Data Engineering for MLOps

Building Efficient Data Workflows for Machine Learning

Lecturer: Vangelis Oden - Technology Lead (Kera)
Assistant: Natalija Mitic - AI/ML Engineer (Kera)

Agenda

- Why it Matters
- Data Collection, Preprocessing, and Feature Engineering
- Data Exploration
- Distributed Data Processing
- Data Pipelines and ETL Processes
- Data Versioning and Lineage
- Q&A

Why it matters?

Intuition:

ML Models are only as good as the data they are built on.
DE ensures that data flows are reliable, clean, and scalable across the entire ML lifecycle.

- data quality : Poor quality data leads to poor model performance.
- scalability : As data grows, proper pipelines and processing systems ensure scalability.
- reproducibility : Versioning and automation are critical to reproducing results and maintaining long-term models.

Data Collection, Preprocessing, and Feature Engineering

python

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load dataset
data = pd.read_csv('data.csv')

# Handle missing values
data.fillna(method='ffill', inplace=True)

# Feature scaling
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data[['feature1', 'feature2']])
```

Data Collection, Preprocessing, and Feature Engineering

Intuition:

data collection : gathering diverse and relevant data is the foundation of most ML models - databases, APIs, etc. JSON, CSV, JPEGS, etc.
preprocessing : ensuring the data is clean and prepared allows models to learn from accurate, well-structured inputs. Split data!!!
feature engineering : crafting the right features can transform raw data into highly informative inputs for your ML model

Data Exploration

Intuition:

Explores several other preprocessing and feature engineering steps that need to happen.

- Provides a general understanding of the kind of data you are working with
- Helps validate need for hypothesis
- Makes it easier to make decisions

Distributed Data Processing

python

Copy

```python
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder.appName("DataProcessing").getOrCreate()

# Load large dataset in distributed mode
df = spark.read.csv('large_dataset.csv', header=True, inferSchema=True)

# Perform transformations
df_transformed = df.filter(df['age'] > 18).select('name', 'age')

# Save result
df_transformed.write.csv('processed_data.csv')
```

Distributed Data Processing

Intuition:

- Allows us to handle large-scale datasets by splitting tasks across multiple machines
- Helps reduce processing time and computational load for both batch and real-time processing.
- Enables scaling from small datasets to massive, real-world datasets
- **tools** **:** Apache Spark, Dask, Modin, Hadoop

Data Pipelines and ETL Processes

python

Copy

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

def extract():
    # Extract data
    return pd.read_csv('raw_data.csv')

def transform(data):
    # Simple data transformation
    return data.dropna()

def load(data):
    # Load transformed data
    data.to_csv('processed_data.csv')

# Define the Airflow DAG
dag = DAG('etl_pipeline', start_date=datetime(2023, 1, 1))

extract_task = PythonOperator(task_id='extract', python_callable=extract, dag=dag)
transform_task = PythonOperator(task_id='transform', python_callable=transform, dag=dag)
load_task = PythonOperator(task_id='load', python_callable=load, dag=dag)

extract_task >> transform_task >> load_task
```

Data Pipelines and ETL Processes

Intuition:

These are automated processes that move data from one stage to another, ensuring clean, processed and prepared data is always available for ML models.

- etl : automates data collection, transformation, storage for ML models, ensuring consistency and repeatability.
- scalability : pipelines allow for handling larger datasets and more complex workflows without human intervention.

Data Versioning and Lineage

```bash
# Initialize DVC in a project
dvc init

# Track dataset with DVC
dvc add data/raw_data.csv

# Save changes with Git
git add data/raw_data.csv.dvc .gitignore
git commit -m "Added raw dataset"

# Push data to remote storage
dvc remote add -d myremote s3://bucket-name/path
dvc push
```

Data Versioning and Lineage

Intuition:

Ensures every version of data, code, and model is trackable, making models reproducible. Allows you trace data flow from collection to usage in a model.

- reproducibility : ability to recreate models from exact data
- auditability : ensures data accuracy and debugging
- **tools** : DVC, LakeFS, Apache Atlas, etc.

Best Practices for Data Engineering in MLOps

- **automate data pipelines** : use tools like Apache Airflow for automation
- **monitor data quality** : implement checks for missing values, duplicates, outliers, and other details that may reduce the quality of your data
- **version everything** : track versions of data, code and models.
- **scalability** : design with scalability in mind to handle future growth
- **use distributed** : where available, leverage accelerated processing

Conclusion

- Data engineering plays a critical role in ensuring that machine learning models are trained on clean, high-quality data.
- Distributed data processing allows scaling, while versioning and lineage ensure reproducibility.
- Use tools to improve your data engineering workflows.

Q&A

Thank You!