

# Smart Crop Management System

A modular, AI-powered agricultural decision support system built with Python and Tkinter. This application provides crop recommendations, fertilizer suggestions, and yield predictions using machine learning algorithms.

## Project Structure

```
crop_management_system/  
├── config.py          # Configuration and constants  
├── data_generator.py  # Sample data generation  
├── ml_models.py       # Machine learning models  
├── prediction_engine.py # Prediction logic and result formatting  
├── visualizations.py  # Data visualization components  
├── data_manager.py    # Data management and file operations  
├── ui_components.py   # UI components and widgets  
├── main_gui.py        # Main GUI application  
├── requirements.txt   # Required dependencies  
└── README.md          # Project documentation
```

## Features

### Core Functionality

- **Crop Recommendation:** AI-powered crop suggestions based on soil and weather parameters
- **Fertilizer Recommendation:** Smart fertilizer suggestions based on soil composition and crop type
- **Yield Prediction:** Predictive analytics for crop yield estimation
- **Data Visualization:** Interactive charts and graphs for data analysis
- **Data Management:** Import/export functionality for CSV files

### Technical Features

- **Modular Architecture:** Clean separation of concerns with interconnected modules
- **Machine Learning:** Random Forest algorithms for classification and regression
- **User-friendly GUI:** Intuitive interface built with Tkinter
- **Real-time Predictions:** Instant results with confidence scores
- **Data Export:** Save predictions and analysis results

## Prerequisites

- Python 3.7 or higher
- Required packages (see requirements.txt)

## Installation

### 1. Clone or download the project files

```
bash  
  
mkdir crop_management_system  
cd crop_management_system
```

### 2. Install required dependencies

```
bash  
  
pip install -r requirements.txt
```

### 3. Run the application

```
bash  
  
python main_gui.py
```

## Usage



### Starting the Application

Run `main_gui.py` to start the application. The system will automatically:



- Generate sample datasets
- Train machine learning models
- Initialize the user interface

### Using Different Modules



#### Crop Recommendation

1. Navigate to the " Crop Recommendation" tab
2. Enter soil and weather parameters:
  - Nitrogen (N), Phosphorus (P), Potassium (K) levels
  - Temperature, Humidity, pH, Rainfall
3. Click " Get Crop Recommendation"
4. View detailed results with confidence scores


## Fertilizer Recommendation

1. Go to the "  Fertilizer Recommendation" tab
2. Select soil type and crop type from dropdowns
3. Enter environmental and nutrient parameters
4. Click "  Get Fertilizer Recommendation"
5. Review fertilizer suggestions and nutrient analysis


## Yield Prediction

1. Access the "  Yield Prediction" tab
2. Select location (state, district) and crop details
3. Enter area and production values
4. Click "  Predict Yield"
5. Analyze yield predictions and recommendations

## Data Analysis

1. Visit the "  Data Analysis" tab
2. Choose from visualization options:
  - Crop Distribution
  - Parameter Analysis
  - Yield Trends
3. Interactive charts will display data insights

## Data Management

1. Use the "  Data Management" tab to:
  - Load custom CSV datasets
  - Export results and predictions
  - Retrain models with new data
  - View dataset information

## Module Details

### config.py

Contains all configuration settings, constants, and data definitions:

- Application settings (window size, colors, styles)
- Model parameters (n\_estimators, test\_size, etc.)
- Field definitions for input forms
- Crop and fertilizer information databases

## **data\_generator.py**

Generates synthetic datasets for demonstration:

- Crop recommendation data (NPK levels, weather, soil pH)
- Fertilizer recommendation data (soil types, crop types, nutrients)
- Yield prediction data (location, production, area)

## **ml\_models.py**

Handles all machine learning operations:

- Model training (RandomForest for classification/regression)
- Data preprocessing and encoding
- Prediction methods for all model types
- Model evaluation and metrics

## **prediction\_engine.py**

Processes predictions and formats results:

- Formats prediction outputs with detailed explanations
- Provides crop-specific recommendations
- Analyzes nutrient levels and deficiencies
- Calculates yield comparisons and insights

## **visualizations.py**

Creates interactive data visualizations:

- Crop distribution charts (pie charts, bar graphs)
- Parameter correlation analysis (scatter plots, histograms)
- Yield trend analysis (state/crop comparisons)
- Feature importance plots

## **data\_manager.py**

Manages data operations:

- CSV file import/export functionality
- Data validation and quality checks
- Backup and restore operations
- Dataset information and metadata

## **ui\_components.py**

Provides reusable UI components:

- Input panels with validation
- Result display areas
- Button panels and controls
- Status bars and progress indicators
- Data tree views

## **main\_gui.py**

Main application controller:

- Initializes all system components
- Coordinates between modules
- Handles user interactions
- Manages application lifecycle



## **Customization**

### **Adding New Crops**

Edit `config.py` to add new crop types:

```
python
```

```
CROP_INFO['new_crop'] = {
    'recommendations': [
        "• Specific cultivation tip 1",
        "• Specific cultivation tip 2"
    ]
}
```

## Modifying UI Styles

Update styling in `config.py`:

```
python

STYLES['custom_style'] = {
    'font': ('Arial', 12, 'bold'),
    'foreground': '#your_color'
}
```

## Adding New Visualizations

Extend `visualizations.py`:

```
python

def show_custom_analysis(self, data):
    # Your custom visualization code
    pass
```

## Troubleshooting

### Common Issues

#### 1. Import Errors

- Ensure all files are in the same directory
- Check that all dependencies are installed

#### 2. Model Training Failures

- Verify data integrity
- Check for sufficient sample sizes (minimum 50 samples)

#### 3. GUI Display Issues

- Ensure tkinter is properly installed

- Check system compatibility

#### 4. Performance Issues

- Reduce dataset size for faster training
- Adjust model parameters in config.py

## Contributing

To extend or modify the system:

1. **Adding New Features:** Create new modules following the existing patterns
2. **Modifying Models:** Update `ml_models.py` with new algorithms
3. **UI Changes:** Modify `ui_components.py` and update `main_gui.py`
4. **Data Sources:** Extend `data_manager.py` for new data formats

## Technical Specifications

- **Framework:** Python 3.7+ with Tkinter
- **ML Libraries:** scikit-learn, pandas, numpy
- **Visualization:** matplotlib, seaborn
- **Architecture:** Modular object-oriented design
- **Data Format:** CSV files with structured schemas
- **Model Types:** Random Forest (Classification & Regression)

## Performance Notes

- Initial model training: ~2-3 seconds
- Prediction time: <100ms per request
- Memory usage: ~50-100MB depending on dataset size
- Supports datasets up to 10,000+ samples efficiently

## Data Privacy

- All processing is done locally
- No external data transmission
- Sample data is synthetically generated
- User data remains on local machine

## License

This project is provided as-is for educational and research purposes. Feel free to modify and distribute according to your needs.

---

**Happy Farming!** 🌱 🚜