What is MLOps

08 May 2024 13:50

MLOps, short for Machine Learning Operations, is a set of practices that aims to streamline and automate the lifecycle of machine learning models. This discipline merges machine learning (ML) system development and Machine Learning operations (Ops) to deliver consistent and efficient deployment and maintenance of ML models in production. The goal of MLOps is to bridge the gap between the development of ML models and their operational deployment, ensuring that they can be effectively scaled and maintained within an operational environment.

mba

mlops = ml system development + ML operations

ml

analogy example

chef /cook

veg noodles

website →

- Scalability
- Reproducibility
- Governance
- Collaboration
- Inventory management

shop → 5 – 10

crores of Rs

why?

Crores of rs

Why?

[5-10] noodles

1L-10L → scale

- 1 shop → more shops
  capital / human
-
- Reproducil / standardization

- → Automation
- → resource manin + inventory
- → Collaboration
- → Governance legal

ml

product + operations

[model development]

Our approach till now!

08 May 2024 13:55

ml lifecycle

csv

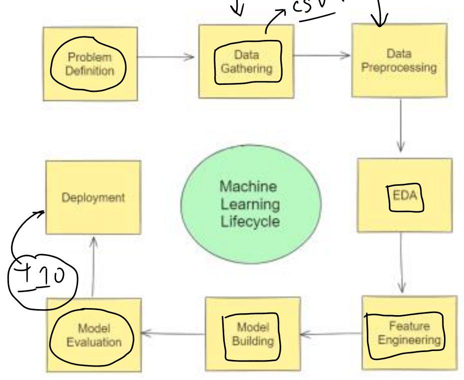
170 180 190 rmse → 10

± 10 cricket

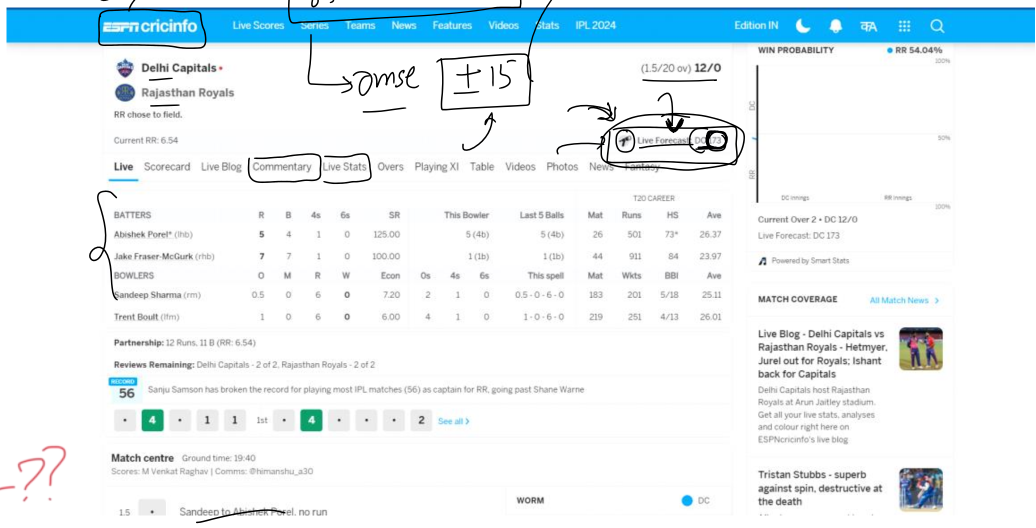
fresher model

165 → 180 → 195

frontend → frontend envs



±10



rmse ±15

ml model → xgboost

serialize —??

pickle/Job .pkl → model

model.pkl

✓ api.py

✓ backend modify

backend dev

✓ frontend ~

github ← push

live →

model → Amazon S3 → model will be stored here.

api → ml envi

flask/fastapi

Bakend → change

173

frontend

→ Drift in the data e.g Impact player.

Problem 1 - The Data Problem

09 May 2024 09:45

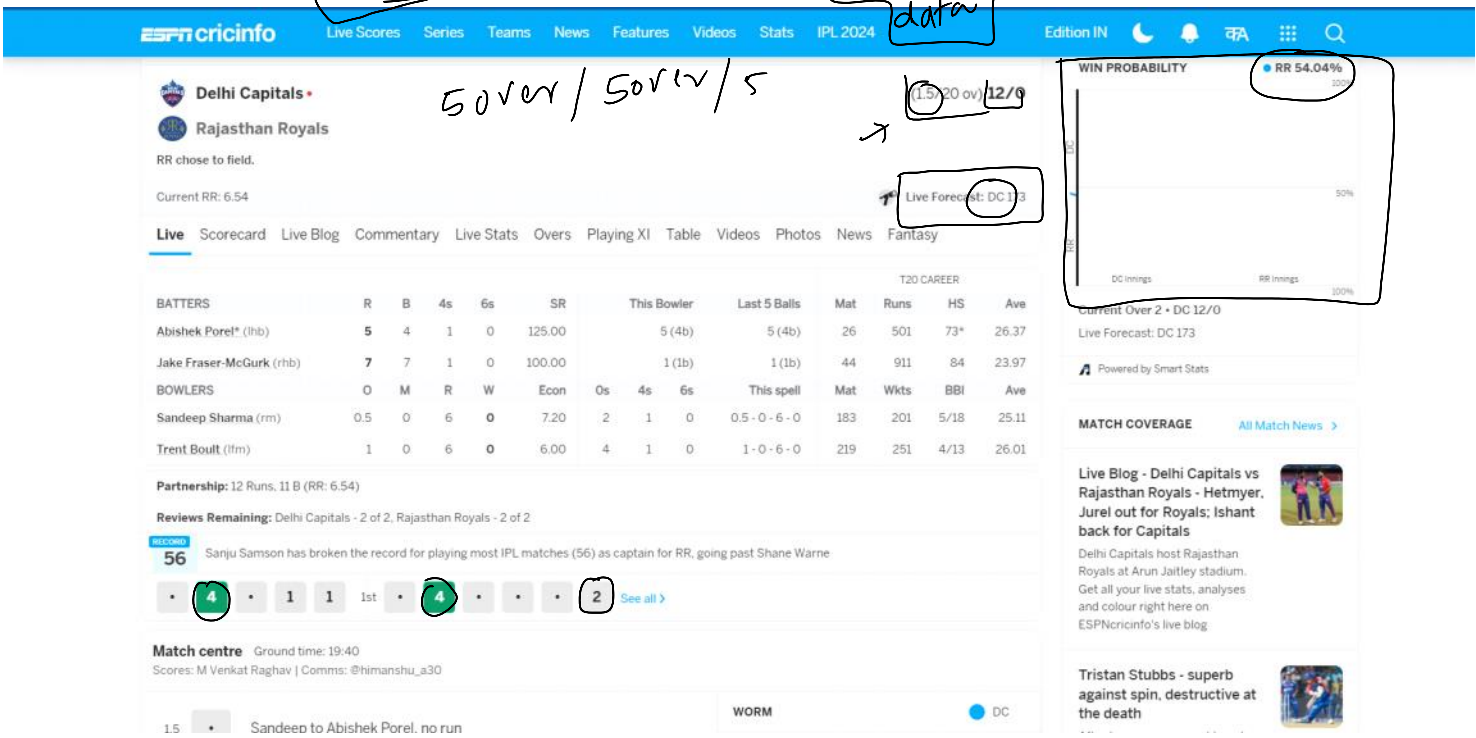
weather

data

historical

currency

real time data



Historic data → DB

Real time scores → streaming service

weather → api

CSV files

py

read_csv.

3 steps

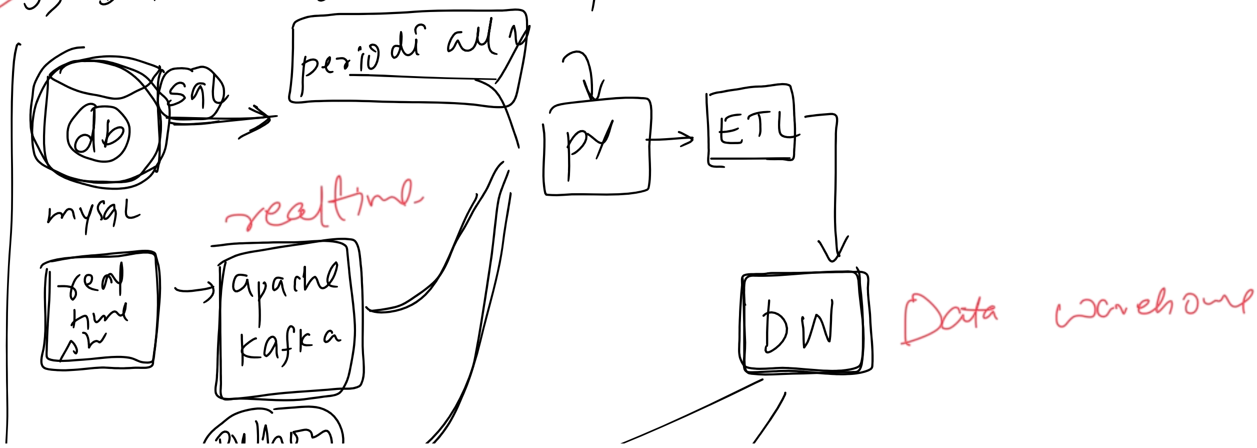
Data management Architecture

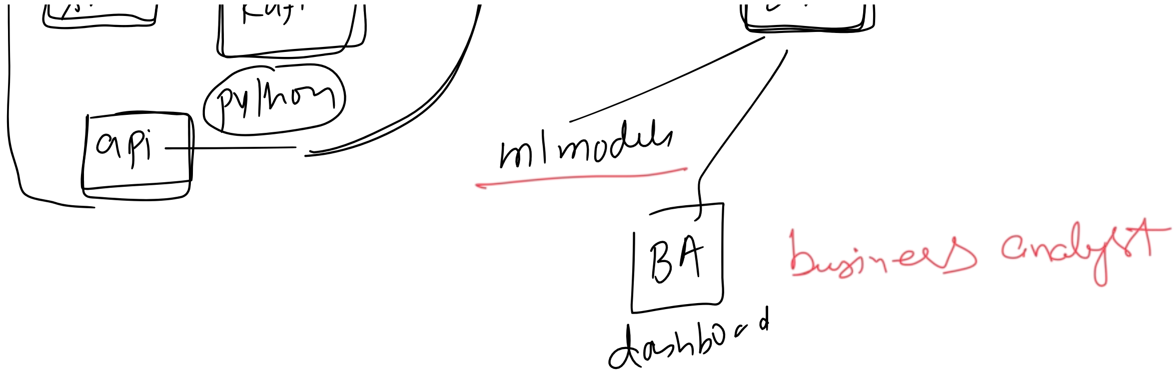
step1 1) Data ingestion

step2 2) Data transformation

s3 3) Data storage 1 week

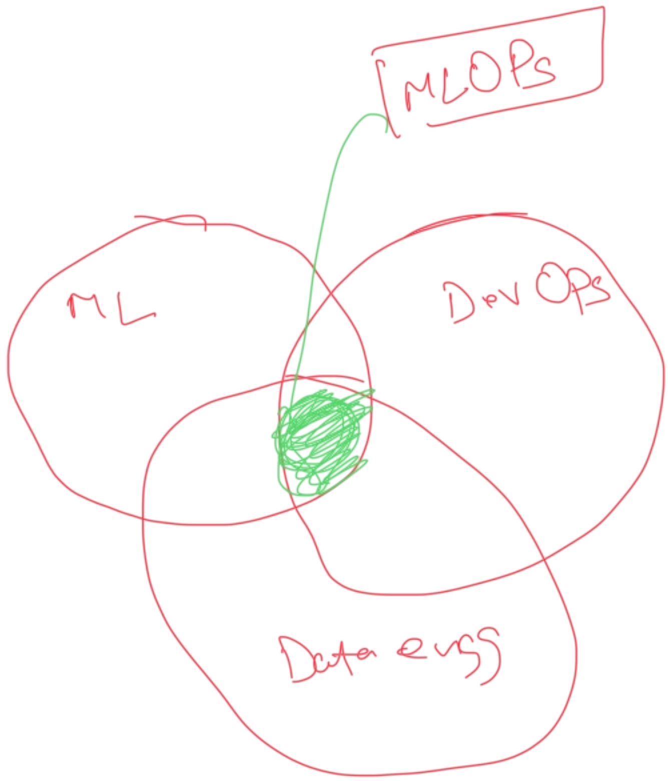
Batch process





1 Data management → Data engineer → ETL

MLOps → Data engineer



Problem 2 - The code problem

Data management

09 May 2024 09:46

```python
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the Iris dataset
data = load_iris()
X = data.data
y = data.target

# Preprocess the data: Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, rando

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

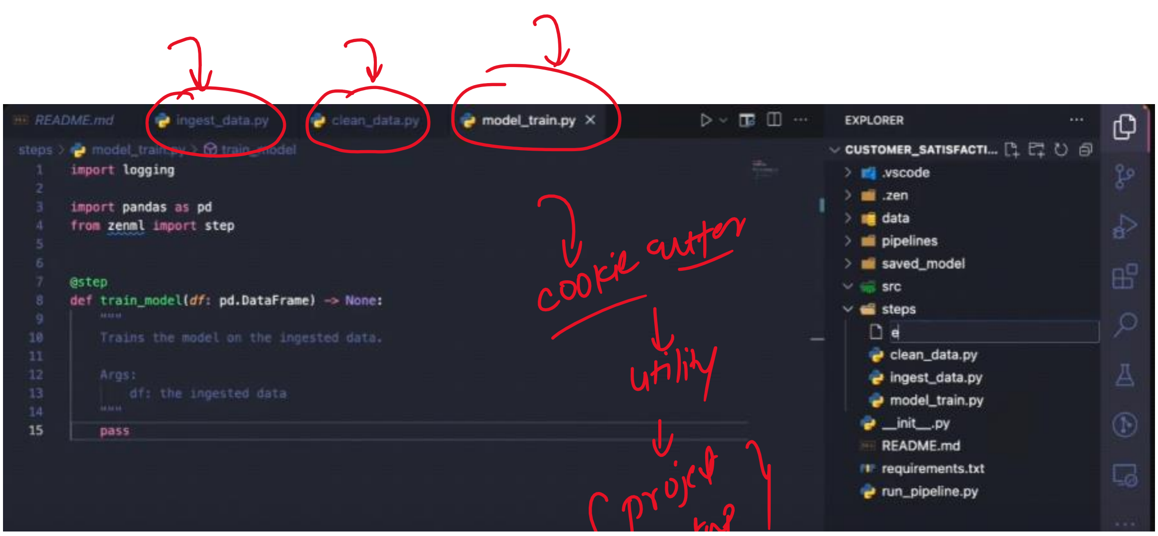
DW

g code

1. Difficulty in Maintenance: As the complexity of the code increases, it becomes harder to maintain. Changes in one part of the code can unexpectedly affect other parts due to the tightly coupled nature of the implementation. This can lead to increased time spent debugging and verifying that changes do not break existing functionality.

code

2. Limited Reusability: In a non-modular setup, code reusability is minimal. Functions and components are often written to solve a specific problem and are tightly integrated with other parts of the code, making them difficult to extract and reuse in other projects or contexts.
3. Harder Collaboration: When code is not modular, it's more challenging for multiple developers to work on the same project simultaneously. Since everything is interconnected, developers need to be more cautious about the changes they make, which can slow down development and increase the risk of merge conflicts.
4. Poor Scalability: Scaling a non-modular application can be problematic. As more features and functionalities are added, the codebase can become unwieldy and difficult to manage. This lack of scalability extends not only to the size of the codebase but also to the handling of larger data sets or more complex models.
5. Testing Challenges: Testing a non-modular codebase is typically more challenging and less effective. Without clear boundaries between components, it's tough to perform unit testing; thus, most testing becomes integration testing, which is less granular and can overlook specific defects.
6. Inefficiency in Iteration and Experimentation: Machine learning projects often require iterative adjustments and experiments with different models, parameters, and data preprocessing methods. A non-modular approach complicates these experiments, as each change might require alterations across multiple sections of the code, increasing the risk of errors and the time needed for each iteration.
7. Difficulty in Tracking Changes and Versioning: In a non-modular architecture, it's challenging to implement effective version control practices. Changes to the codebase can affect many parts of the application, making it harder to track changes, roll back updates, or manage different versions of the code for different experiments or deployment scenarios.
8. Integration Problems: Integrating the system with other services or pipelines (like data ingestion, real-time data processing, etc.) can be cumbersome without clear module boundaries. This lack of separation can lead to issues when connecting different parts of a project or ensuring that they interact smoothly.



good development practices

MLOps

Problem 3 - The versioning problem

09 May 2024 09:46

helps in collaboration

Score predictor → git / github

versioning

collaborate

versioning → rollback

software

code

ml project

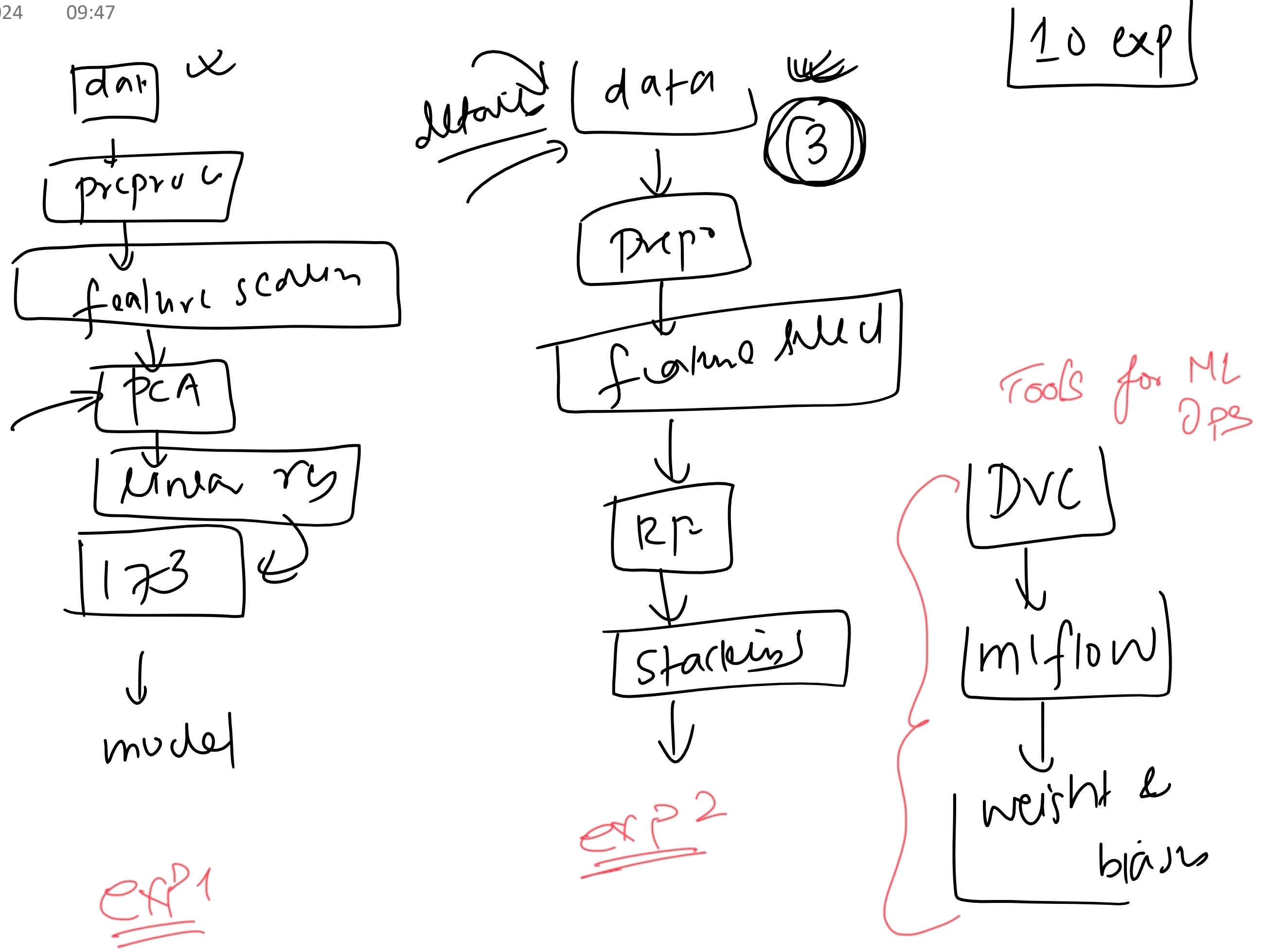
code + data + model

DVC

Problem 4 - The Experimentation Problem

09 May 2024 09:47

Experiment Tracking



Problem 5 - Automation

09 May 2024 21:54

Problem 6 - The Deployment Problem

09 May 2024 09:47

mlops

automate

Devops

ML

model.pkl → S3 AWS

api

backend code

frontend

tooling

CI | CD.

CI

1 codebase

2 prediction

3 api

containerization

Docker / kubernetes

magic mode.

model

prediction

local machine

CI

Problem 7 - The Drift problem

09 May 2024 09:47

Impact Player

model tracking

model deploy monitoring

grafana / Prometheus

Retraining

1. grafana
2. Prometheus
3. Continuous training

Problem 8 - The Infrastructure Problem

09 May 2024 09:48

GPU

DB

Experimentation.

Problem 9 - The Collaboration Problem

09 May 2024 09:48

Data engineer
Data scientist
ML engineer
Frontend
Backend

engineering manager

project manager

Problem 10 - The Legal Problem

09 May 2024 09:49

espn predict $\boxed{\pm 5}$ rmse

ethics

Probabi

chatgpt → $\boxed{\text{data}}$

ethics → biases

Aspects of MLOps

08 May 2024 14:09

ml dev / ml ops

1. Data Management ml dev
    a. Data Collection
    b. Data Preprocessing
    c. Data Validation
    d. Data Security
    e. Data Compliance
    f. Feature Store
2. Development Practices dev
    a. Modular Coding
3. Version Control dev
    a. Code versioning
    b. Data versioning
    c. Model versioning
4. Experiment Tracking dev
    a. Tracking ml experiments
    b. Test and validation
    c. Model registry
5. Model Serving and CI/CD Ops
    a. Continuous Integration
    b. Containerization
    c. Continuous Deployment
6. Automation ops
    a. Pipeline automation [Data ingestion pipeline, model training pipeline, model validation and testing, model deployment, model monitoring and retraining]
    b. Orchestration

    Airflow

    Airflow Kubeflow

    DAG
7. Monitoring and Retraining dev

    ops
    a. Model Monitoring
    b. Drift Detection
    c. Retraining
8. Infrastructure Management Ops
    a. Cloud based solutions to handle scalability concerns
    b. Cost management
    c. Managing multiple vendors
9. Collaboration and Operations ops

c. Managing multiple vendors

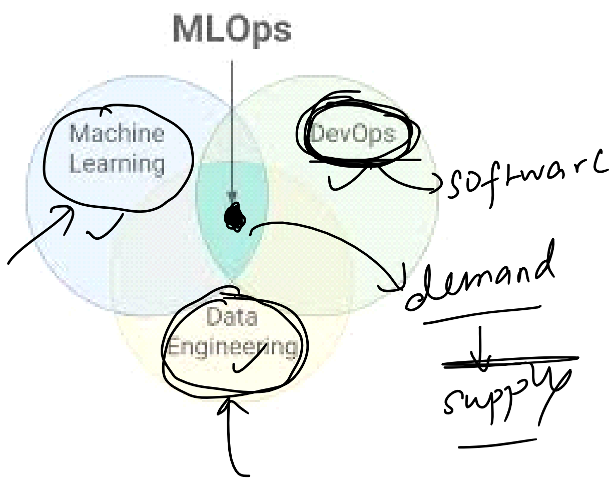
9. Collaboration and Operations Ops
   a. Unified workspace
   b. Role based access

10. Governance and Ethics Ops

What is MLOPs [looking back]

08 May 2024 14:22

MLOps refers to the practice and discipline within machine learning that aims to unify and streamline the machine learning system development (Dev) and machine learning system operation (Ops). It involves collaboration between data scientists, ML engineers, and IT professionals to automate and optimize the end-to-end lifecycle of machine learning applications.



MLOps Maturity Levels

08 May 2024 14:19

Benefit of MLOps

08 May 2024 14:19

1. Scalability
2. Improved performance
3. Reproducibility
4. Collaboration and efficiency
5. Risk reduction
6. Cost Savings
7. Faster time to market
8. Better compliance and governance

1. Data Management
2. Development Practices
3. Version Control
4. Experiment Tracking/Model Registry
5. Model Serving and CI/CD
6. Automation
7. Monitoring and Retraining
8. Infrastructure Management
9. Collaboration and Operations
10. Governance and Ethics

Challenges

08 May 2024 14:19

1. Complexity of ml models [variability, black box nature]
2. Quality of data
3. Cost and resource constraints
4. Handling scale
5. Security risks
6. Compliance and regulatory concerns
7. Integration with existing systems
8. Limited Expertise/Skill gap

Prerequisites to become a MLOps Engineer

08 May 2024 14:20

1. Basic understanding of ML
    a. Cleaning and preprocessing
    b. Feature engineering
    c. Model building
2. Software development skills
    a. Python
    b. Git
    c. Software development best practices [OOP, Design Patterns]
3. Data Engineering
    a. SQL
    b. Big Data Tech [Spark, Kafka]
    c. Data Storage Solutions [Databases, Data Warehouses, Data lakes]
4. DevOps Principles and Tools
    a. CI/CD Pipeline
    b. Automation
5. Familiarity with cloud platforms
    a. AWS, GCP and Azure
6. Containerization technologies
    a. Docker
    b. Kubernetes
7. Networking Principles
    a. Distributed computing
8. Security Fundamentals
    a. Cybersecurity fundamentals
9. Soft Skills