Project Documentation

Telecom Churn Prediction System

Software Engineering Project

---

1. Title

Telecom Churn Prediction System

2. Objective

To design and implement a modular software system capable of predicting customer churn in the telecom sector using a complete ML workflow — from data ingestion to web-based deployment.

3. Scope

The system automates the entire ML lifecycle, enabling users to preprocess data, train models, and deploy a prediction interface for real-time use.

4. System Requirements

Software:

- Python 3.10
- Flask Framework
- scikit-learn, pandas, numpy, joblib, PyYAML

Hardware:

- Minimum 4GB RAM
- Dual-core processor
- 1GB free disk space

5. Project Modules

1. **Data Ingestion** – Reads and splits dataset into training/testing subsets.
2. **Data Validation** – Validates data schema and structure.
3. **Data Transformation** – Performs feature engineering and preprocessing.
4. **Model Trainer** – Trains the machine learning model.
5. **Model Evaluation** – Evaluates model accuracy and performance.
6. **Pipeline Execution** – Manages sequential workflow.
7. **Web Application Interface** – Flask-based interface for training and prediction.

6. System Design

Architecture Layers:

- **Data Layer** – Ingestion, validation, and transformation.
- **Model Layer** – Model training, evaluation, and persistence.
- **Application Layer** – User interaction via Flask web interface.

7. Configuration Files

- **config.yaml** – File paths and configuration parameters.
- **params.yaml** – Model hyperparameters and tuning parameters.
- **schema.yaml** – Schema validation for input dataset.

8. Project Structure

```
.
 artifacts/
 config/
    config.yaml
 data/
    telecom_churn.csv
 logs/
    logging.log
 src/
    telecom_churn/
```

```
        components/
        config/
        constant/
        entity/
        pipeline/
        utility/
        __init__.py
static/
   style.css
templates/
   index.html
   result.html
app.py
main.py
params.yaml
schema.yaml
```

9. Implementation Steps

1. Update configuration files.
2. Modify entities and configuration manager.
3. Implement pipelines and components.
4. Run `main.py` to start training.
5. Launch `app.py` for web interface.

10. Testing Overview

Testing involved module-wise and integrated testing to ensure correctness and performance of data flow, model pipeline, and web app. All test cases were validated successfully.

11. Results

- Model trained successfully on telecom churn dataset.
- Predictions were accurate and displayed via the Flask interface.
- Logs and artifacts generated automatically for traceability.

12. Conclusion

The Telecom Churn Prediction System was successfully designed, implemented, and tested following modular software engineering principles. The system ensures maintainability, reusability, and real-time usability through its web interface.