Scalability & Cloud Simulation Document

Activity Forecasting – Daily Steps Prediction

---

1. Objective

The objective of this section is to describe how the current single-patient activity forecasting pipeline can be **scaled to support large-scale deployment**, such as **100,000+ patients**, while maintaining performance, reliability, and data security.

---

2. Current System Overview

The current implementation:

- Processes activity and clinical data for a **single patient**
- Performs daily aggregation and feature engineering locally
- Trains forecasting and explainable ML models in a notebook environment
- Outputs a 365-day forecast as a CSV file

While suitable for prototyping and experimentation, this setup requires architectural changes for large-scale usage.

---

3. Scalable Architecture Design

3.1 Data Ingestion Layer

- Raw activity data and clinical data can be ingested using:
  - Cloud object storage (Amazon S3 / Google Cloud Storage)
  - Streaming pipelines (Kafka / AWS Kinesis) for real-time data
- Data would be partitioned by:
  - patient_id
  - date

This enables parallel processing across patients.

3.2 Distributed Data Processing

- Daily aggregation and feature engineering can be implemented using:
  - Apache Spark / Spark SQL
  - Cloud-based ETL services (AWS Glue / Dataflow)

Each patient's data can be processed independently, allowing horizontal scaling.

---

4. Model Training & Inference at Scale

4.1 Forecasting Models

- The baseline forecasting model (Prophet or similar) can be:
  - Trained per patient
  - Or trained as a global model with patient-specific adjustments
- Batch forecasting jobs can generate future predictions periodically (e.g., daily or weekly).

4.2 Explainable ML Models

- Multivariate models such as EBM can be trained using:
  - Shared global structure
  - Patient-level feature inputs
- Explainability outputs can be generated on demand for auditing and clinical interpretation.

---

5. Cloud Storage & Output Management

Forecast outputs (365-day predictions) can be stored in:

- Cloud object storage (S3 buckets)
- Partitioned by patient ID and forecast date

Example simulated function used in the notebook:

```
def upload_forecast_to_s3(dataframe, bucket_name, file_name):
    import boto3
    s3 = boto3.client('s3')
    # s3.put_object(Bucket=bucket_name, Key=file_name, Body=dataframe.to_csv(index=False))
```

(The actual upload call is commented out for safety.)

---

6. Security & Credential Management

To ensure secure deployment:

- **IAM roles** should be used instead of hard-coded credentials
- Environment variables or secret managers should store access keys
- Notebook environments should never expose AWS keys directly

This aligns with cloud security best practices.

7. Monitoring & Maintenance

At scale, the system would include:

- Automated data validation checks
- Model performance monitoring (RMSE/MAE drift)
- Scheduled retraining pipelines
- Logging and alerting for failed jobs

---

8. Conclusion

The proposed cloud-based architecture enables the current activity forecasting pipeline to scale efficiently from a single patient to **hundreds of thousands of patients**. By leveraging distributed data processing, secure cloud storage, and modular model deployment, the system can support real-world healthcare analytics while maintaining transparency and explainability.