Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **Regulix - Ridge & Lasso Regression Tuner**. Regulix is a FastAPI-based web tool enabling users to upload datasets, perform feature scaling, tune Ridge and Lasso regression models using GridSearchCV, visualize regularization effects, and evaluate model performance.

---

1. System Components

Component	Description	Key Responsibilities
API Service	FastAPI microservice	Expose endpoints for upload, tuning, results
Data Processor	Handles dataset parsing and feature scaling	Clean, validate, scale data
Model Trainer	Manages model selection, training, and hyperparameter tuning	Fit Ridge/Lasso, run GridSearchCV
Visualizer	Generates plots for regularization and metrics	Create and serve matplotlib plots
Evaluation Module	Computes model metrics	Calculate RMSE, MAE, R^2, etc.

---

2. Class/Interface Overview

Class/Interface	Key Methods/Attributes	Relationships
`DatasetHandler`	`load_csv()`, `scale_features()`, `get_features_labels()`	Used by ModelTrainer
`ModelTrainer`	`train_ridge()`, `train_lasso()`, `grid_search()`	Uses DatasetHandler, EvaluationModule
`EvaluationModule`	`compute_metrics()`	Used by ModelTrainer
`Visualizer`	`plot_coefficients()`, `plot_metrics()`	Used by API Service
`APIRouter`	`upload_endpoint()`, `train_endpoint()`, `results_endpoint()`	Orchestrates all components

Key Methods Example:

```
class ModelTrainer:
    def train_ridge(self, X, y, params): ...
    def train_lasso(self, X, y, params): ...
    def grid_search(self, model, param_grid, X, y): ...
```

3. Data Structure Overview

Data Model	Fields/Schema	Notes
Dataset	`features: np.ndarray`, `labels: np.ndarray`	Loaded from user-uploaded CSV
ModelConfig	`model_type: str`, `param_grid: dict`	Specifies Ridge/Lasso and hyperparameters
MetricsResult	`rmse: float`, `mae: float`, `r2: float`	Evaluation output
PlotData	`coef_path: list`, `metric_path: list`	For visualization

4. Algorithms/Logic

Model Training & Tuning Flow:

```
def train_and_evaluate(data, model_type, param_grid):
    X_scaled = scale_features(data.features)
    model = Ridge() if model_type == 'ridge' else Lasso()
    grid = GridSearchCV(model, param_grid)
    grid.fit(X_scaled, data.labels)
    metrics = compute_metrics(grid.best_estimator_, X_scaled, data.labels)
    plots = generate_plots(grid)
    return metrics, plots
```

5. Error Handling

Scenario	Handling Approach
Invalid file format/upload	Return 400 with error message
Data parsing/scaling errors	Log error, return 422 with details
Model convergence failure	Return 500 with diagnostic info
Hyperparameter grid issues	Validate input, return 400 if invalid
Visualization generation errors	Return 500, log stack trace

End of Document