Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **StackVote - Ensemble Learning Playground**. StackVote is a Streamlit-based application enabling users to experiment with stacking, blending, and voting ensemble classifiers. Users can upload datasets, select and configure base models, and visualize ensemble versus individual model performance.

---

1. System Components

Component	Description	Key Responsibilities
UI Layer	Streamlit-based frontend	User interaction, data upload, parameter input
Data Manager	Handles dataset loading and preprocessing	Data validation, cleaning, splitting
Model Manager	Manages model selection, training, and evaluation	Model instantiation, fitting, scoring
Ensemble Engine	Implements stacking, blending, voting logic	Ensemble construction, prediction
Visualization Layer	Generates performance plots and metrics	Display charts, tables, comparison metrics

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`DataManager`	Handles dataset operations	`load_data()`, `preprocess()`, `split()`
`ModelManager`	Manages ML models	`get_models()`, `train()`, `evaluate()`
`EnsembleEngine`	Builds and applies ensembles	`build_ensemble()`, `predict()`
`Visualizer`	Handles result visualization	`plot_metrics()`, `plot_comparison()`

Relationships:

- `UI Layer` interacts with all managers.
- `ModelManager` and `EnsembleEngine` depend on `DataManager` for data.
- `Visualizer` consumes outputs from `ModelManager` and `EnsembleEngine`.

---

3. Data Structure Overview

Data Model	Fields/Schema	Description
Dataset	`X` (features, DataFrame), `y` (labels, Series)	User-uploaded or sample data

ModelConfig	`model_type`, `params`	Selected model and hyperparameters
EnsembleConfig	`ensemble_type`, `base_models`, `meta_model`	Ensemble method and components
Metrics	`accuracy`, `f1`, `roc_auc`, etc.	Evaluation results

---

4. Algorithms/Logic

Main Flow Pseudocode:

```
# User uploads data and selects models/parameters
data = DataManager.load_data(uploaded_file)
X_train, X_test, y_train, y_test = DataManager.split(data)

base_models = ModelManager.get_models(selected_models, params)
ModelManager.train(base_models, X_train, y_train)

ensemble = EnsembleEngine.build_ensemble(ensemble_type, base_models, meta_model)
ensemble.fit(X_train, y_train)

results = {
    "base": ModelManager.evaluate(base_models, X_test, y_test),
    "ensemble": ensemble.score(X_test, y_test)
}

Visualizer.plot_comparison(results)
```

---

5. Error Handling

Scenario	Handling Approach
Invalid file format/upload	Show error message, prompt re-upload
Data preprocessing failure	Display error, log details, halt further steps
Model training errors	Catch exception, show user-friendly message
Ensemble config mismatch	Validate config, prompt user to correct selection
Visualization errors	Fallback to text output, log error

End of Document