To prepare for the second interview round, we would like you to have a look at the following web page: MLflow - A platform for the machine learning lifecycle | MLflow.

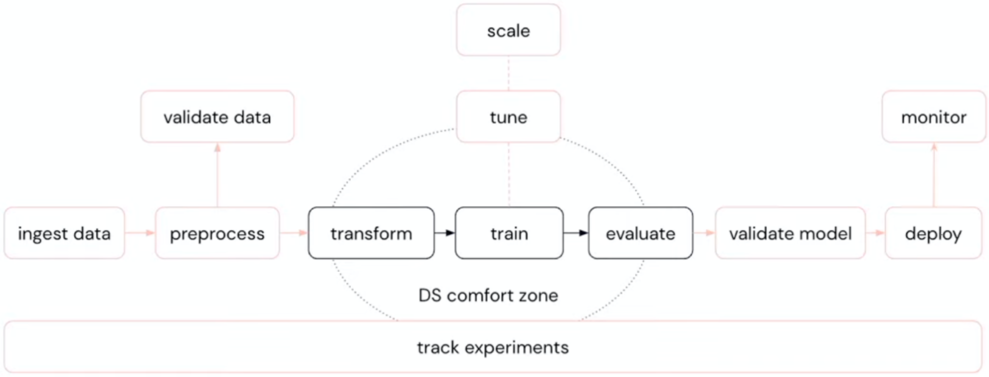
We will then discuss how you propose to implement a workflow to train and deploy an object detection model (for example something from the TensorFlow object detection API - TensorFlow 2 Object Detection API tutorial — TensorFlow 2 Object Detection API tutorial documentation (tensorflow-object-detection-api-tutorial.readthedocs.io)).

The task is to get an understanding of the concepts. We do not expect you to implement anything before the interview.

The questions to ask before the interview:

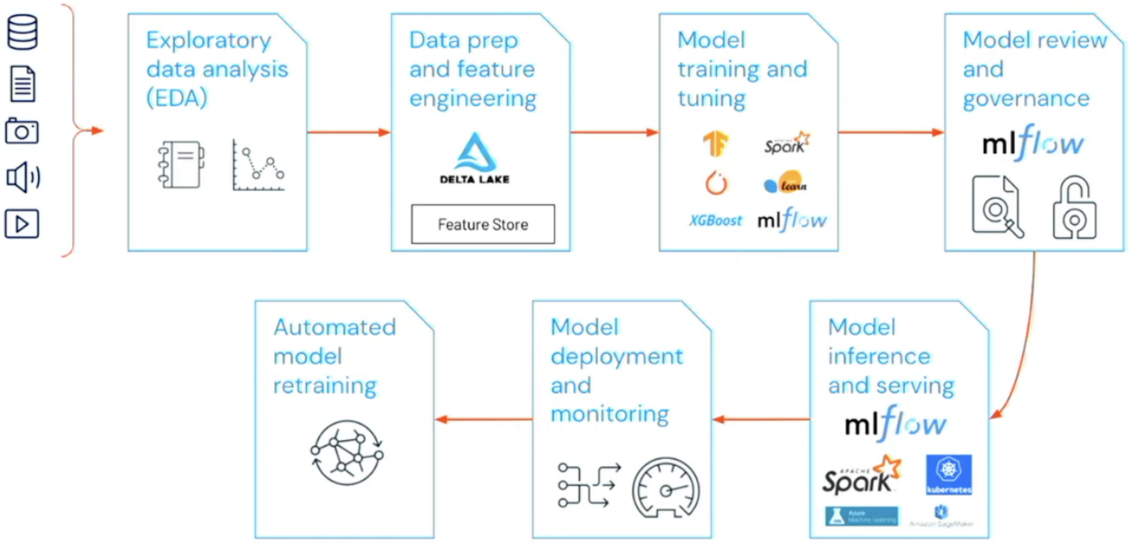
1. Does the team have existing labeled and annotated data?
2. Does the team have **mlflow** tracking sever either self-hosted or managed service from a cloud provider like Databrick?
3. Where is the desired location for model-related artifacts to store? Self-hosted databases or storage buckets from cloud storage service?
4. What is the current code environment look like? Self-hosted source code versions control service like GitLab or public git repo or store personal laptop?

Machine Learning Life Cycle



Details Workflow article:
https://www.linkedin.com/pulse/building-workflow-object-detection-model-training-using-hlaing

The machine learning life-cycle



Exploratory data analysis (EDA)

Data prep and feature engineering

Feature Store

Model training and tuning

Model review and governance

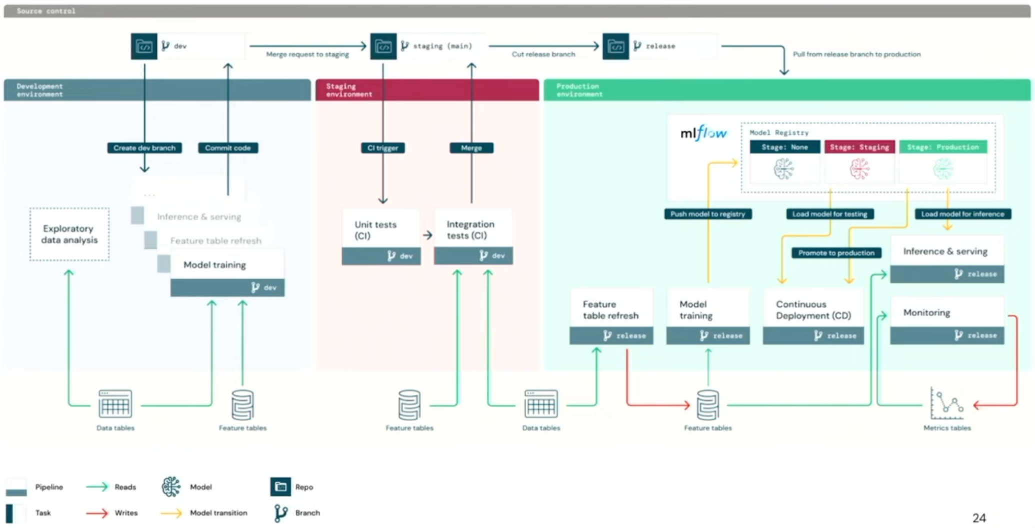
Model inference and serving

Model deployment and monitoring

Automated model retraining

Source: databricks.com

Overview



Source control

dev

Merge request to staging

staging (main)

Cut release branch

release

Pull from release branch to production

Development environment

Create dev branch

Commit code

Exploratory data analysis

...

Inference & serving

Feature table refresh

Model training

dev

Staging environment

CI trigger

Merge

Unit tests (CI)

dev

Integration tests (CI)

dev

Production environment

Model Registry

Stage: None

Stage: Staging

Stage: Production

Push model to registry

Load model for testing

Load model for inference

Promote to production

Inference & serving

release

Feature table refresh

release

Model training

release

Continuous Deployment (CD)

release

Monitoring

release

Pipeline

Task

Reads

Writes

Model

Model transition

Repo

Branch

Logging and registering to mlflow tracking server

```
# Set the tracking server uri and create expriemnt
mlflow.set_tracking_uri(uri)
mlflow.create_experiment(experiment_name)
mlflow.set_experiment(experiment_name)

# log the TensorFlow model training with autolog
mlflow.tensorflow.autolog(every_n_iter=1)

# log the parameters and metrics
mlflow.log_params(
       {"max_depth": max_depth,
        "max_leaf_nodes": max_leaf_nodes}
   )
mlflow.log_metric("test_accuracy", test_score)

# log the model and artifact
mlflow.sklearn.log_model(tree, "model",
signature=signature)   # <-- Now log the model with a
signature
mlflow.register_model(run_uri, model_name)

# log text and other artifact
mlflow.log_artifact("1_Run and track experiments.ipynb")
mlflow.log_text("Here you can add general inforamtion
about the run","run_info.txt")
```

Deployment from mlflow registered model with docker image

```
mlflow models serve --no-conda -m "models:/penguins_clf/1" -p 4242
```

```
# record-oriented DataFrame input (fine for vector rows, loses ordering for JSON records)
curl http://127.0.0.1:4242/invocations -H 'Content-Type: application/json; format=pandas-records' -d '[
   {"Culmen Length (mm)": 1,"Culmen Depth (mm)": 3},
   {"Culmen Length (mm)": 14,"Culmen Depth (mm)": 120}
]'
```

Deployment from mlflow registered model with docker image

```
mlflow models build-docker --name penguins_clf \
                           --model-uri "models:/penguins_clf/1"

docker run -p 5000:8080 "penguins_clf"
```