Nagini ML Wizard – Project Documentation

# 1. Project Overview

Nagini ML Wizard is an interactive machine learning helper built with Python and Gradio, designed to guide users step-by-step through the ML workflow: from data upload and preprocessing, to exploratory data analysis (EDA), model training, evaluation, and prediction.

The goal is to provide a wizard-like experience (inspired by Windows 98 wizards), so that beginners and learners can experiment with different models, see metrics and plots, and understand the impact of their choices.

# 2. Components

The project is structured into several key parts:

## 2.1 app.py

- The main entry point of the wizard.  
- Defines the Gradio interface with tabs: Data, Preprocess & EDA, Train, Predict.  
- Manages user interactions and connects them to the underlying logic in the wizard/ package.

## 2.2 wizard/

This folder holds the reusable logic and pipeline steps:

- data\_io.py: Handles dataset loading (CSV upload or built-in samples like Iris, Wine, Diabetes, and Wine Quality). Also includes helper functions for type inference.  
- preprocess.py: Functions for missing-value handling, scaling, encoding, and row dropping.  
- eda.py: Functions for generating plots such as target distribution, correlation, and feature-vs-target plots.  
- state.py: Defines WizardState to keep track of dataset, target, features, preprocessing choices, and trained model.  
- models/: Contains registry of supported models (Decision Tree, Random Forest, MLP, SVM, Logistic/Linear Regression, Naive Bayes, KNN) and their training/evaluation utilities.

## 2.3 Individual Demo Files (e.g., app\_DT.py, app\_RF.py, ...)

- These were earlier standalone demos for each model (Decision Tree, Random Forest, etc.).  
- Each file defines its own Gradio UI, training loop, and plots for one algorithm.  
- They inspired and provided base logic for the unified Nagini ML Wizard.

# 3. Current Progress

As of now, the following steps have been implemented:

- Data upload: CSV or sample dataset loading (Iris, Wine, Diabetes, Wine Quality).  
- Preprocessing choices: target selection, feature selection, handling missing values, scaling, encoding.  
- EDA plots: target distribution, correlation heatmap, and feature vs target plots.

# 4. Project Architecture

The project follows a modular architecture:

app.py  
 └── wizard/  
 ├── \_\_init\_\_.py  
 ├── state.py (WizardState – tracks data, features, target, preprocessing config, model)  
 ├── data\_io.py (dataset loading utilities)  
 ├── preprocess.py (preprocessing functions)  
 ├── eda.py (plotting functions)

└── tabs/

├── \_\_init\_\_.py

├── data\_tab.py

├── eda\_tab.py

├── train\_tab.py

└── predict\_tab.py  
 └── models/

├── \_\_init\_\_.py

├── registry.py (dictionary of available models)

└── train\_eval.py (training and evaluation pipeline)  
  
Legacy demo files (app\_DT.py, app\_RF.py, etc.) remain in the repo as references.