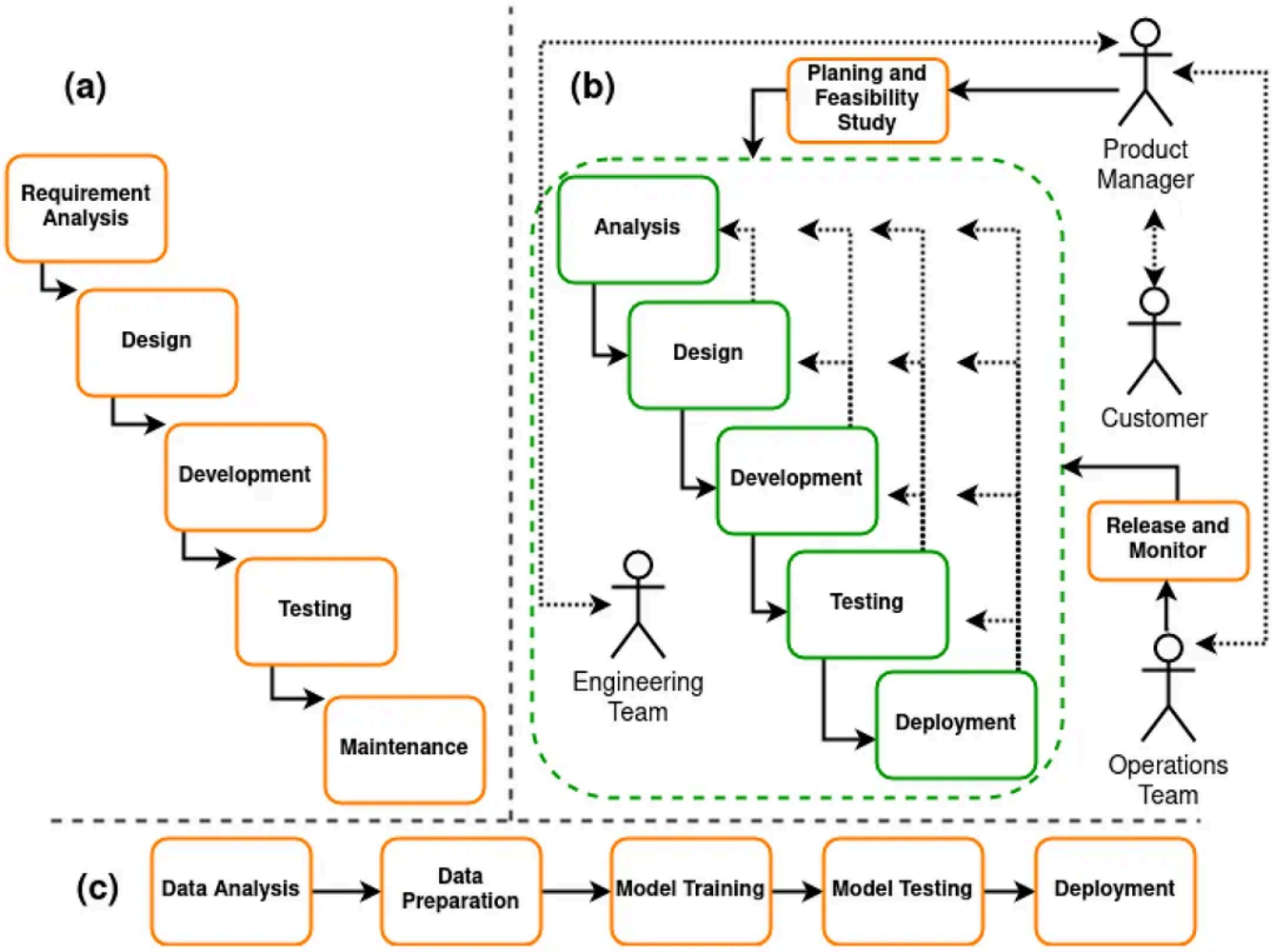
Streamlining Machine Learning with MLOps: A Hands-on Guide with DVC and Airflow

Machine learning projects are often plagued by challenges like untracked data, inconsistent workflows, and lack of automation. These issues hinder scalability and reproducibility. Enter MLOps— the practice of applying DevOps principles to machine learning pipelines.

In this blog, we'll dive into how MLOps practices were applied to a real-world weather prediction project. By leveraging tools like DVC, Apache Airflow, and MLflow, we automated and streamlined the workflow, ensuring:
- Reproducibility: Versioning datasets and models with DVC.
- Automation: Orchestrating tasks with Airflow.
- Scalability: Building pipelines that adapt to new data.

By the end of this blog, you'll gain practical insights into integrating MLOps into your own projects.

Ml work flow for this project:



Work flow

Project Overview

Objective:

The goal of this project was to predict temperature using weather data such as wind speed and humidity. The focus was on building a robust, automated workflow that could handle:

1. Data Collection: Automating data retrieval via APIs.
2. Data Preprocessing: Normalizing and preparing data for modeling.
3. Model Training: Using a linear regression model for predictions.

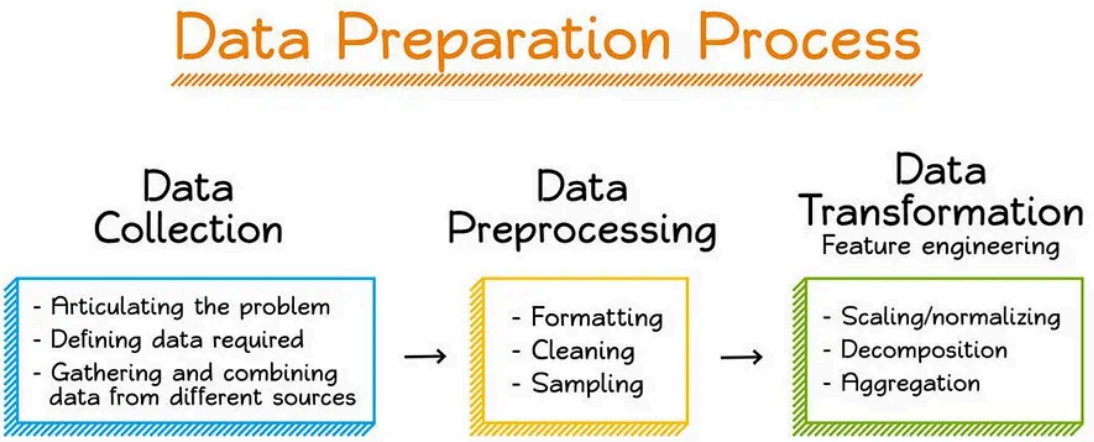
4. Automation: Employing Airflow to streamline workflows.
5. Versioning: Using DVC for dataset and model tracking.

Workflow Overview
Here's a high-level view of the workflow:

1.Data Collection: Weather data fetched using APIs.
2. Data Preprocessing: Automated normalization and feature extraction.
3. Model Training: Training and saving models using DVC for version control.
4. Deployment: Serving predictions via a Flask app.

Tools Used:
- DVC: For tracking and versioning datasets and models.
- Apache Airflow: For workflow automation.
- MLflow: For model tracking and versioning.
- Flask: For serving predictions via APIs.
- SQLite: For lightweight database needs.
- GitHub Actions: For CI/CD pipeline automation.

Data Collection and Preprocessing



Data Collection:
Data was collected using the OpenWeatherMap API. The Python script `fetch_weather.py` automated this process, ensuring consistent retrieval. Below is the key code snippet:

python:
import requests

API_KEY = “api_key”
BASE_URL = “http://api.openweathermap.org/data/2.5/forecast"

def fetch_weather_data(city):
params = {“q”: city, “appid”: API_KEY, “units”: “metric”}
response = requests.get(BASE_URL, params=params)
return response.json() if response.status_code == 200 else None

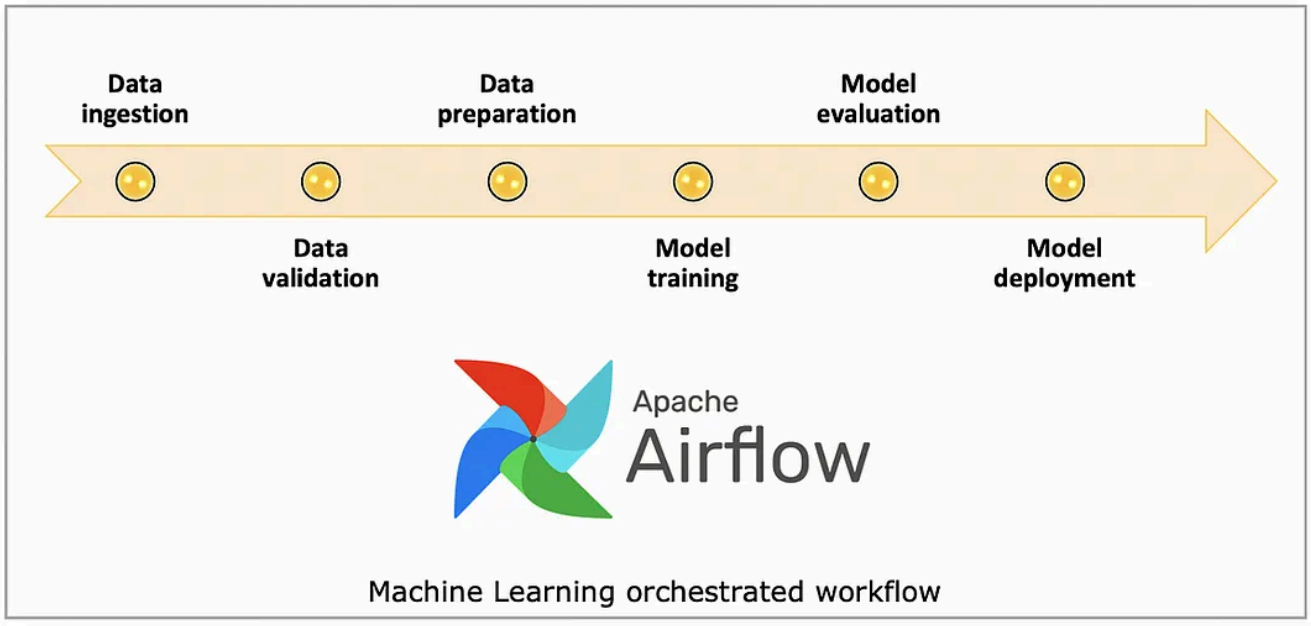
data = fetch_weather_data(“London”)

```
#Data Preprocessing

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df['Humidity'] = scaler.fit_transform(df[['Humidity']])
df['Wind Speed'] = scaler.fit_transform(df[['Wind Speed']])
```

Version Control with DVC

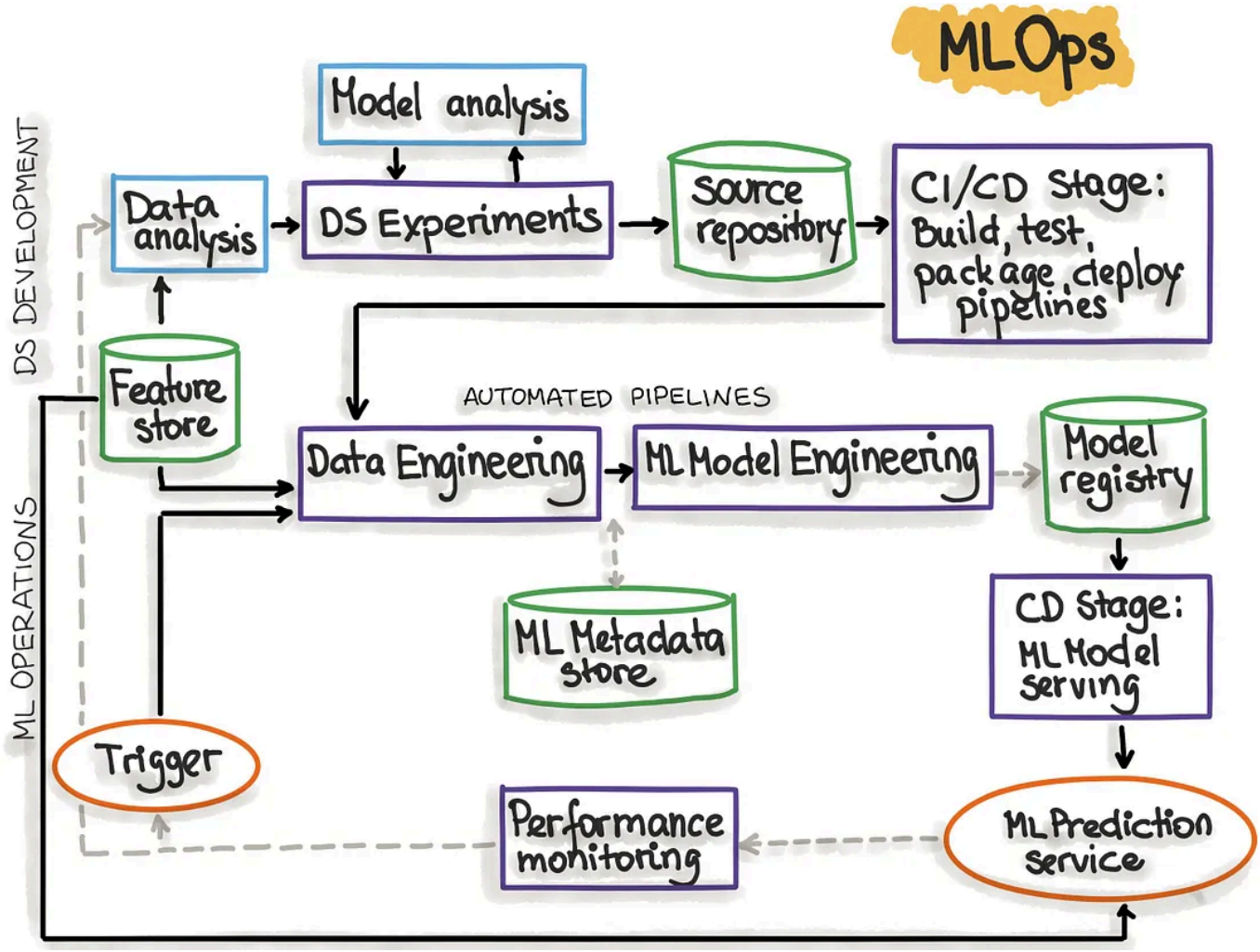
```
Initialize DVC
dvc init

Add dataset to DVC
dvc add raw_data.csv

Commit and push to remote storage
git add raw_data.csv.dvc .gitignore
git commit -m "Track dataset with DVC"
dvc push
```

Workflow Automation with Apache Airflow



In machine learning projects, repetitive tasks like data fetching, preprocessing, and model training can consume valuable time and introduce errors. Workflow automation tools like Apache Airflow help orchestrate these tasks in a structured and repeatable manner.

Setting Up Airflow:
Apache Airflow was set up to automate the following tasks:
1. Fetching Data: Automating weather data collection using the OpenWeatherMap API.
2. Preprocessing Data: Cleaning and normalizing the dataset.
3. Model Training: Retraining the model whenever new data becomes available.
4. Versioning: Tracking datasets and models using DVC.



Below is an example of an Airflow DAG (Directed Acyclic Graph):

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime

def fetch_data():
# Code to fetch weather data
pass

def preprocess_data():
# Code to preprocess data
pass
```

```
def train_model():
# Code to train the model
pass

with DAG(
dag_id="mlops_pipeline",
schedule_interval="@daily",
start_date=datetime(2023, 1, 1),
catchup=False,
) as dag:
fetch = PythonOperator(task_id="fetch_data", python_callable=fetch_data)
preprocess = PythonOperator(task_id="preprocess_data",
python_callable=preprocess_data)
train = PythonOperator(task_id="train_model",
python_callable=train_model)

fetch >> preprocess >> train
```

Significance of Automation

- **Error Reduction**: By automating repetitive tasks, we minimized the risk of human error.
- **Scalability**: New tasks (e.g., monitoring, retraining) can easily be added to the pipeline.
- **Time Savings**: Automation freed up resources for higher-value tasks like model optimization

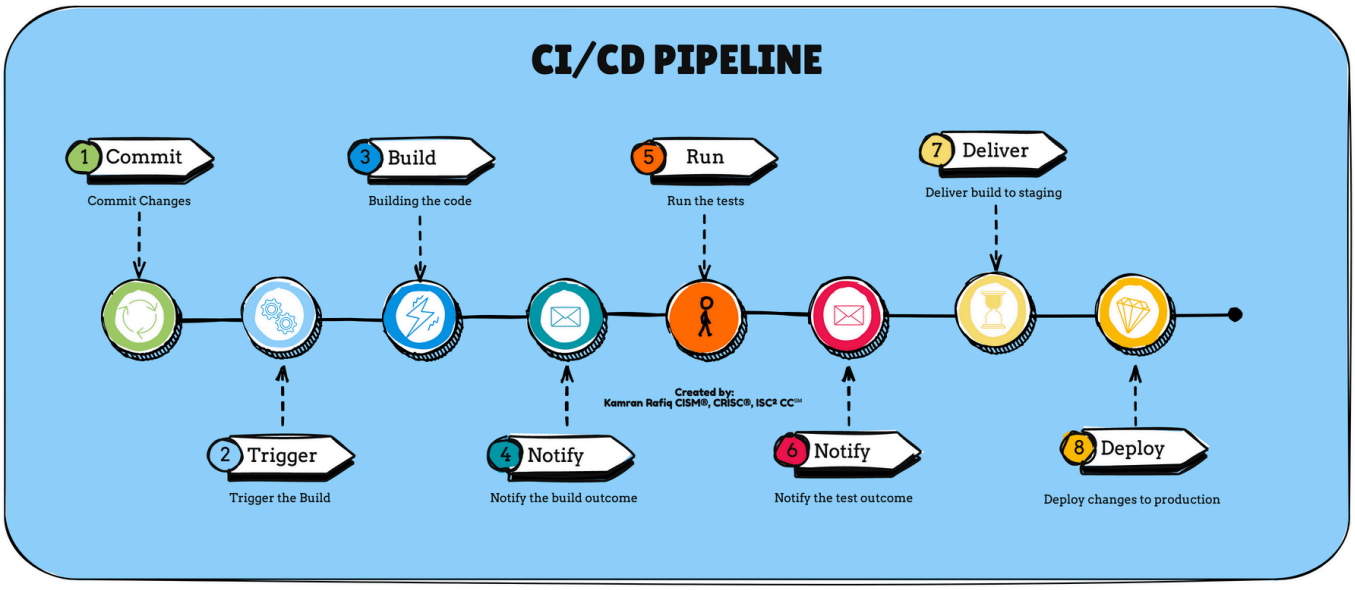
Model Training

The model training process involved using a linear regression algorithm to predict temperature based on weather features. The training script was integrated into the pipeline and automated via Airflow.

Below is an example of the training code:

python:

from sklearn.linear_model import LinearRegression from sklearn.model_selection import train_test_split import pickle # Split data X = df[[‘Humidity’, ‘Wind Speed’]] y = df[‘Temperature’] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Train model model = LinearRegression() model.fit(X_train, y_train) # Save model with open(“model.pkl”, “wb”) as f: pickle.dump(model, f)



DVC was used to track the trained models, ensuring reproducibility:

Track the model file:
dvc add model.pkl

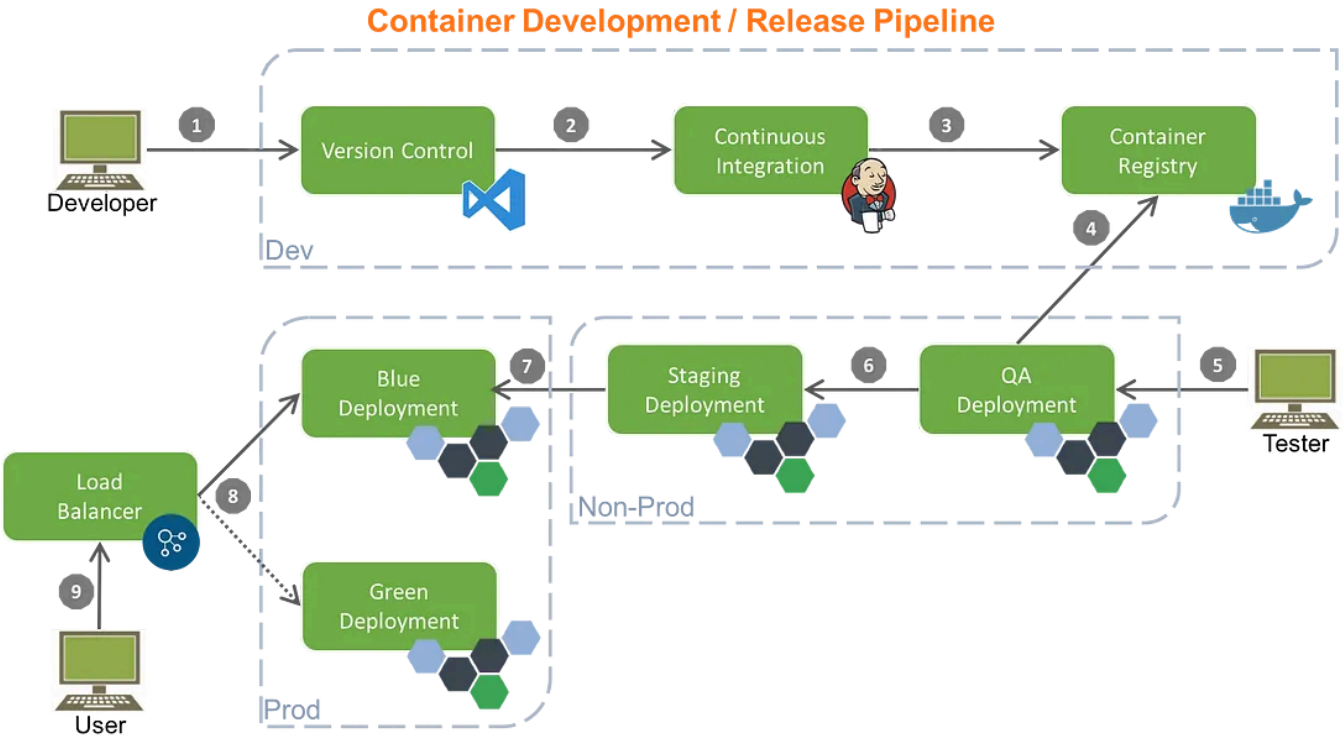
Commit and push the model:

git add model.pkl.dvc .gitignore

git commit -m “Track model with DVC”

dvc push

Deployment and Docker Integration:



Deployment Overview:

To make the application accessible, we deployed it as a Flask-based web service. This involved:

1. Serving the Backend: Exposing the API for predictions.

2. Frontend Integration: Hosting the front-end pages (welcome, login, signup).

3. Database Connection: Using SQLite for user management.

Using Docker for Deployment:
To simplify deployment and ensure consistency across environments, Docker was used to containerize the application.

Here's the `Dockerfile` used:

dockerfile:
Use the official Python image
FROM python:3.11

Set the working directory
WORKDIR /app

Copy the project files
COPY . /app

Install dependencies
RUN pip install — no-cache-dir -r requirements.txt

Expose the port the app runs on
EXPOSE 5000

Start the app
CMD ["python", "backend/app.py"]

Building and Running the Docker Image

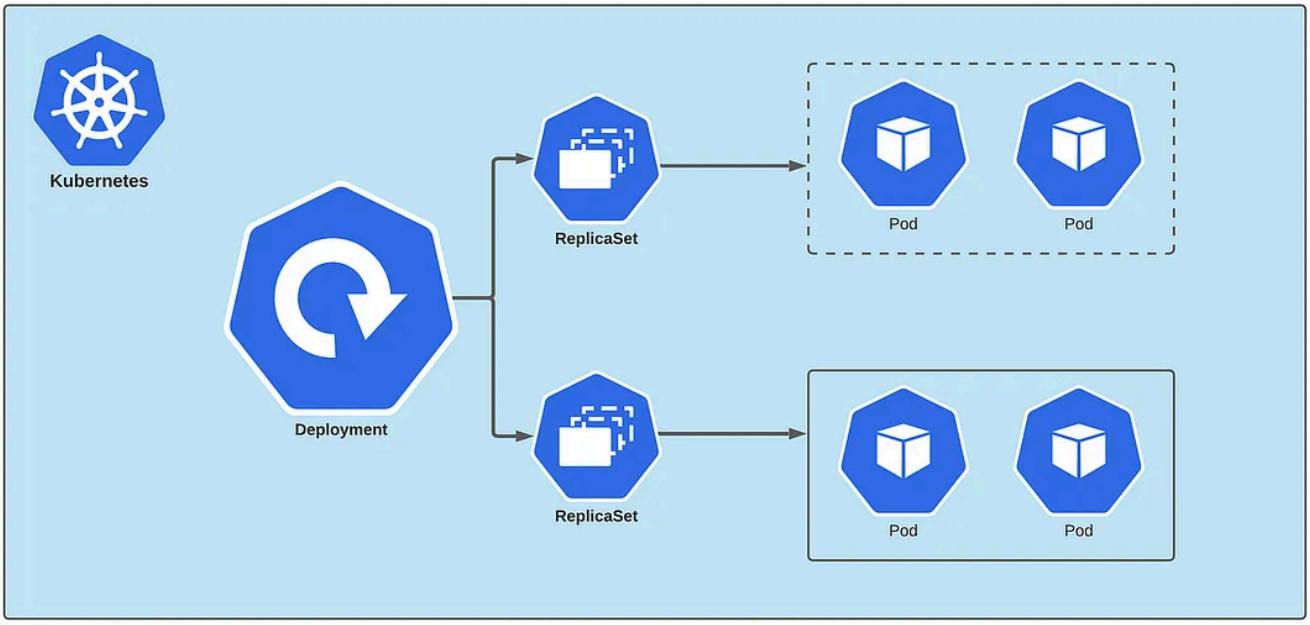
1. Build the Docker image

docker build -t weather-prediction-app .

Run the container:

```
docker run -p 5000:5000 weather-prediction-app
```

Kubernetes Deployment



For scalability, the Docker image can be deployed in a Kubernetes cluster. A simple Kubernetes deployment file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: weather-app
spec:
replicas: 3
selector:
matchLabels:
app: weather-app
```

template:
metadata:
labels:
app: weather-app
spec:
containers:
— name: weather-app
image: weather-prediction-app:latest
ports:
— containerPort: 5000

Key Learnings and Conclusion:

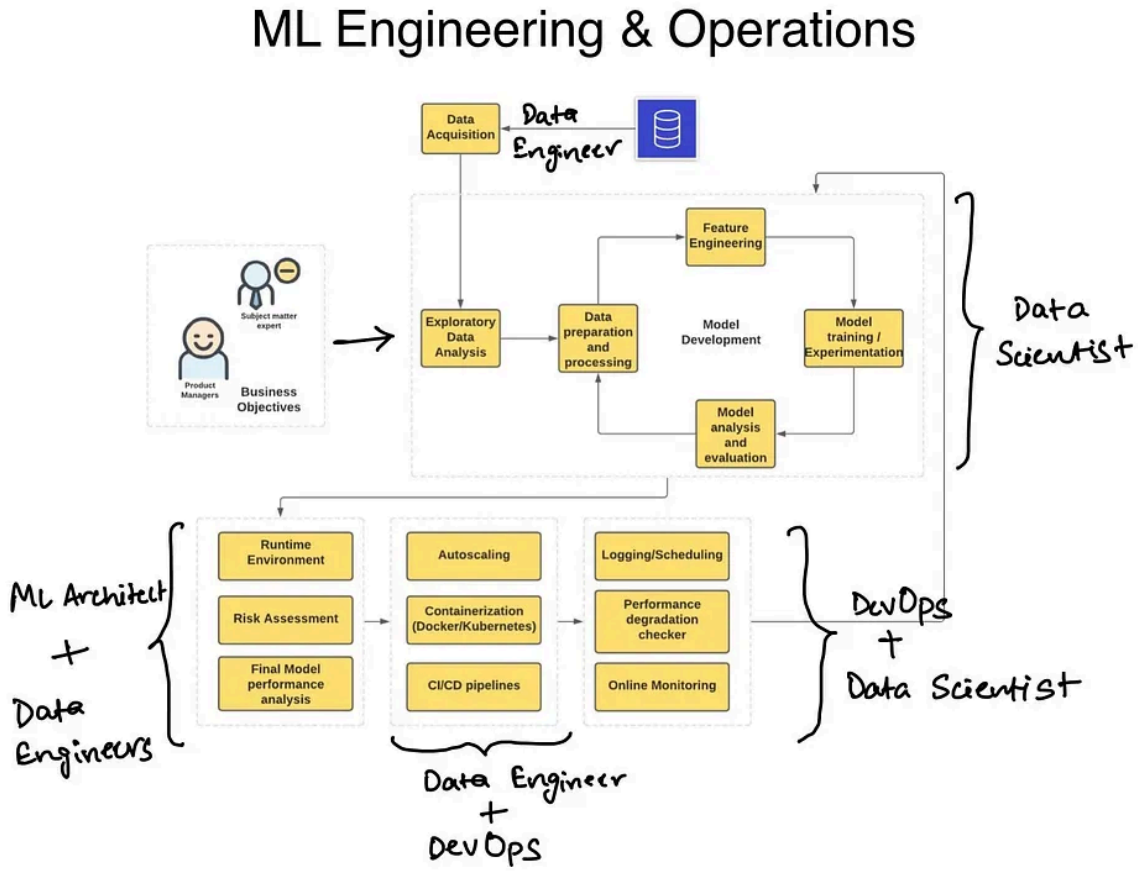
Key Learnings

This project provided valuable insights into MLOps practices:
- Version Control: Using DVC ensured reproducibility for datasets and models.
- Workflow Automation: Apache Airflow streamlined repetitive tasks, reducing manual errors.
- Scalability: Docker and Kubernetes made deployment and scaling seamless.
- Collaboration: Tools like GitHub and CI/CD pipelines simplified team workflows.

Conclusion

MLOps is not just a buzzword — it's a necessity for real-world machine learning projects. Tools like DVC, Airflow, and Docker not only enhance

reproducibility but also ensure scalability and reliability. By adopting MLOps practices, we can build systems that are production-ready and resilient.



Whether you're a beginner or a seasoned ML engineer, incorporating MLOps into your projects will undoubtedly elevate their impact. Start small, automate processes, and watch your projects transform.

Written by Ali Musharaf Baig

0 Followers · 1 Following

Follow

Recommended from Medium

In GoPenAI by Sri Varshan

Machine Learning Project: Real Estate Price Prediction

Real Estate Price Prediction is the process of estimating or forecasting the future prices of...

3d ago 20

Komal Agrawal

MLOps

MLOps, or Machine Learning Operations, is a set of practices that combines machine...

Oct 13 20

Lists

Staff picks

780 stories · 1491 saves

Stories to Help You Level-Up at Work

19 stories · 891 saves

Self-Improvement 101

20 stories · 3122 saves

Productivity 101

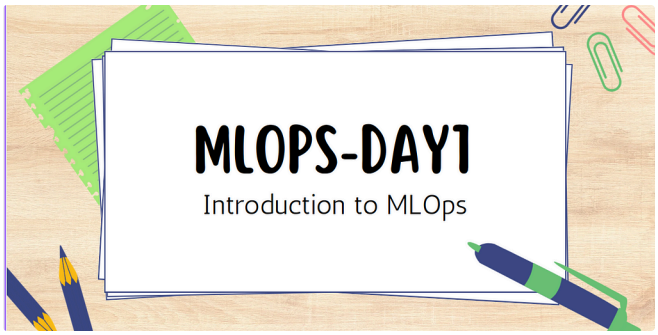
20 stories · 2637 saves

Carlos Argueta

Robot Perception: Prompting Grounded SAM 2

Explore how Visual Language Models like Grounded SAM 2 revolutionize robot...

4d ago 37

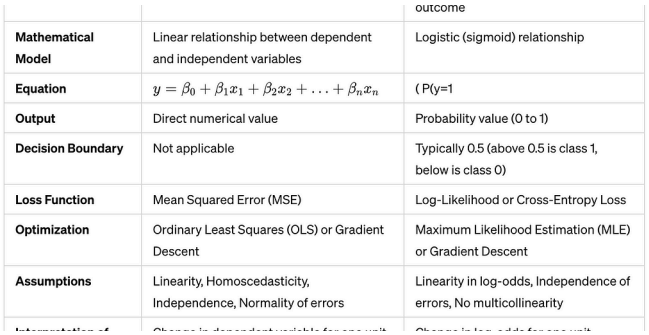


Siddharth Singh

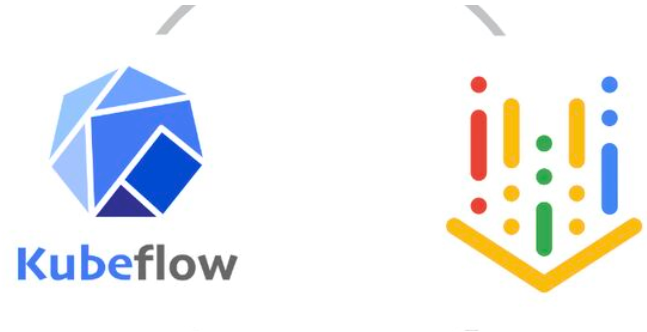
MLOPS Day1: Introduction to MLOps

Bridging Machine Learning and Operations for Scalable, Reliable AI Systems

Nov 26 26 1



		outcome
Mathematical Model	Linear relationship between dependent and independent variables	Logistic (sigmoid) relationship
Equation	$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$	(P(y=1
Output	Direct numerical value	Probability value (0 to 1)
Decision Boundary	Not applicable	Typically 0.5 (above 0.5 is class 1, below is class 0)
Loss Function	Mean Squared Error (MSE)	Log-Likelihood or Cross-Entropy Loss
Optimization	Ordinary Least Squares (OLS) or Gradient Descent	Maximum Likelihood Estimation (MLE) or Gradient Descent
Assumptions	Linearity, Homoscedasticity, Independence, Normality of errors	Linearity in log-odds, Independence of errors, No multicollinearity



Avicsebooks

ML Part10: Supervised Machine Learning: Classification

Classification refers to a type of supervised learning where the goal is to predict the…

Jun 17

Raju Dawadi

MLOps 101 with Kubeflow and Vertex AI

If you've ever tried taking an ML model from your local environment to production, you…

Oct 31 27 1

See more recommendations