Product Requirements & Specification Document

Project Name

Klassify - Multi-Model Classification Explorer

Description

Klassify is an interactive Streamlit application enabling users to upload datasets, explore, and compare KNN, SVM, and Decision Tree classifiers. Users can tune hyperparameters, visualize decision boundaries, and evaluate models using confusion matrix, ROC, and F1 scores.

1. Goals & Objectives

Goal	Description
Educational Tool	Facilitate understanding of classification algorithms
Model Comparison	Enable side-by-side comparison of KNN, SVM, and Decision Tree
Interactive Exploration	Allow real-time hyperparameter tuning and visualization
User-Friendly Interface	Simple, intuitive UI for non-experts

2. Target Users

User Type	Needs
Students	Learn and experiment with classifiers
Educators	Demonstrate ML concepts interactively
Startups/Analysts	Rapidly prototype and compare models

3. Core Features

Feature	Description
Dataset Upload	Upload CSV files with labeled data
Data Preview	Display sample rows, feature info, and class distribution
Model Selection	Choose between KNN, SVM, Decision Tree
Hyperparameter Tuning	Adjustable sliders/inputs for key parameters per model
Train/Test Split	Adjustable split ratio
Model Training	Fit selected model(s) on user data
Evaluation Metrics	Display confusion matrix, ROC curve, F1 score

Decision Boundary Visualization	2D plot for datasets with 2 features
Model Comparison	Side-by-side metric and visualization comparison
Download Results	Export evaluation metrics and predictions

4. Functional Requirements

4.1 Data Handling

- Accept CSV files with numeric/categorical features and a target column
- Validate data format and handle missing values (drop or impute)
- Allow user to select target column and features

4.2 Model Support

Model	Hyperparameters
KNN	n_neighbors, weights, metric
SVM	kernel, C, gamma
Decision Tree	max_depth, criterion, splitter

- Use scikit-learn implementations

4.3 Visualization

- Display data distribution and feature histograms
- Plot decision boundaries (2D, for 2 features)
- Show confusion matrix, ROC curve, F1 score

4.4 User Interface

- Streamlit-based, responsive layout
- Sidebar for controls (upload, model selection, hyperparameters)
- Main area for visualizations and results

5. Non-Functional Requirements

Requirement	Specification
Performance	<2s response for datasets <10,000 rows
Usability	Intuitive, minimal steps to run comparisons
Compatibility	Chrome, Firefox, Edge (latest versions)
Security	No data stored server-side; session-based only
Extensibility	Modular code for adding new models/metrics

6. Technical Stack

Component	Technology
Frontend	Streamlit
Backend	Python
ML Libraries	scikit-learn
Data Handling	pandas, numpy
Visualization	matplotlib, seaborn, plotly (optional)

7. User Flow

```
graph TD
A[Upload Dataset] --> B[Select Target & Features]
B --> C[Choose Model(s)]
C --> D[Set Hyperparameters]
D --> E[Train/Test Split]
E --> F[Train & Evaluate]
F --> G[View Visualizations & Metrics]
G --> H[Download Results]
```

8. Acceptance Criteria

ID	Criteria
AC1	User can upload and preview dataset
AC2	User can select and configure any of the three models
AC3	App displays confusion matrix, ROC, and F1 for each model
AC4	Decision boundary is visualized for 2-feature datasets
AC5	Users can compare models side-by-side
AC6	Results and metrics can be downloaded

9. Constraints & Risks

- Only 2D decision boundaries (for 2 features)
- Large datasets (>10k rows) may impact performance
- No support for text/image data

10. Milestones

Milestone	Description	Target Date
Prototype UI	Basic Streamlit layout	Week 1

Data Upload & Preview	CSV upload, data validation	Week 2
Model Integration	KNN, SVM, Decision Tree	Week 3
Visualization	Metrics, decision boundaries	Week 4
Model Comparison	Side-by-side results	Week 5
Final Testing & Release	QA, documentation	Week 6

11. Appendix

Example Model Training Pseudocode

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, roc_curve, f1_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
```

End of Document