Supplementary AI-Usage Report

Tej Deepak Amin *Matr. No. 264777* | Shamita Nalamutt *Matr. No. 264985* | Prabhav Patel *Matr. No. 269589* | Anvita Koyande *Matr. No. 265122*

1. AI Model(s) Employed

- **OpenAI GPT-4 Turbo** (via ChatGPT)
- **GitHub Copilot** (VS Code extension)

2. AI-Assisted Components

Component	AI Role
Scaffolding & I/O	Generated imports, directory traversal, file I/O templates.
Filtering & Visualization	IQR/median outlier functions and plotting templates.
Clustering & Splitting	KMeans/PCA setup; train/val/calib/test split logic.
Model Definitions	NARXNetReg & QuantileNet skeletons with loss functions.
Training & Checkpointing	PyTorch loops for weighted MSE, quantile training, and saves.

3. Nature of the Assistance

- **Code Generation & Completion**
  - Function/class templates for loops, filtering, clustering, models.
  - Suggested hyperparams, dropout, DataLoader setup.
- **Debugging & Optimization**
  - Handled silhouette exceptions; advised vectorized rolling windows.
- **Iterative Refinement**
  - Added rolling-median smoothing to IQR filters.
  - Flattened multi-index names; standardized checkpoints.
- **Advanced Modules Support**
  - Built PinballLoss & conformal -level code.
  - Drafted =0.05/0.95 quantile loops & interval logic.
- **Technical Writing & Documentation**
  - Generated docstrings (`read_txt`, `split_files`, `rolling_iqr_filter`).
  - Framed log messages (`[INFO]`, `[WARN]`).

4. Illustrative Iteration (NARX Training Loop)

1. **Prompt:** "Define a PyTorch `NARXNetReg` with three hidden layers, dropout, and weighted MSE…"
2. **AI's Initial Draft:**

```
class NARXNetReg(nn.Module):
    def __init__(self,d_in,d_out):
        super().__init__()
        self.net=nn.Sequential(...)
    def forward(self,x): return self.net(x)
```

3. **Members' Feedback:** "Add `weighted_mse`, integrate `weights`, log best MSE."
4. **AI's Revised Snippet:**

```
def weighted_mse(y_pred,y_true):
    mse=(y_pred-y_true)**2
    return (mse*weights).mean()
```

5. **Final Integration:** Logging rewritten; device-aware casting; `torch.save`.

5. Authorship & Effort Attribution

- **Core Logic & Expertise (85%)**: Members authored transforms, clustering, conformal derivations.
- **AI Contributions (15%)**: Scaffolding only; snippets reviewed & refactored.
- **Validation & Testing**: Unit tests, benchmarks, edge-case checks by members.
- **Documentation & Style**: Members rewrote AI comments & docstrings.

Conclusion: AI sped up scaffolding and debugging hints; MLME's design, tuning, & validation were driven by project members.