**Process and Tooling**

1. **Experiment Tracking**

We have integrated our training with MLFlow for experiment tracking. We have detailed our grid search program for finetuning in the M3 report. Once the best hyperparameters are found and the model is built, the model is pushed to MLFlow with the below code. We are storing the information below in the mlflow tracking.

* The best params found
* The model itself
* The evaluation metrics associated

#### **Model training with MlFlow integration code snippet**

A screenshot of a computer code

Description automatically generated

We exposed the training as an API for on-demand training. The API responds with the best parameters and the evaluation metrics also.

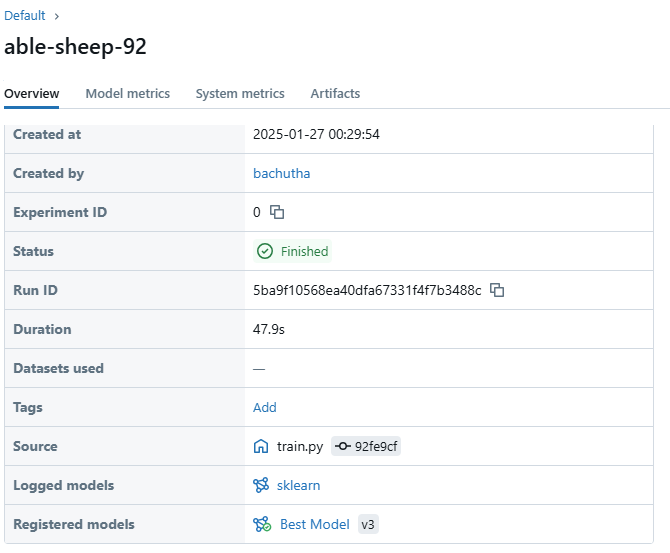
**MLflow Tracking**:

MLflow Tracking is a feature of MLflow designed to log and query experiments, helping users systematically track parameters, metrics, artifacts, and models across multiple runs. It enables the organization of experiments into named groups called "experiments," which can contain individual runs representing specific executions. Users can log information programmatically via MLflow APIs or automatically through autologging integrations with popular ML frameworks. The tracking server can be local or remote, allowing centralized storage and collaboration for teams. With its user-friendly web UI, MLflow Tracking simplifies the process of comparing runs, visualizing metrics, and maintaining reproducibility in machine learning workflows

A screenshot of a computer

Description automatically generated

Tracked best parameters



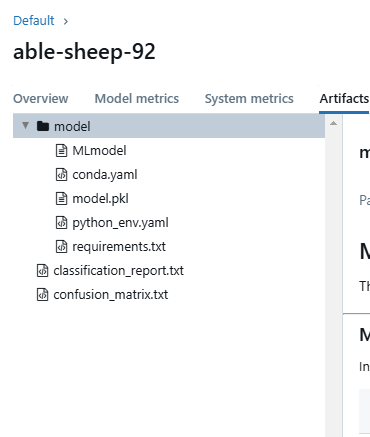
A screenshot of a computer

Description automatically generated

A blue bar graph with white text

Description automatically generated

**MLflow Models** :  
MLflow Models is a component of MLflow that standardizes the packaging, deployment, and management of machine learning models. Models are stored in a versatile format (MLmodel file) that includes metadata about the model, its dependencies, and its flavor (e.g., TensorFlow, PyTorch, Scikit-learn). It supports deployment across multiple environments, such as local servers, cloud platforms, or Docker containers, ensuring flexibility. MLflow Models integrates seamlessly with serving tools, enabling APIs for real-time predictions. This standardization simplifies model portability, reproducibility, and collaboration between development and production teams. We used to store the model as pkl(pickle file) extension.



1. **Data Versioning**

Data Version Control (DVC) is an open-source tool that enables versioning and management of datasets, machine learning models, and pipelines. It is designed to address the challenges of tracking data and model changes while ensuring reproducibility and scalability in data science projects. DVC extends the functionality of Git by providing support for handling large datasets and binary files that cannot be stored efficiently within Git repositories.

We used DVC versioning and ML pipeline building

A black screen with white text

Description automatically generated

The cached reference file will be submitted to the git , but it can be stored in a remote location like Azure BLOB, AWS S3 bucket etc. The files will be referenced using the hash name present in the dvc extension file.

A screenshot of a computer program

Description automatically generated

DVC is used for to create ML pipelines the workflow of preprocessing, training, evaluation, and deployment steps required to build a machine learning model.

This helped

* Ensures consistency across experiments.
* Makes the ML workflow modular and reusable.
* Enables reproducibility and collaboration

A screen shot of a computer

Description automatically generated