8. Data Versioning and Lineage

8.1 Versioning Approach

Data versioning and lineage tracking are implemented using **DVC (Data Version Control)**. DVC enables version control of large datasets without storing the actual data files in the Git repository.

This approach ensures:

- Reproducibility of experiments
- Clear tracking of dataset evolution
- Separation of code and data versioning

Both raw and processed datasets are versioned to maintain end-to-end data lineage.

8.2 Datasets Under Version Control

The following data layers are tracked using DVC:

Raw Data

`data/raw/`

This includes:

- Clickstream interaction data
- Product metadata ingested from the REST API

Processed Data

`data/processed/`

This includes:

- Cleaned and prepared interaction datasets
- EDA artifacts such as plots and summaries

Tracking both layers ensures that transformations can always be traced back to the original data source.

8.3 DVC Initialization

DVC is initialized at the repository level using:

```
dvc init
```

This creates the required DVC configuration files:

```
.dvc/
.dvcignore
```

These files are committed to Git to enable data versioning across the project.

8.4 Dataset Versioning Workflow

Raw and processed datasets are added to DVC tracking using:

```
dvc add data/raw
dvc add data/processed
```

This generates lightweight metadata files:

```
data/raw.dvc
data/processed.dvc
```

Only these `.dvc` files are committed to Git, while the actual data remains stored locally.

8.5 Repository Structure After Versioning

After enabling DVC, the repository structure includes:

```
DM4ML-Assignment-1/
├── data/
│   ├── raw/                # Tracked by DVC
│   ├── processed/          # Tracked by DVC
│   ├── raw.dvc
│   └── processed.dvc
├── .dvc/
├── .dvcignore
├── src/
├── config/
└── reports/
```

This structure clearly separates code, configuration, and versioned data artifacts.

8.6 Data Lineage Tracking

Data lineage is established through:

- Hierarchical data layers (raw → processed → features)
- Timestamp-based ingestion directories
- DVC metadata files that record dataset hashes and versions
- Git versioning of transformation scripts

This provides a clear mapping between:

data source → transformation logic → resulting datasets

8.7 Reproducibility Workflow

A specific version of the dataset can be restored using:

```
git checkout <commit-hash>
dvc checkout
```

This restores the exact versions of raw and processed datasets associated with that commit, ensuring reproducible data pipelines and experiments.

8.8 Summary

The use of DVC provides a reliable mechanism for data versioning and lineage tracking across the data management pipeline. It ensures that all datasets used for feature engineering and model training are traceable, reproducible, and aligned with their corresponding transformation logic.