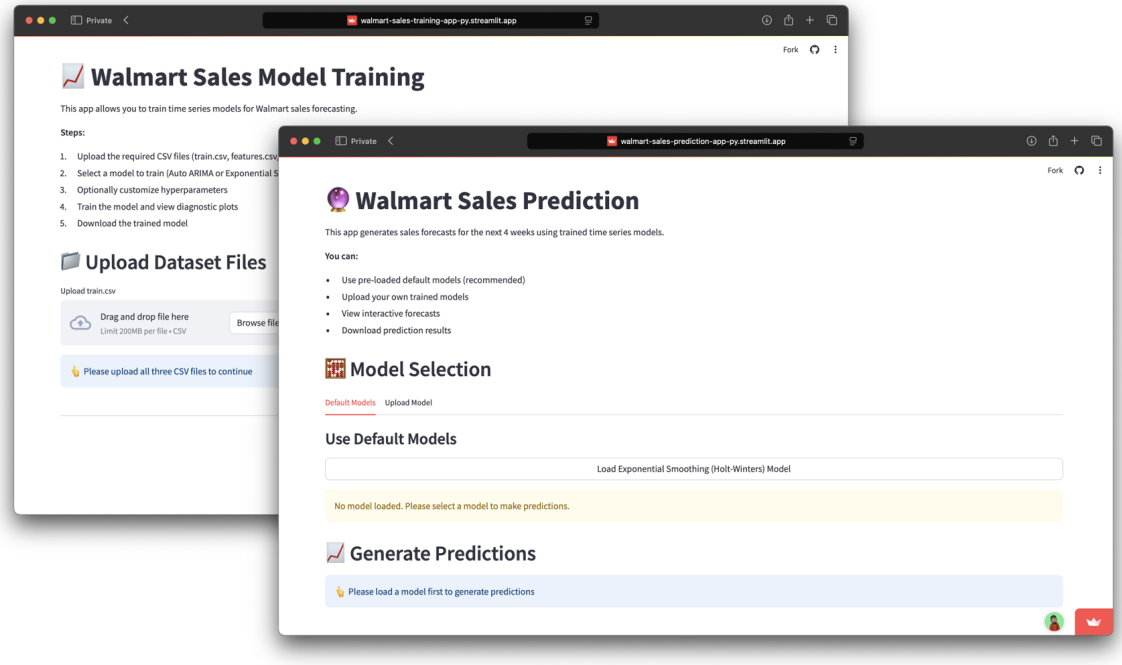
University of Applied Sciences

HOCHSCHULE EMDEN·LEER

Manual

Walmart Sales Forecasting Web Applications



Author: Hemanth Jadiswami Prabhakaran
Matriculation No.: 7026000
Course of Studies: Mechanical Engineering

First examiner: Prof. Dr. Elmar Wings
Submission date: June 30, 2025

University of Applied Sciences Emden/Leer · Faculty of Technology · Mechanical Engineering Department
Constantiaplatz 4 · 26723 Emden · `http://www.hs-emden-leer.de`

Contents

List of Figures

List of Tables

Acronyms

1 Introduction and Main Function

1.1 Overview

The Walmart Sales Forecasting System is a comprehensive two-application solution developed as part of a Master's Business Analytics course project. This system demonstrates advanced time series forecasting capabilities using state-of-the-art machine learning algorithms to predict weekly sales changes for retail environments.

The system architecture consists of two complementary applications designed to provide a complete forecasting workflow from model development to production predictions.

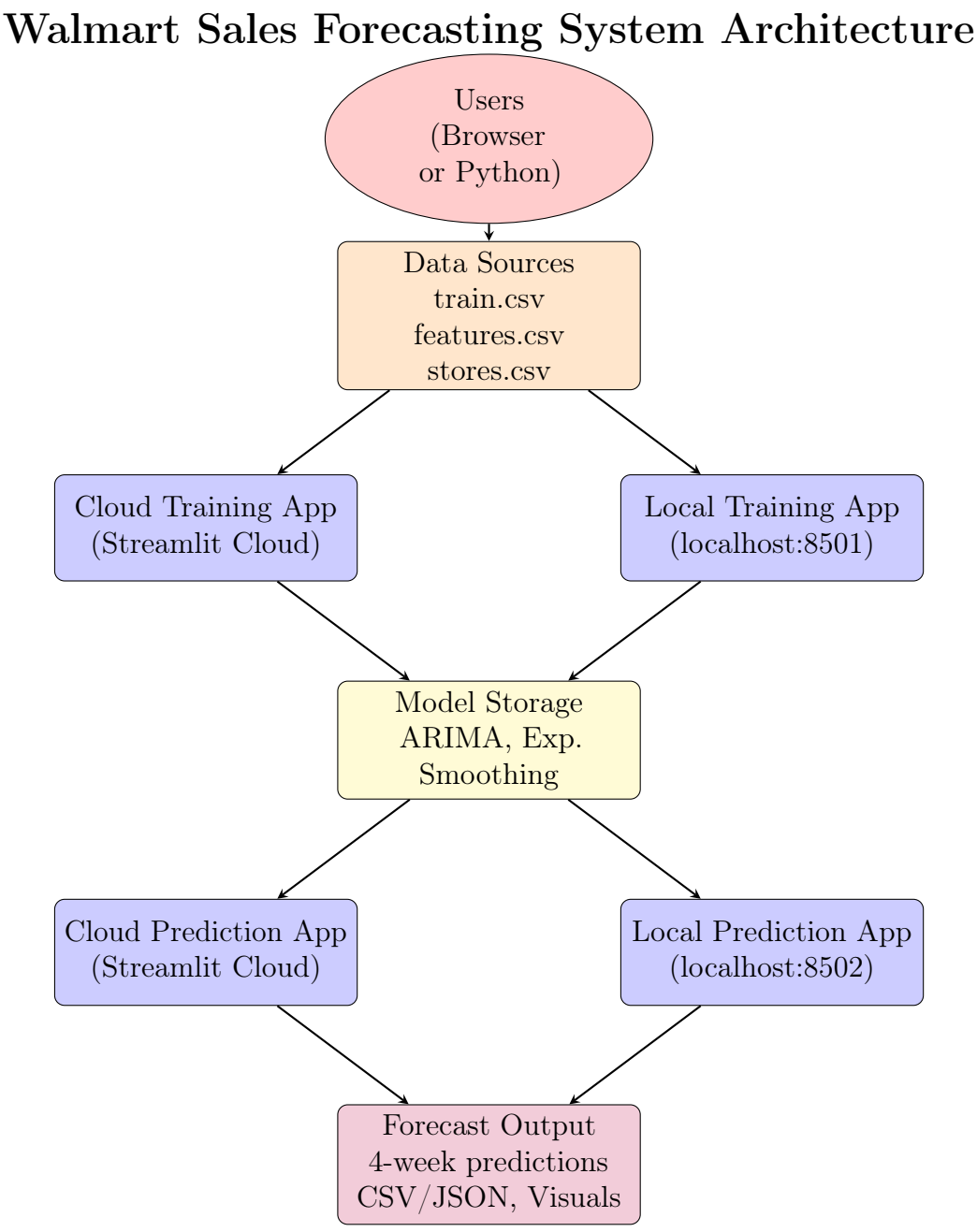


Figure 1.1: Walmart Sales Forecasting System Architecture Overview

1.2 Primary Capabilities

1.2.1 Dual-Application Architecture

The system operates through two specialized applications, each serving distinct functions in the forecasting pipeline:

1. **Training Application**: Provides a user-friendly interface for developing and training new forecasting models

2. **Prediction Application**: Delivers production-ready forecasting capabilities with pre-trained models

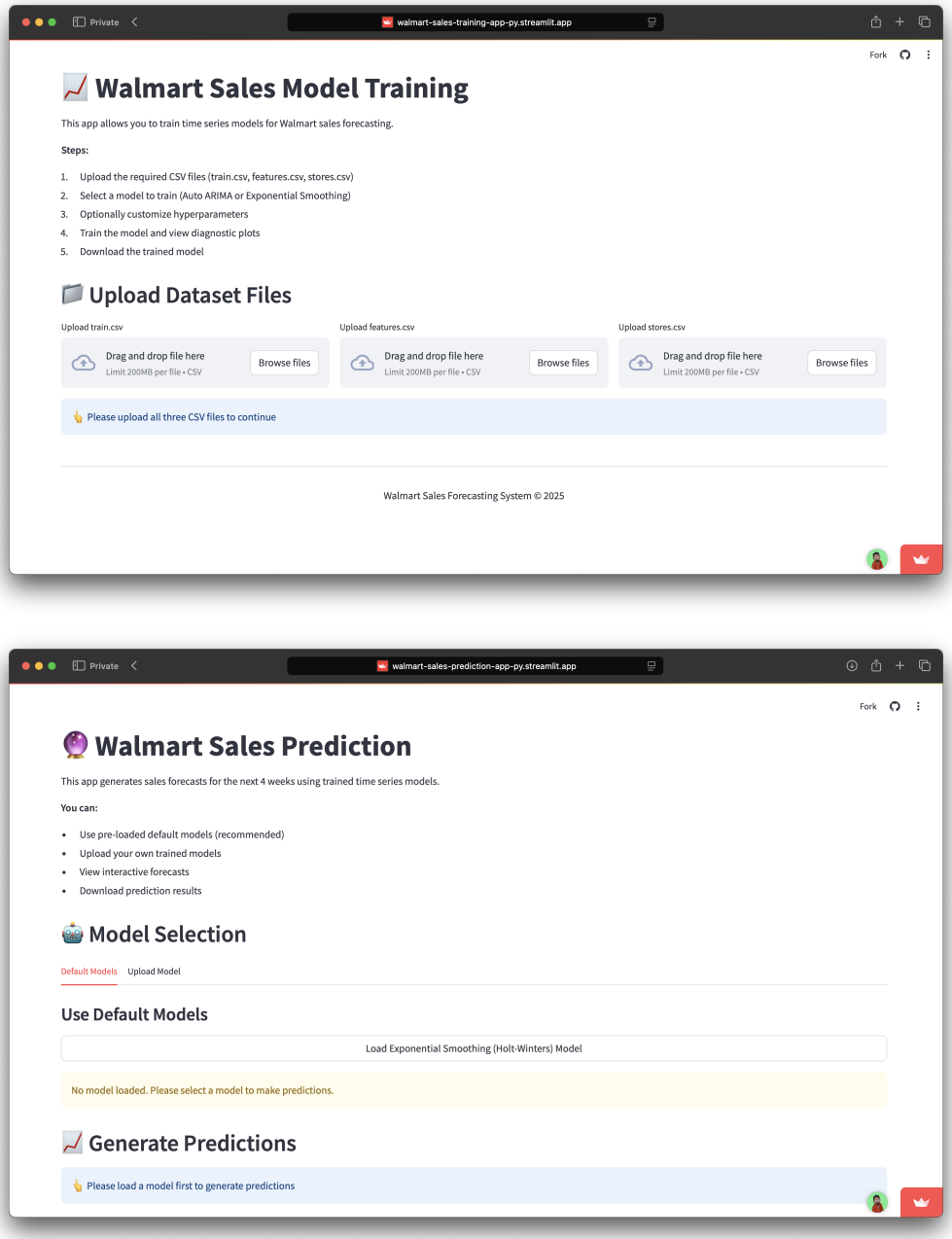


Figure 1.2: Dual-Application Interface Comparison

1.2.2 Advanced Time Series Modeling

The system implements two sophisticated forecasting algorithms:

- **Auto ARIMA**: Automated ARIMA model selection with seasonal components
- **Exponential Smoothing (Holt-Winters)**: Triple exponential smoothing with trend and seasonal components

1.2.3 Flexible Deployment Options

Both applications support multiple deployment scenarios:

- **Cloud Access**: Immediate browser-based access via Streamlit Cloud
- **Local Installation**: Python 3.12 virtual environment setup for offline usage

1.3 Core Functionality

1.3.1 Model Training Capabilities

The Training Application provides comprehensive model development features:

- **Data Upload**: Support for standard Walmart dataset format (train.csv, features.csv, stores.csv)
- **Automated Preprocessing**: Data cleaning, validation, and feature engineering
- **Hyperparameter Tuning**: Customizable model parameters for optimal performance
- **Performance Evaluation**: WMAE-based model assessment with visual diagnostics
- **Model Export**: Trained models saved in .pkl format for deployment

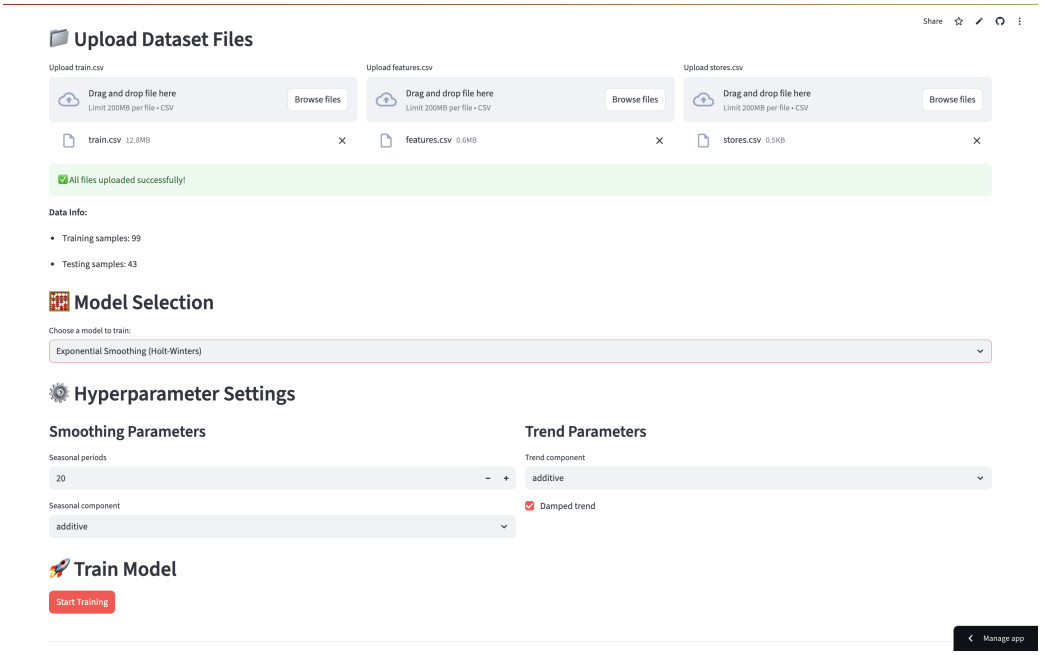


Figure 1.3: Model Training Workflow Interface

1.3.2 Production Forecasting Features

The Prediction Application offers robust forecasting capabilities:

- **Pre-loaded Model**: High-performance Exponential Smoothing model (3.58% WMAE)
- **Custom Model Support**: Upload and use personally trained models
- **4-Week Forecasting**: Generate week-over-week sales change predictions
- **Interactive Visualizations**: Dynamic charts with positive/negative change indicators
- **Export Capabilities**: Download results in CSV and JSON formats

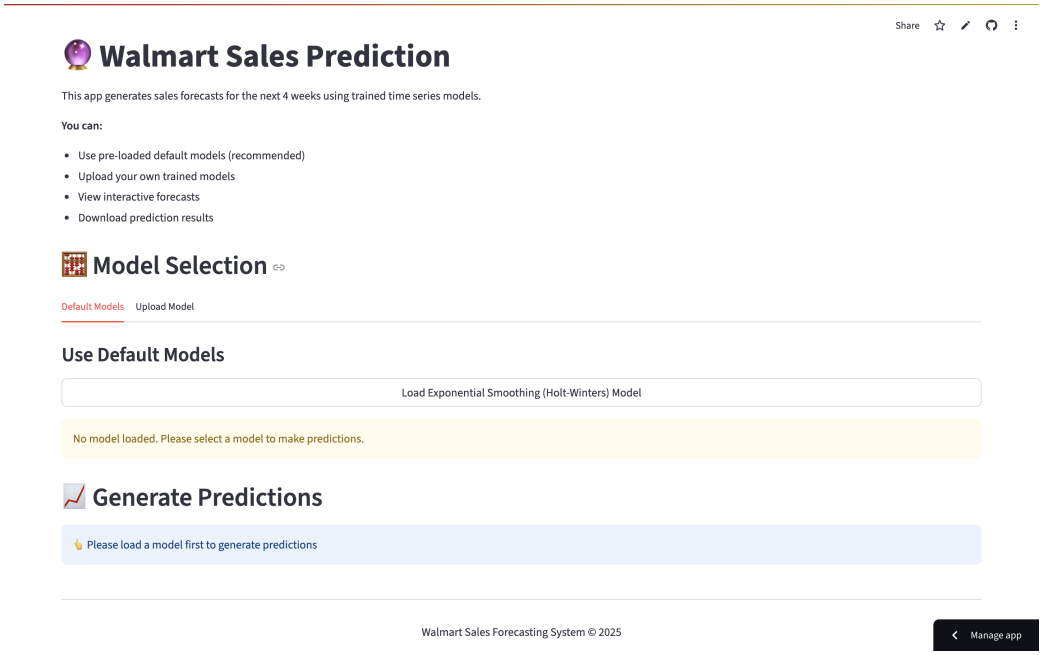


Figure 1.4: Prediction Application Main Interface

1.4 Key Performance Characteristics

1.4.1 Model Accuracy

The default Exponential Smoothing (Holt-Winters) model demonstrates excellent performance:

- **Normalized WMAE**: 3.58% (Excellent category - less than 5% error)
- **Absolute WMAE**: 923.12 (dollars in weekly sales change)
- **Prediction Horizon**: 4 weeks ahead

1.4.2 Technical Specifications

- **Python Version**: 3.12 (exact version requirement)
- **Framework**: Streamlit for web interface
- **Model Format**: Joblib .pkl serialization
- **Data Format**: CSV input with specific schema requirements

1.5 Target Applications

This forecasting system is designed for multiple use cases:

- **Academic Research**: Educational tool for time series analysis learning
- **Business Analytics**: Sales forecasting for retail planning
- **Model Development**: Platform for experimenting with forecasting algorithms
- **Production Deployment**: Operational forecasting for decision support

1.6 Document Structure

This user manual provides comprehensive guidance organized into nine core chapters:

1. **Introduction and Main Function**: System overview and capabilities
2. **Installation and Setup**: Local and cloud deployment procedures
3. **First Steps Guide**: Initial usage walkthrough
4. **GUI and User Interface**: Detailed interface documentation
5. **Application Functions**: Complete feature coverage
6. **System Specifications**: Technical requirements and data formats
7. **Maintenance and Best Practices**: Operational guidance
8. **Troubleshooting**: Problem resolution procedures
9. **Help and Support**: Additional assistance resources

Each chapter includes practical examples, screenshots, and step-by-step procedures to ensure effective system utilization.

2 Installation and Setup

2.1 Deployment Options Overview

The Walmart Sales Forecasting System offers two deployment methods to accommodate different user needs and technical environments. Each option provides full system functionality with specific advantages for different use cases.

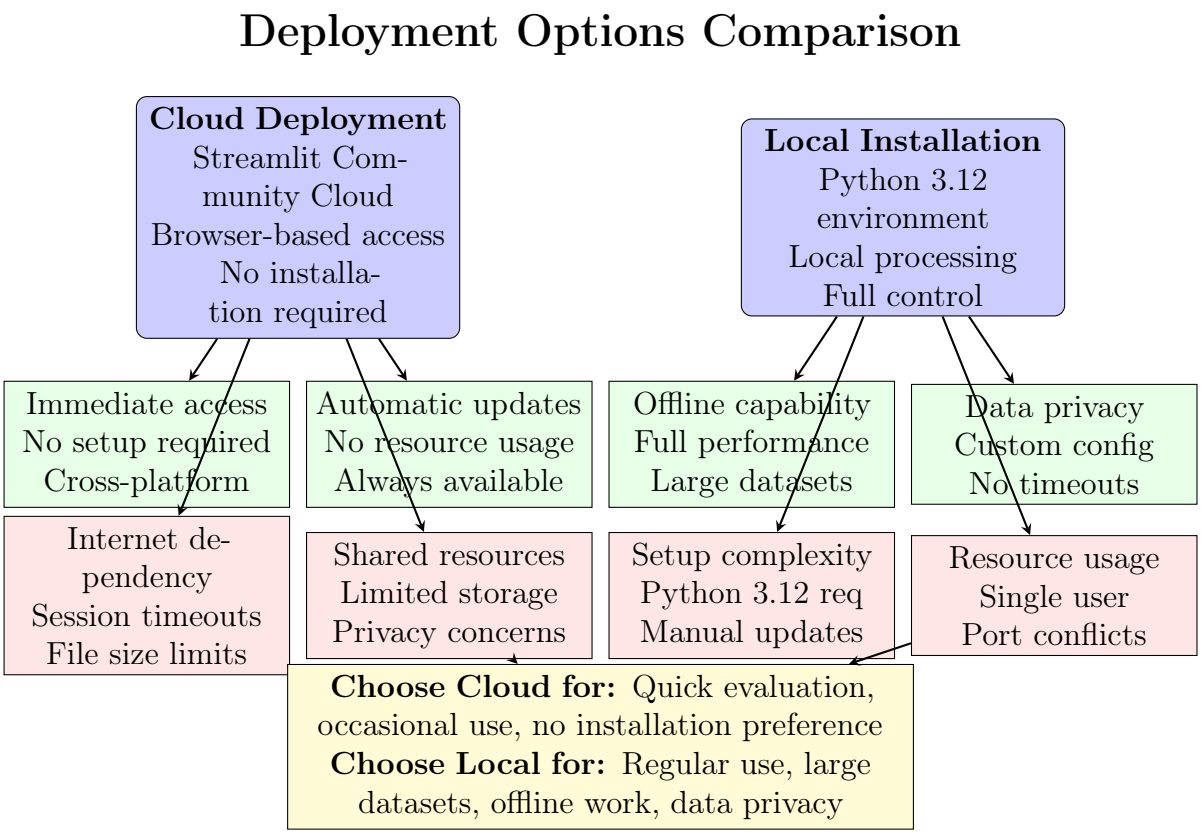


Figure 2.1: Deployment Options Technical Comparison

2.2 Cloud Access (Recommended for Quick Start)

2.2.1 Immediate Browser Access

For immediate system access without any installation requirements, use the cloud-hosted applications:

- **Training Application**: `https://walmart-sales-training-app-py.streamlit.app/`
- **Prediction Application**: `https://walmart-sales-prediction-app-py.streamlit.app/`

2.2.2 Cloud Access Procedure

1. Open a modern web browser (Chrome, Firefox, Safari, or Edge)
2. Navigate to the desired application URL
3. Wait for the Streamlit application to load (typically 10-30 seconds)
4. Begin using the application immediately

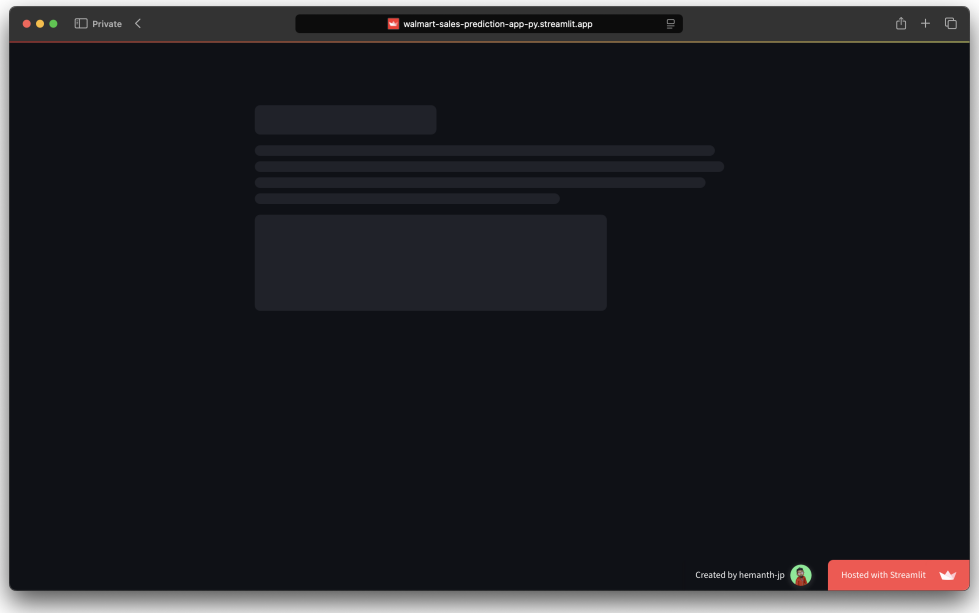


Figure 2.2: Cloud Application Loading Interface

2.2.3 Cloud Advantages

- **No Installation Required**: Immediate access without software setup
- **Cross-Platform Compatibility**: Works on any device with a web browser
- **Automatic Updates**: Always running the latest version
- **No Resource Requirements**: Server-side processing

2.2.4 Cloud Limitations

- **Internet Dependency**: Requires stable internet connection
- **File Upload Size Limits**: Large dataset restrictions may apply
- **Session Timeout**: Extended idle periods may reset the application

2.3 Local Installation (Advanced Users)

2.3.1 System Requirements

Before proceeding with local installation, ensure your system meets these requirements:

- **Python Version**: Exactly Python 3.12.x (not 3.12+ or other versions)
- **Operating System**: Windows, macOS, or Linux
- **Available Storage**: Minimum 2GB free space
- **RAM**: Minimum 4GB (8GB recommended for large datasets)

2.3.2 Python 3.12 Installation Verification

First, verify your Python installation:

```
# Check Python version (must be exactly 3.12.x)
python --version
# or on some systems:
python3 --version
```

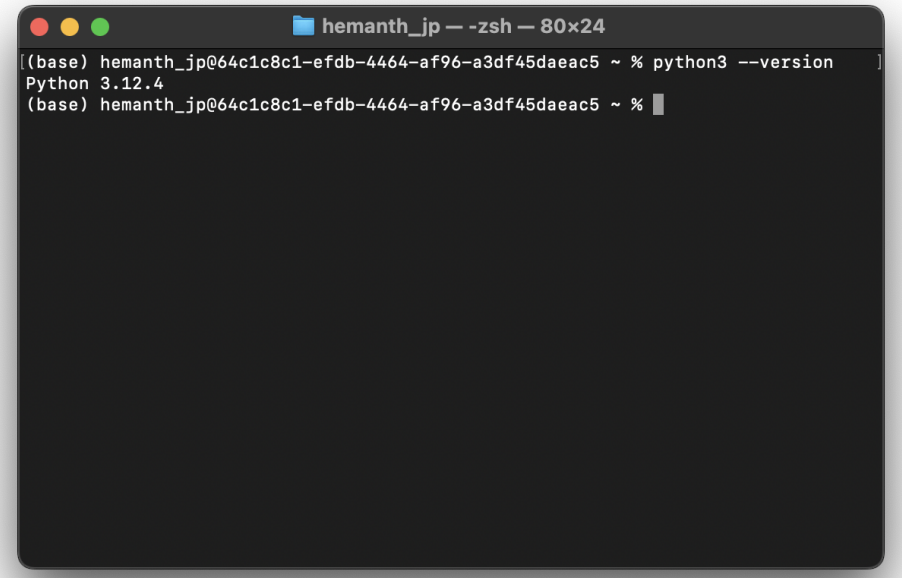


Figure 2.3: Python Version Verification

Important: If you do not have Python 3.12, download it from `https://www.python.org/downloads/` and install the exact 3.12.x version.

2.3.3 Virtual Environment Setup

Create an isolated Python environment for the forecasting system: Navigate to ../GitHub/BA25-01-Time-Series/Code/WalmartSalesPredictionApp and ../GitHub/BA25-01-Time-Series

Windows Setup

```
# Create virtual environment
python -m venv walmart_forecast_env

# Activate virtual environment
walmart_forecast_env\Scripts\activate

# Verify activation (prompt should show environment name)
```

macOS/Linux Setup

```
# Create virtual environment
```

```
python3 -m venv walmart_forecast_env

# Activate virtual environment
source walmart_forecast_env/bin/activate

# Verify activation (prompt should show environment name)
```

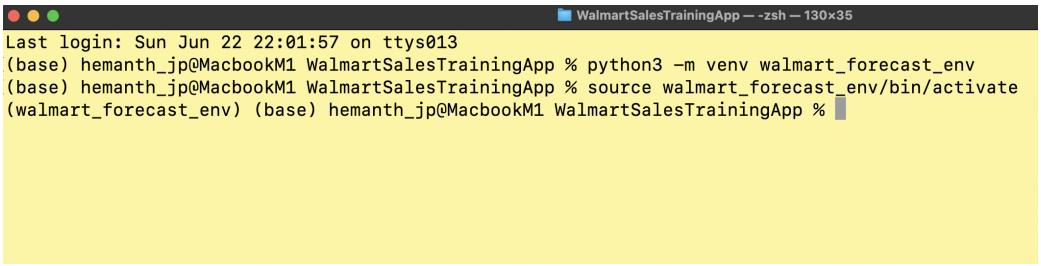


Figure 2.4: Virtual Environment Creation and Activation

2.3.4 Dependencies Installation

With the virtual environment activated, install required packages:

```
# Install all required dependencies
pip install -r requirements.txt

# Verify installation success
pip list
```

The `requirements.txt` file includes these essential packages:

- streamlit==1.32.0
- pandas==2.2.2
- numpy==1.26.4
- matplotlib==3.8.4
- seaborn==0.13.2
- plotly==5.24.1
- joblib==1.4.2
- statsmodels==0.14.2
- pmdarima==2.0.4
- pytest==7.4.4

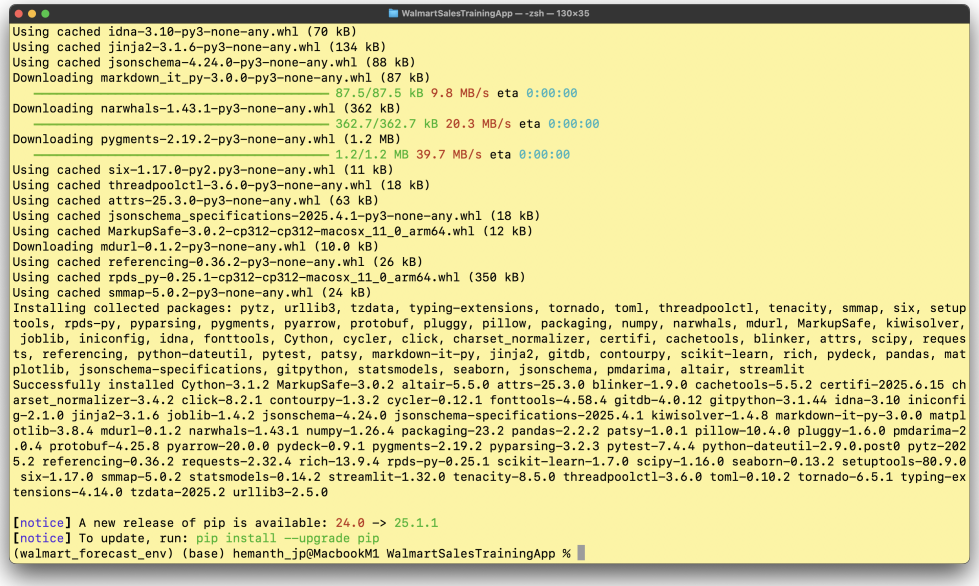


Figure 2.5: Dependencies Installation Process

Do the same for the other app as well.

2.3.5 Local Application Launch

Launch both applications on different ports to run simultaneously:

Training Application

```
# Launch Training Application (default port 8501)
streamlit run walmartSalesTrainingApp.py
```

Prediction Application

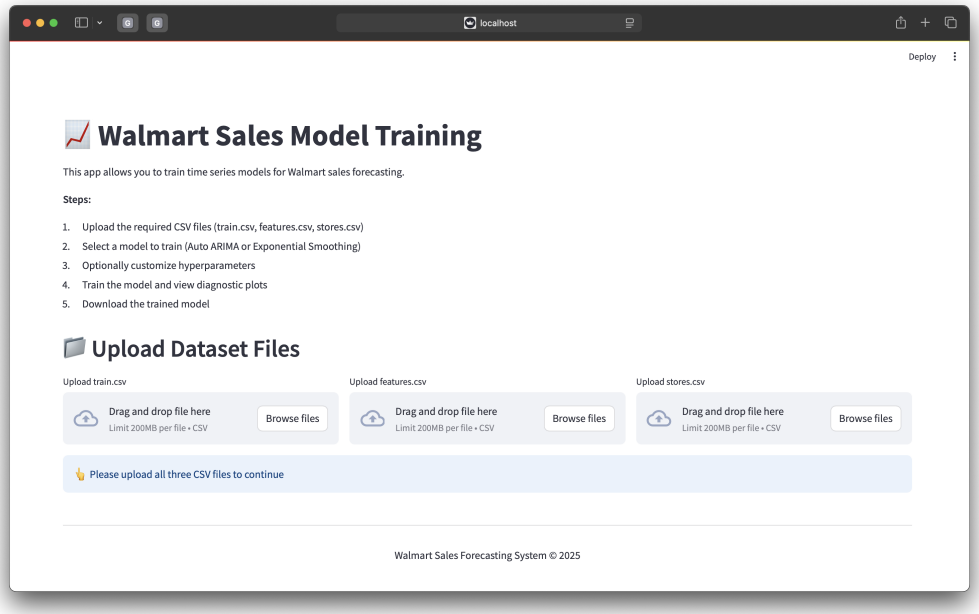
```
# Launch Prediction Application (port 8502)
streamlit run walmartSalesPredictionApp.py --server.port=8502
```

Both applications will open automatically in your default web browser.

```
(walmart_forecast_env) (base) hemanth_jp@MacbookM1 WalmartSalesTrainingApp % streamlit run walmartSalesTrainingApp.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.178.177:8501
```




```
(walmart_forecast_env) (base) hemanth_jp@MacbookM1 WalmartSalesPredictionApp % streamlit run walmartSalesPredictionApp.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.178.177:8502
```

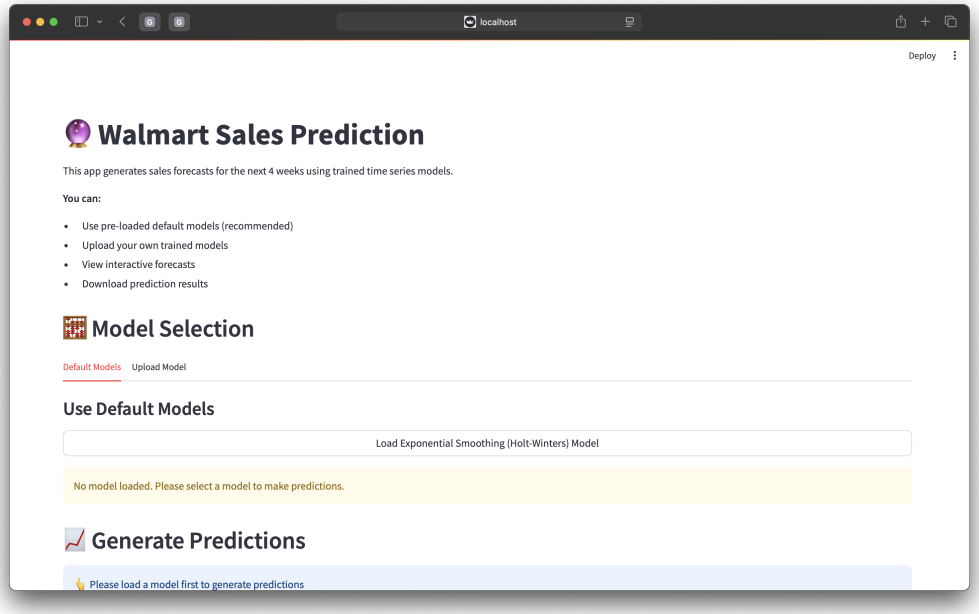


Figure 2.6: Local Application Launch Success

2.4 Installation Validation

2.4.1 Test Suite Execution

Verify your local installation by running the comprehensive test suite:

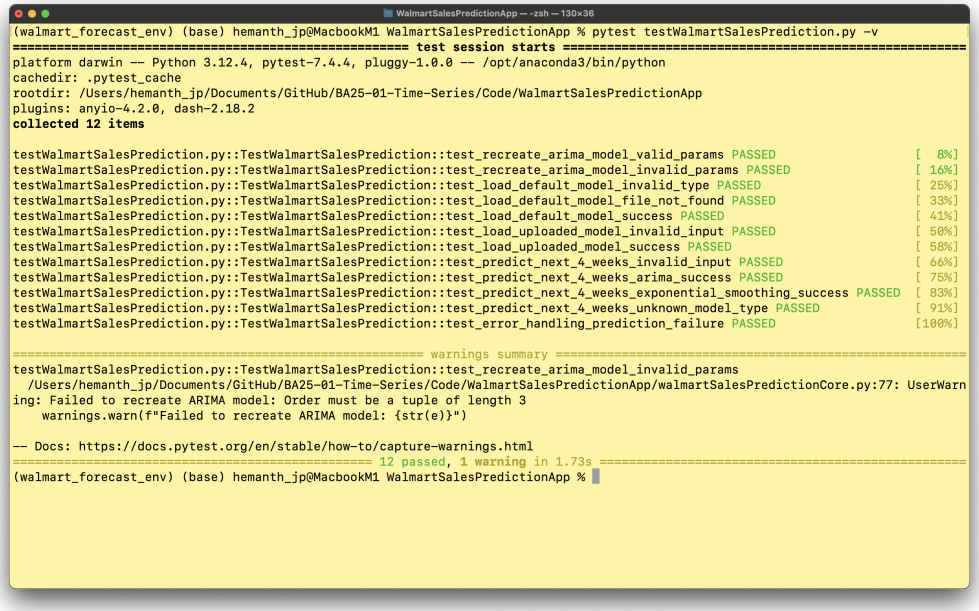
```
# Test Training Application functionality
pytest testWalmartSalesTraining.py -v

# Test Prediction Application functionality
pytest testWalmartSalesPrediction.py -v
```

2.4.2 Successful Installation Indicators

A successful installation is confirmed when:

- All pytest tests pass without errors
- Both applications launch without import errors
- Web interfaces load completely in the browser
- No missing dependency warnings appear



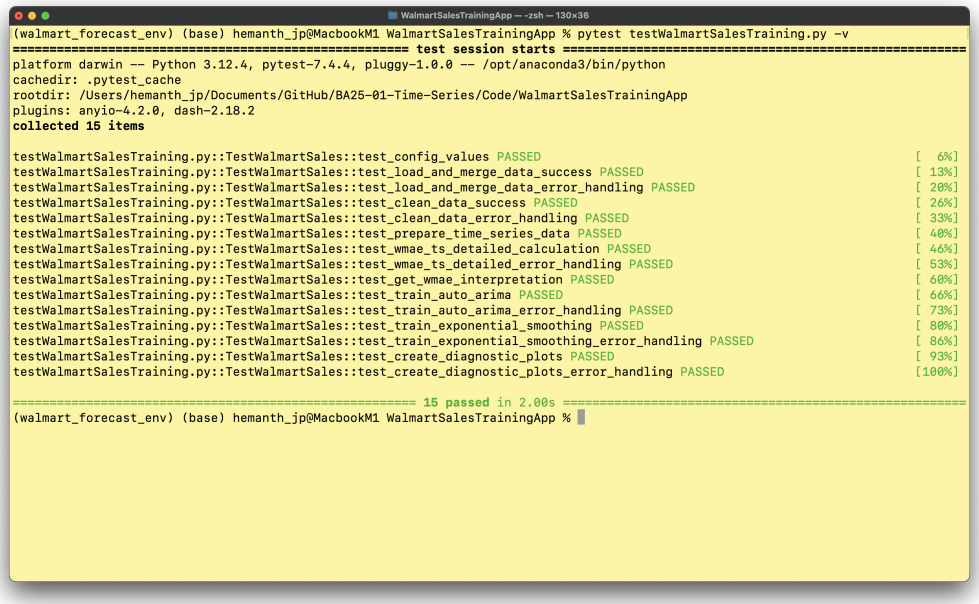


Figure 2.7: Successful Test Suite Execution

2.5 Troubleshooting Common Setup Issues

2.5.1 Python Version Conflicts

Issue: Wrong Python version or multiple versions installed. **Solution**: Use explicit version commands:

```
python3.12 -m venv walmart_forecast_env
```

2.5.2 Virtual Environment Activation Problems

Issue: Virtual environment not activating properly. **Solution**: Ensure correct path and permissions:

- Windows: Check for spaces in path names
- macOS/Linux: Verify execute permissions on activation script

2.5.3 Dependency Installation Failures

Issue: Package installation errors or conflicts. **Solution**: Update pip and use exact versions:

```
pip install --upgrade pip
pip install -r requirements.txt --force-reinstall
```

2.6 Next Steps

After successful installation, proceed to Chapter 3 for your first forecasting session or jump directly to Chapter 4 for detailed interface documentation.

3 First Steps Guide

3.1 Getting Started Overview

This chapter provides a complete walkthrough for new users to generate their first sales forecast using the Walmart Sales Forecasting System. We will demonstrate the quickest path to results using the cloud-based Prediction Application with the pre-loaded model.

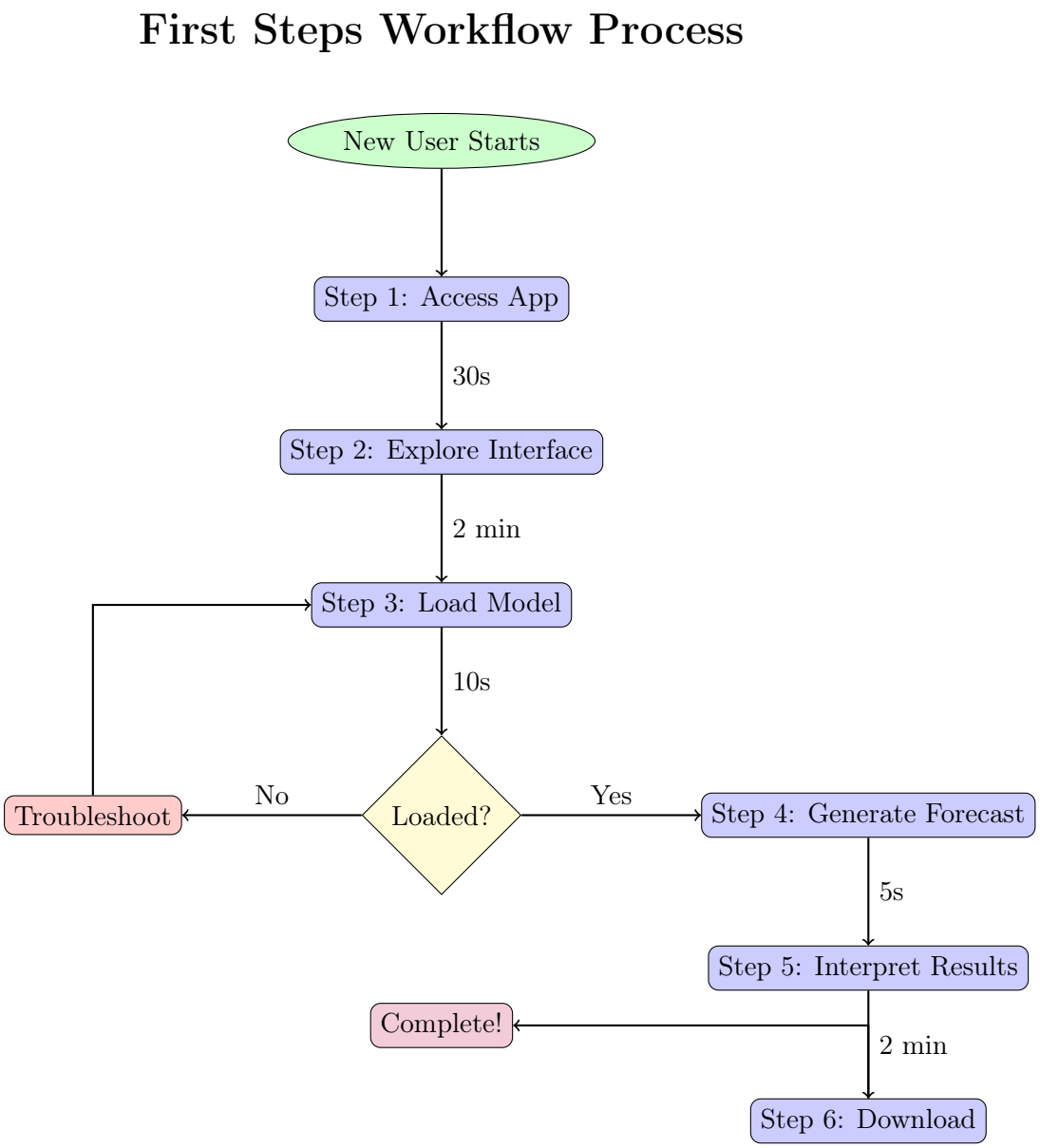


Figure 3.1: First Steps Workflow Process

3.2 Quick Start: Your First Forecast

3.2.1 Step 1: Access the Prediction Application

Open your web browser and navigate to the Prediction Application:
`https://walmart-sales-prediction-app-py.streamlit.app/`
The application will load within 10-30 seconds, displaying the main interface.

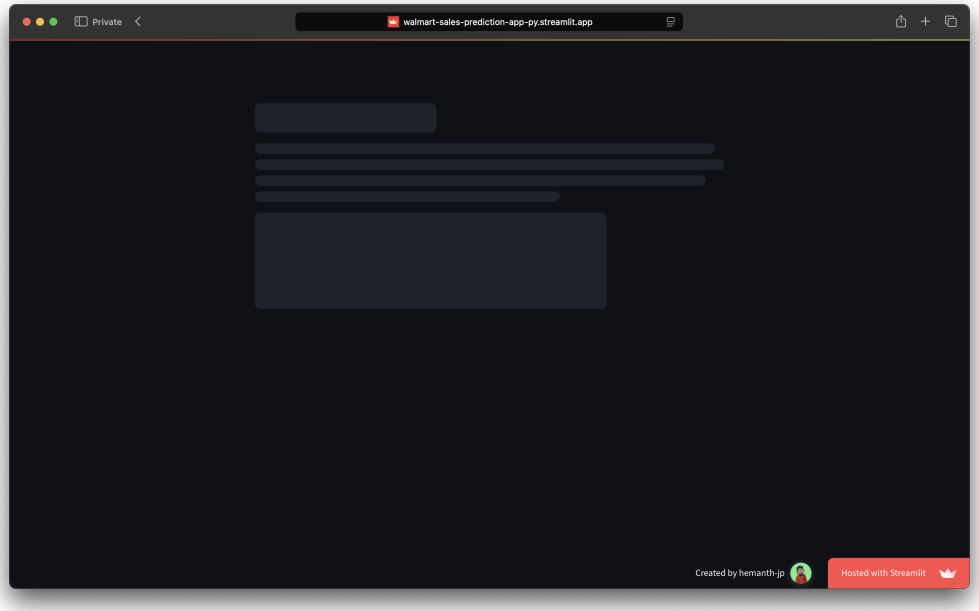


Figure 3.2: Prediction Application Loading Screen

3.2.2 Step 2: Understand the Interface Layout

Upon loading, you will see the main interface with these key sections:

- **Header**: Application title and description
- **Model Selection**: Default and custom model options
- **Generate Predictions**: Forecasting controls
- **Results Display**: Charts and data tables (appears after prediction)

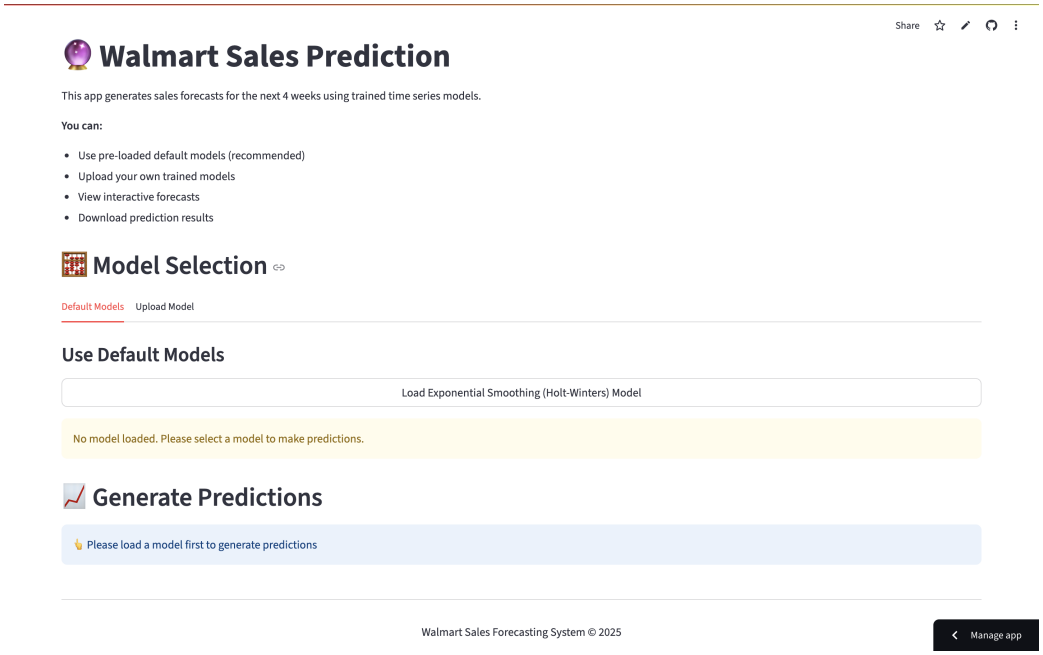


Figure 3.3: Main Interface Layout Overview

3.2.3 Step 3: Load the Default Model

The system comes with a pre-trained high-performance model. To load it:

1. Locate the **Model Selection** section
2. Click on the **Default Models** tab
3. Click the **Load Exponential Smoothing (Holt-Winters) Model** button
4. Wait for the success message: " Exponential Smoothing (Holt-Winters) model loaded successfully!"

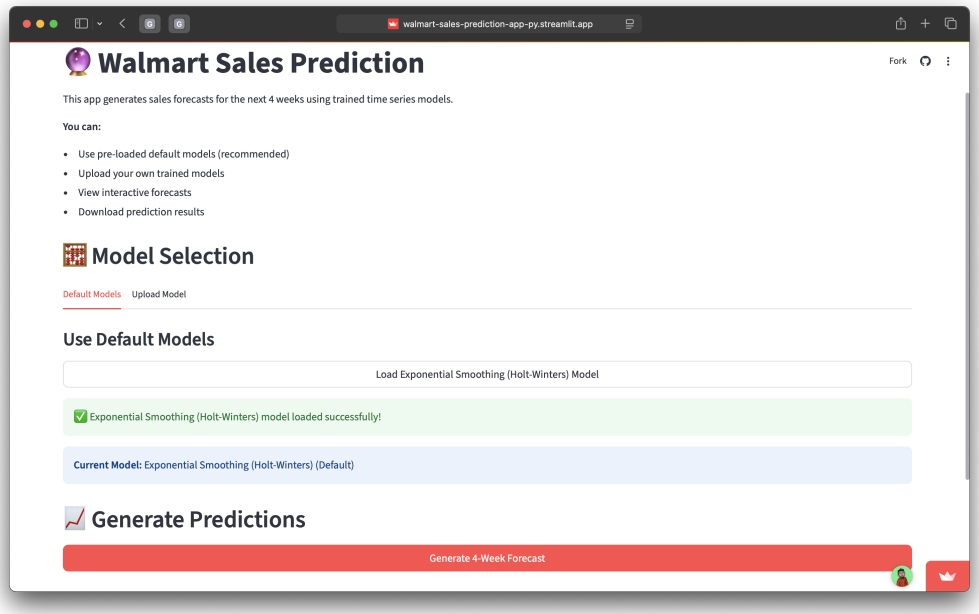


Figure 3.4: Default Model Loading Process

3.2.4 Step 4: Generate Your First Forecast

With the model loaded, generate predictions:

1. Scroll down to the **Generate Predictions** section

2. Click the **Generate 4-Week Forecast** button

3. The system will process for a few seconds

4. Results will appear below with charts and data tables

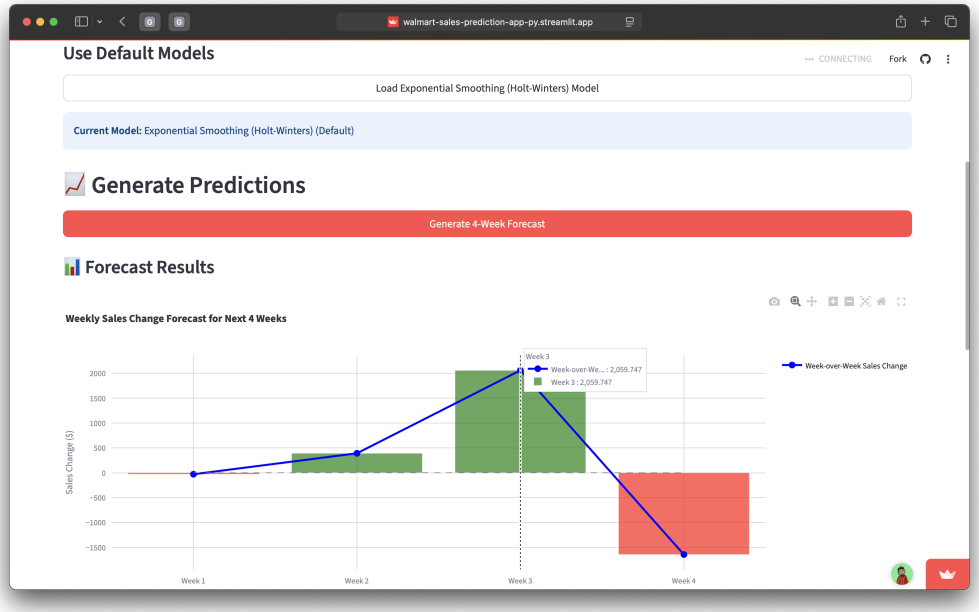


Figure 3.5: Forecast Generation Process

3.3 Understanding Your First Results

3.3.1 Forecast Interpretation

The generated forecast shows **week-over-week sales changes**, not absolute sales values:

- **Positive Values** (Green): Sales increase from previous week
- **Negative Values** (Red): Sales decrease from previous week
- **Dollar Amounts**: Change in sales dollars, not total sales

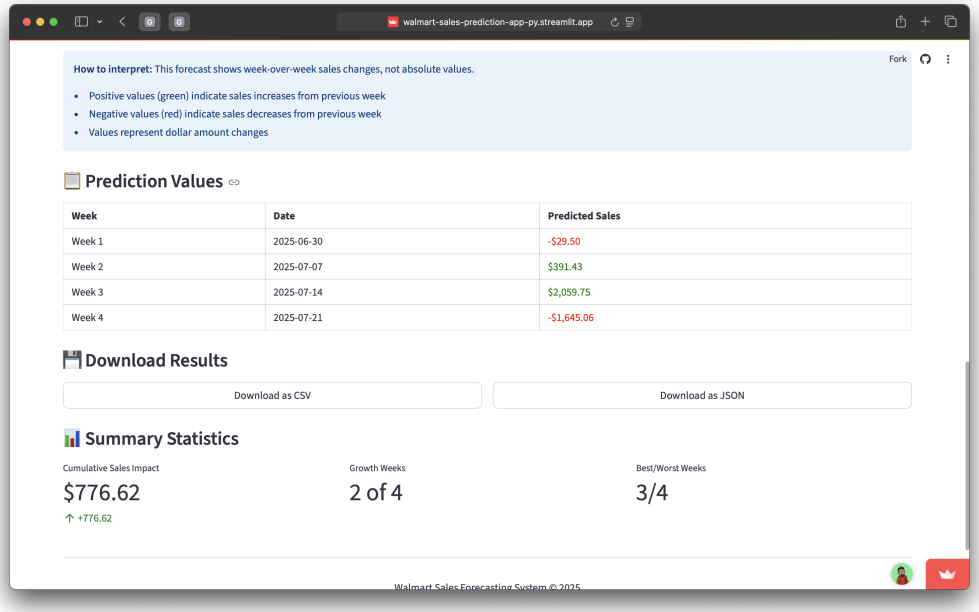


Figure 3.6: Sample Forecast Results with Interpretation

3.3.2 Interactive Visualization

The forecast chart provides interactive features:

- **Hover Information**: Point your mouse over data points for detailed values
- **Zoom**: Use mouse wheel or touch gestures to zoom in/out
- **Pan**: Click and drag to move around the chart
- **Color Coding**: Green bars for growth, red bars for decline

3.3.3 Data Table Review

Below the chart, examine the detailed data table:

Week	Date	Predicted Sales
Week 1	2025-06-29	$1,234.56
Week 2	2025-07-06	-$567.89
Week 3	2025-07-13	$890.12
Week 4	2025-07-20	$345.67

Table 3.1: Example Forecast Data Table

3.4 Exploring Summary Statistics

3.4.1 Performance Metrics

Review the model performance information displayed:

- **Cumulative Sales Impact**: Total predicted change across all weeks
- **Growth Weeks**: Number of weeks with positive sales changes
- **Best/Worst Weeks**: Which weeks show strongest/weakest performance

Figure 3.7: Summary Statistics Display

3.4.2 Model Quality Indicator

The default model shows excellent performance:

- **Normalized WMAE**: 3.58% (Excellent - less than 5% error)
- **Quality Rating**: High confidence for business forecasting decisions

3.5 Downloading Your Results

3.5.1 Export Options

Save your forecast results for further analysis:

1. Scroll to the **Download Results** section
2. Choose your preferred format:
   - **CSV Format**: For Excel or data analysis tools
   - **JSON Format**: For programming or API integration
3. Click the corresponding download button
4. Save the file to your desired location

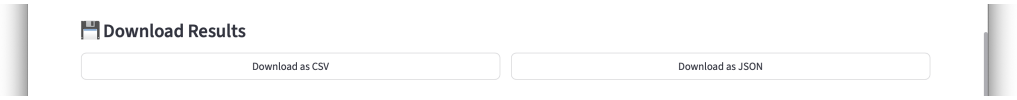


Figure 3.8: Forecast Results Download Options

3.6 Next Steps After Your First Forecast

3.6.1 Immediate Actions

After completing your first forecast:

1. **Analyze Results**: Review the forecast values and trends
2. **Save Data**: Download results in your preferred format
3. **Share Insights**: Export charts for presentations or reports

3.6.2 Advanced Exploration

Once comfortable with basic forecasting:

- **Try Different Models**: If you have your own model .pkl (Auto Arima / Exponential Smoothing) explore custom models
- **Train Custom Models**: Use the Training Application for specific datasets
- **Upload Your Models**: Use personally trained models for prediction

3.7 Common First-Time Questions

3.7.1 Understanding Predictions

Q: Why are some values negative?
A: Negative values indicate sales decreases from the previous week, which is normal in retail patterns.

Q: How accurate are these predictions?
A: The default model has a 3.58% normalized WMAE, indicating excellent accuracy for business forecasting.

Q: Can I predict further than 4 weeks?
A: The current system is optimized for 4-week forecasts. Longer periods may require model retraining and code update.

3.7.2 Technical Questions

Q: Do I need to install anything?
A: No, the cloud version requires only a web browser. Local installation is optional for advanced users.

Q: Can I use my own data?
A: Yes, use the Training Application to train models with your dataset, then upload them to the Prediction Application.

Q: What if the application stops working?
A: Refresh the browser page. Cloud applications may timeout after extended idle periods.

3.8 Troubleshooting First Steps

3.8.1 Application Loading Issues

Problem: Application takes too long to load or shows errors.
Solution:

- Check internet connection stability
- Try refreshing the browser page
- Clear browser cache if problems persist
- Use a different browser (Chrome recommended)

3.8.2 Model Loading Failures

Problem: Default model fails to load with error messages.
Solution:

- Refresh the application completely
- Try again after a few minutes (server may be busy)
- Check if you're using a supported browser

3.8.3 Prediction Generation Errors

Problem: Forecast generation fails or returns errors.
Solution:

- Ensure the model is properly loaded (green success message)
- Refresh the application and retry
- Check that you clicked the correct forecast button

3.9 Building Confidence with Practice

3.9.1 Recommended Practice Steps

To build familiarity with the system:

1. **Generate Multiple Forecasts**: Run several predictions to see consistent results
2. **Explore Interface Features**: Click through different tabs and options
3. **Download Results**: Practice exporting data in different formats
4. **Review Documentation**: Read detailed interface descriptions in Chapter 4

3.9.2 Learning Path

Progress through these stages:

- **Beginner**: Use default model for basic forecasting
- **Intermediate**: Explore custom model uploads and different formats
- **Advanced**: Train your own models using the Training Application

3.10 Summary

This first steps guide demonstrated:

- How to access the cloud-based Prediction Application
- Loading and using the default high-performance model
- Generating your first 4-week sales forecast
- Interpreting week-over-week sales change predictions
- Downloading results for further analysis
- Common troubleshooting for first-time users

You are now ready to explore the advanced features covered in subsequent chapters or begin using the system for regular forecasting activities.

4 GUI and User Interface Documentation

4.1 Interface Design Philosophy

The Walmart Sales Forecasting System employs a user-centric design approach built on the Streamlit framework. Both applications feature intuitive, web-based interfaces designed to minimize the learning curve while providing comprehensive functionality for time series forecasting.

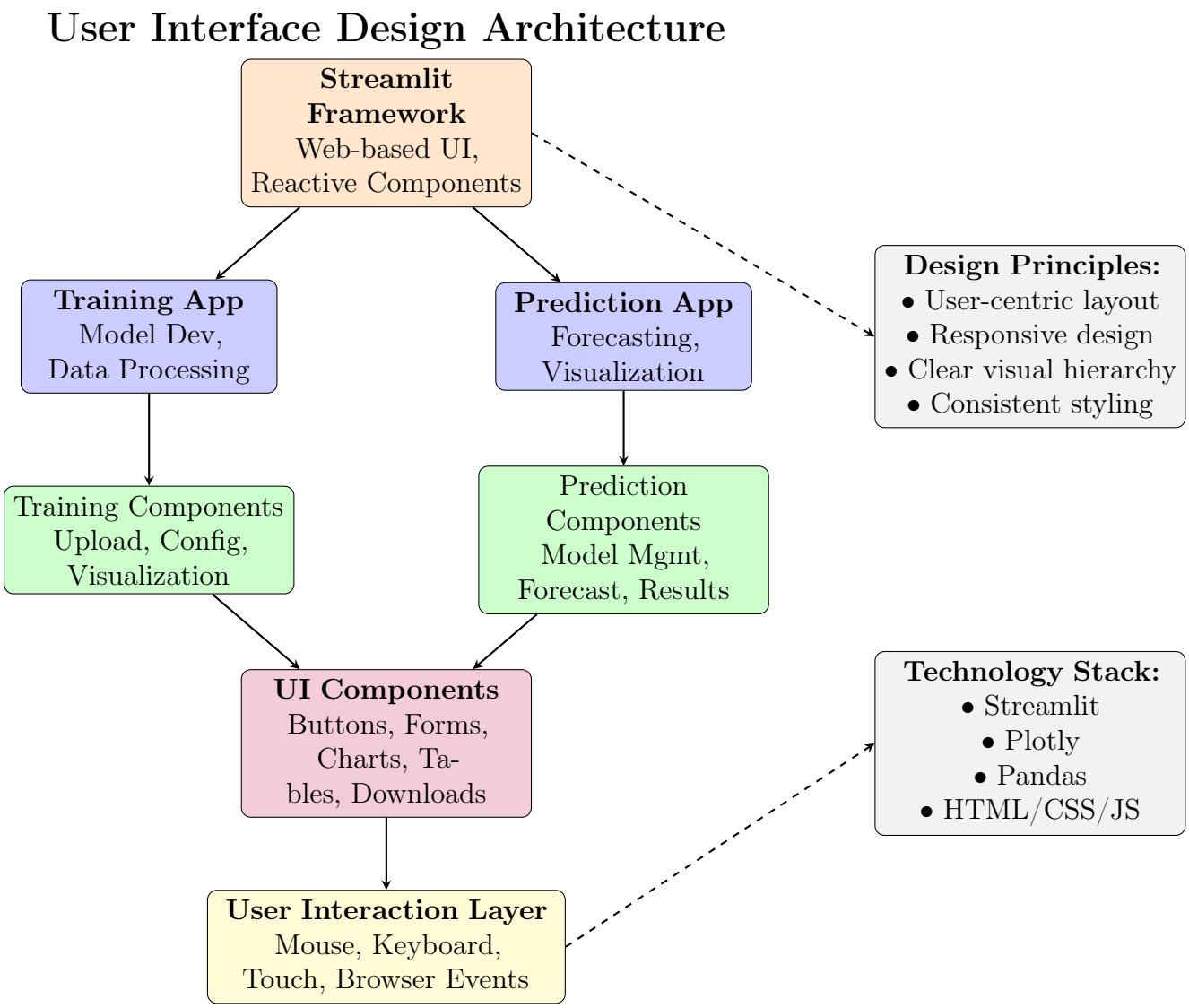


Figure 4.1: User Interface Design Architecture

4.2 Training Application Interface

4.2.1 Main Navigation Structure

The Training Application organizes functionality into logical sections with clear visual hierarchy:

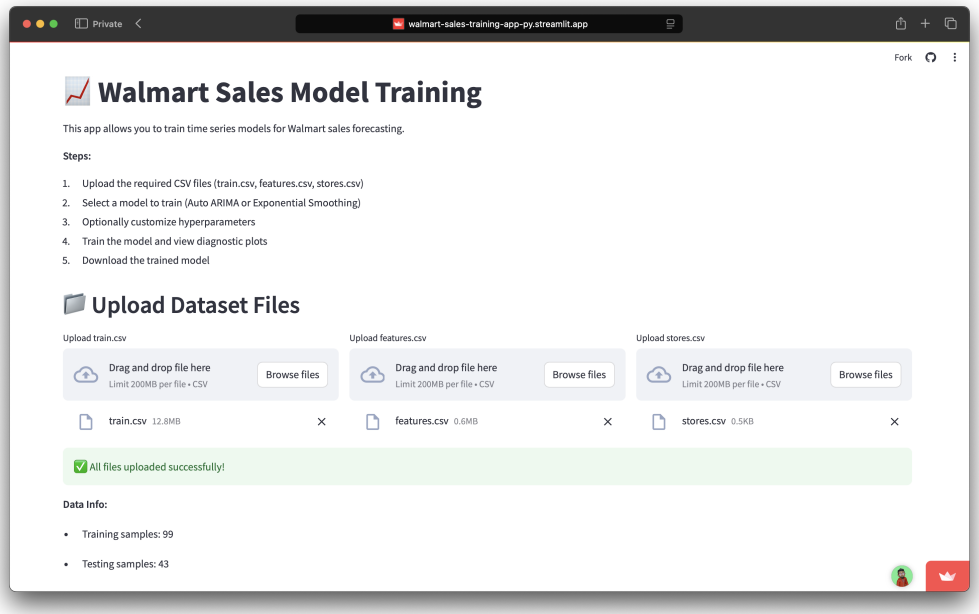


Figure 4.2: Training Application Complete Interface Overview

4.2.2 Header Section

Application Title and Description

- **Main Title**: " Walmart Sales Model Training"
- **Subtitle**: Comprehensive description of training capabilities
- **Step-by-Step Guide**: Visual workflow overview for new users

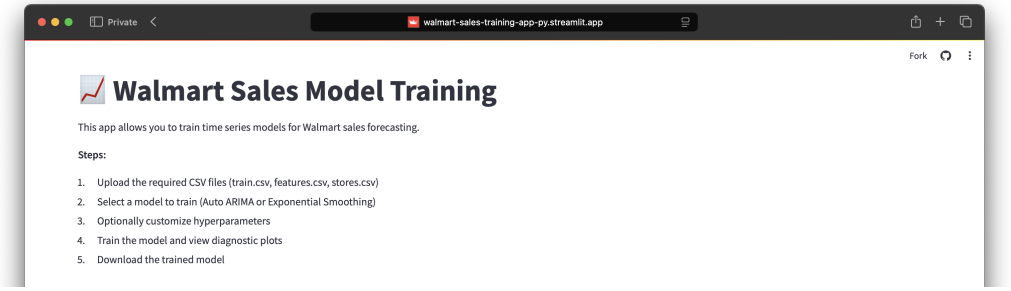


Figure 4.3: Training Application Header and Introduction

4.2.3 File Upload Section

Multi-File Upload Interface

The data upload section uses a three-column layout for required CSV files:

- **Column 1**: train.csv uploader with drag-and-drop support
- **Column 2**: features.csv uploader with file validation
- **Column 3**: stores.csv uploader with format checking

Visual Feedback

- **Success Indicator**: Green checkmark and " All files uploaded successfully!"
- **Progress Display**: Real-time upload progress bars
- **Error Messages**: Clear warnings for invalid file formats

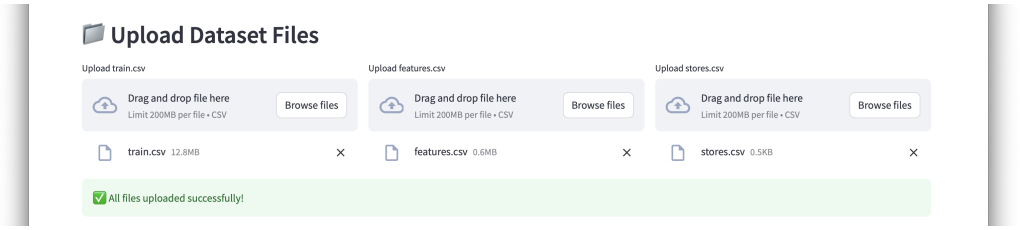


Figure 4.4: File Upload Interface with Multi-Column Layout

4.2.4 Model Selection Interface

Algorithm Choice Dropdown

The model selection uses an intuitive dropdown with clear descriptions:

- **Auto ARIMA**: Automatic parameter selection with seasonal components
- **Exponential Smoothing (Holt-Winters)**: Triple exponential smoothing

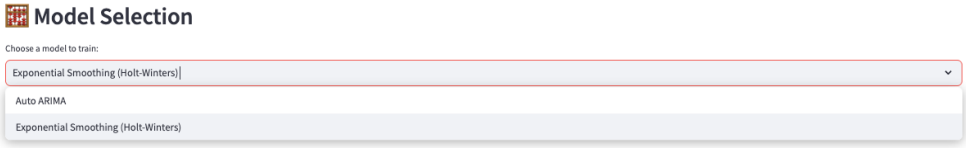


Figure 4.5: Model Selection Dropdown Interface

4.2.5 Hyperparameter Configuration

Dynamic Parameter Interface

The hyperparameter section adapts based on selected model type:

Auto ARIMA Parameters

- **Left Column**: Basic ARIMA parameters (start_p, start_q, max_p, max_q)
- **Right Column**: Seasonal parameters (start_P, start_Q, max_P, max_Q)
- **Input Validation**: Numeric inputs with min/max constraints

Exponential Smoothing Parameters

- **Left Column**: Smoothing parameters (seasonal_periods, seasonal type)
- **Right Column**: Trend parameters (trend type, damped option)
- **Interactive Controls**: Dropdowns and checkboxes for easy configuration

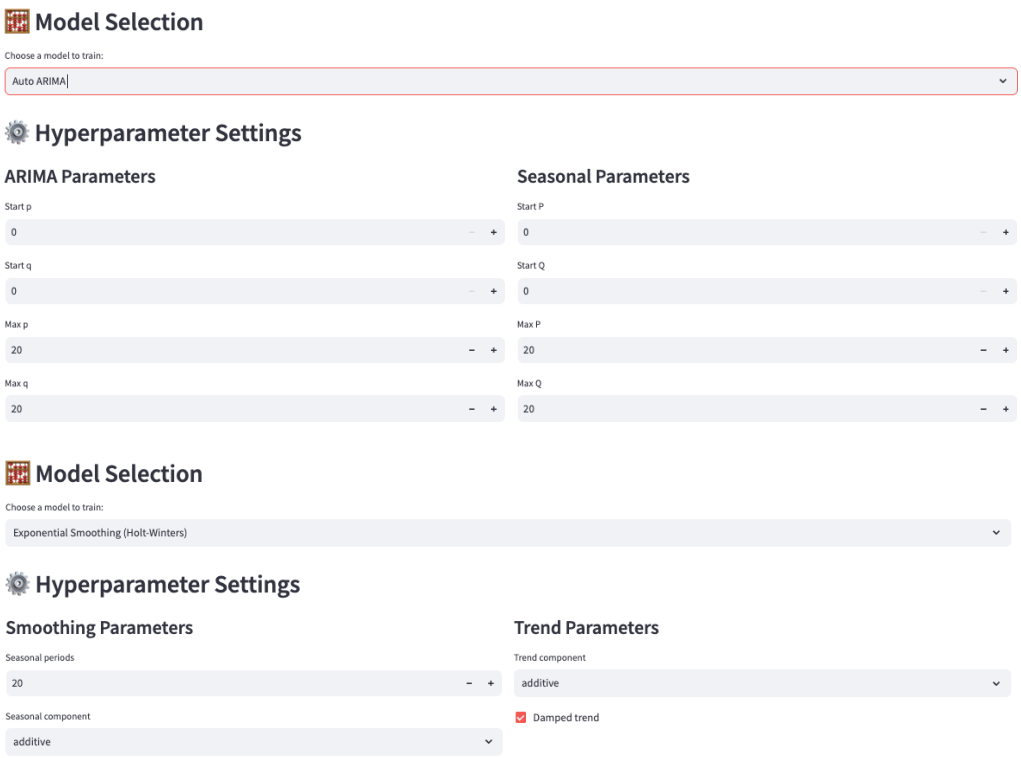


Figure 4.6: Dynamic Hyperparameter Configuration Interface

4.2.6 Training Execution Interface

Training Control and Progress

- **Primary Action Button**: Large "Start Training" button with distinctive styling

- **Progress Indicator**: Animated spinner with status text during training
- **Status Updates**: Real-time feedback on training completion

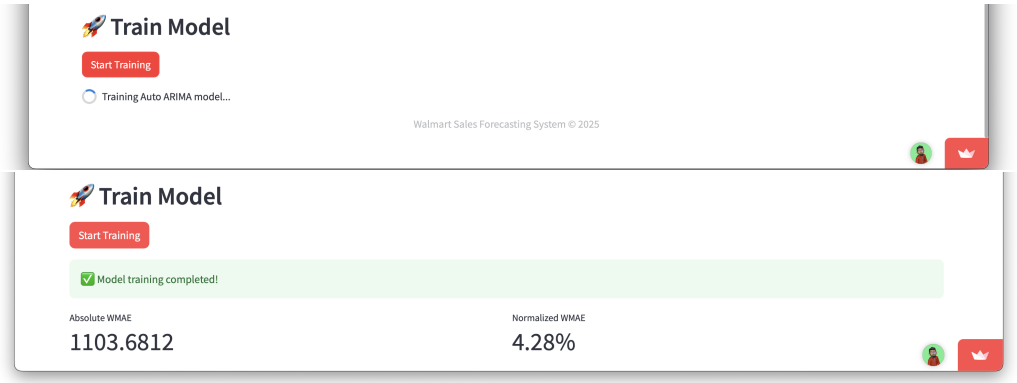


Figure 4.7: Training Execution and Progress Interface

4.2.7 Results Display Interface

Comprehensive Results Presentation

- **Performance Metrics**: WMAE scores with color-coded interpretation
- **Diagnostic Plots**: Interactive matplotlib charts with training/testing visualization
- **Model Download**: Direct download button for trained models
- **Performance Guide**: Interpretive text for WMAE scores

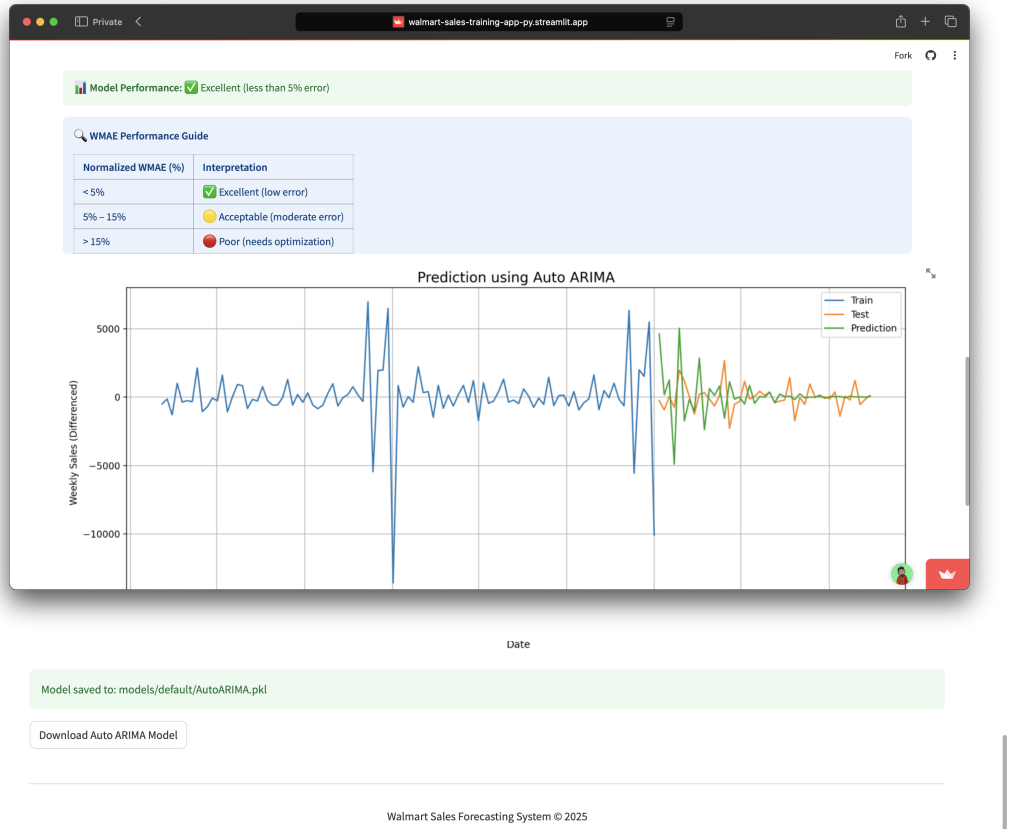


Figure 4.8: Training Results Display with Performance Metrics

4.3 Prediction Application Interface

4.3.1 Main Interface Layout

The Prediction Application emphasizes simplicity and immediate usability:

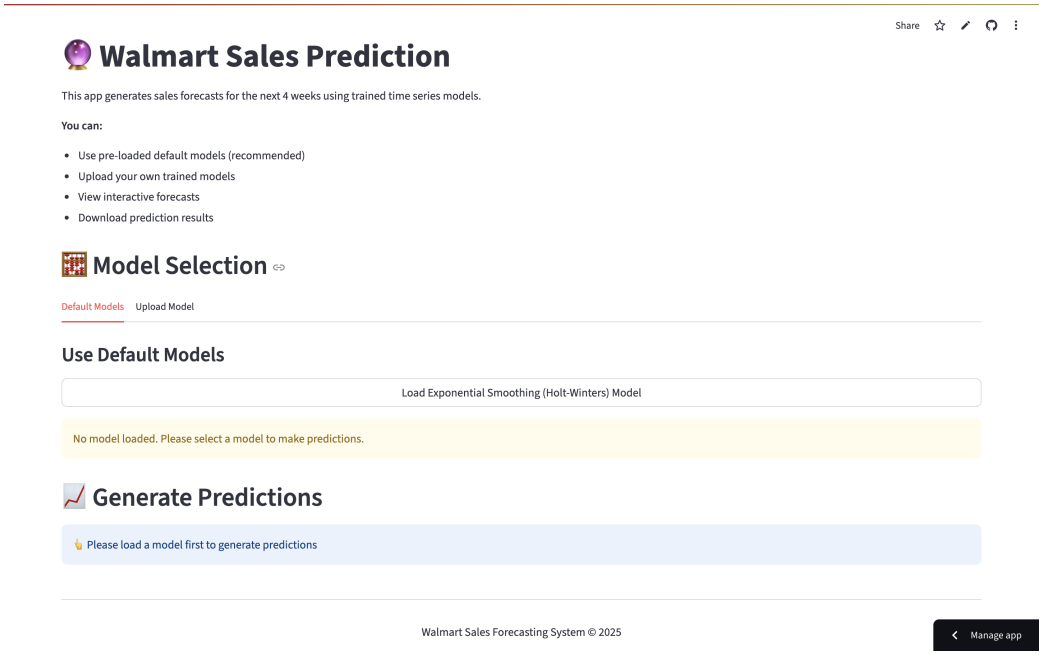


Figure 4.9: Prediction Application Complete Interface Overview

4.3.2 Application Header

Welcome and Introduction

- **Main Title**: " Walmart Sales Prediction" with distinctive icon
- **Feature Overview**: Bullet-pointed capability summary
- **Quick Start Guide**: Essential information for new users

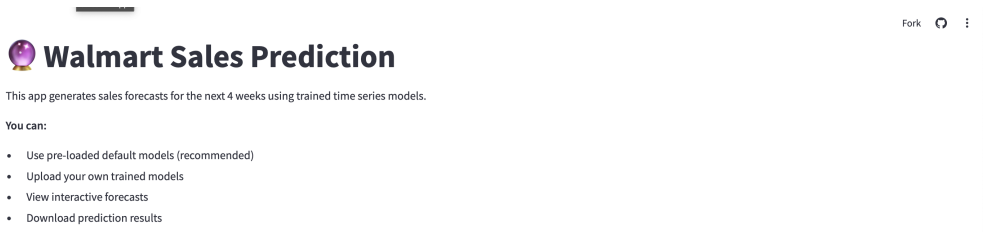


Figure 4.10: Prediction Application Header and Welcome Section

4.3.3 Model Selection Tabs

Tabbed Interface for Model Sources

The model selection uses clean tabs for different model sources:

Default Models Tab

- **Pre-loaded Options**: Exponential Smoothing (Holt-Winters) with performance metrics
- **Load Button**: Full-width button with clear loading action
- **Performance Display**: WMAE scores and accuracy ratings

Upload Model Tab

- **Model Type Selection**: Dropdown for algorithm specification
- **File Uploader**: Drag-and-drop interface for .pkl files
- **Load Button**: Context-sensitive activation based on file selection

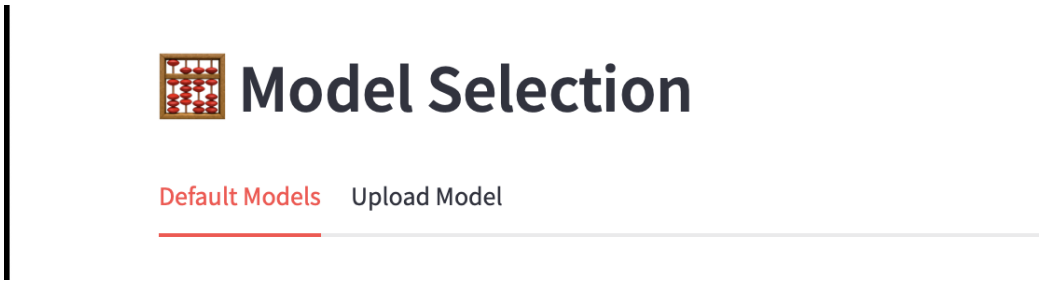


Figure 4.11: Model Selection Tabbed Interface

4.3.4 Current Model Status

Active Model Information Display

- **Model Info Box**: Colored information panel showing loaded model
- **Model Details**: Type, source (Default/Uploaded), and performance metrics
- **Status Indicators**: Visual confirmation of model readiness

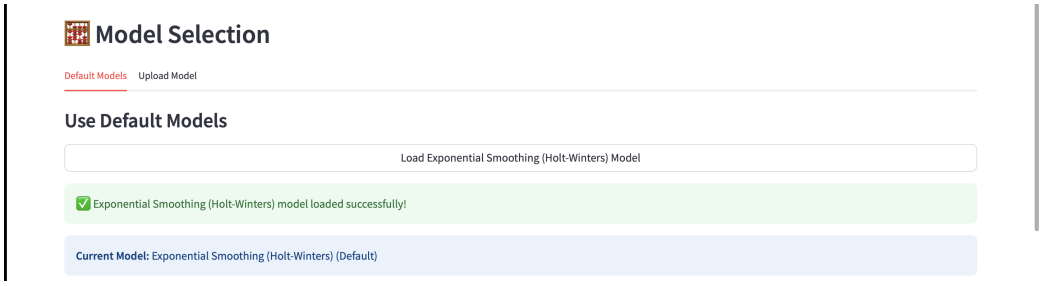


Figure 4.12: Current Model Status Display

4.3.5 Prediction Generation Interface

Forecast Control Section

- **Primary Action Button**: "Generate 4-Week Forecast" with primary styling
- **Processing Indicator**: Spinner animation during prediction generation
- **Conditional Display**: Only appears when model is properly loaded

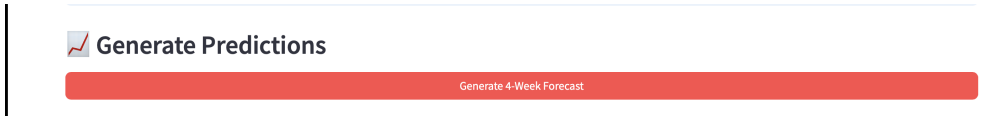


Figure 4.13: Prediction Generation Control Interface

4.3.6 Results Visualization Interface

Interactive Chart Display

The results section provides comprehensive visualization:

- **Plotly Interactive Chart**: Zoom, pan, and hover capabilities
- **Color-Coded Bars**: Green for positive changes, red for negative
- **Reference Line**: Zero-line for easy positive/negative identification
- **Chart Controls**: Built-in zoom, pan, and reset tools

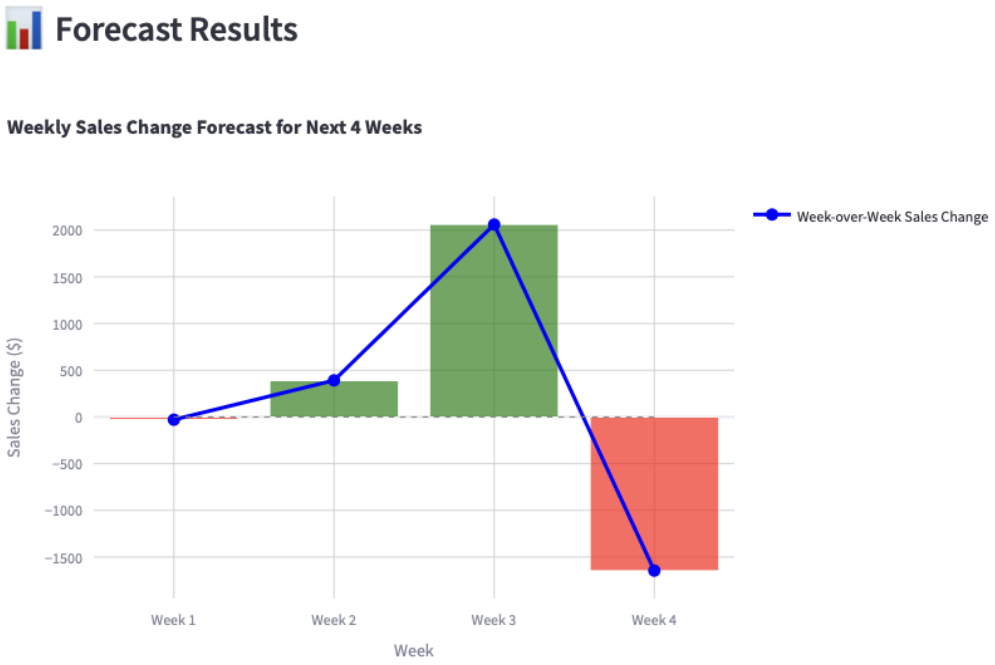


Figure 4.14: Interactive Results Visualization Interface

4.3.7 Data Table Interface

Formatted Results Table

- **Color-Coded Values**: Green for positive changes, red for negative
- **Formatted Currency**: Dollar signs and proper decimal formatting
- **Week Labels**: Clear week numbering and date display
- **Responsive Design**: Table adapts to different screen sizes

Figure 4.15: Formatted Data Table with Color Coding

4.3.8 Download and Export Interface

Multi-Format Export Options

- **Two-Column Layout**: CSV and JSON download buttons
- **File Naming**: Automatic naming with descriptive filenames

- **Format Icons**: Visual indicators for different file types
- **Full-Width Buttons**: Easy-to-click download controls

Figure 4.16: Download and Export Interface Options

4.4 Common Interface Elements

4.4.1 Navigation and Layout

Consistent Design Patterns
Both applications share common interface elements:

- **Sidebar Navigation**: Streamlit's built-in navigation structure
- **Section Headers**: Consistent typography and emoji icons
- **Action Buttons**: Uniform styling with primary/secondary distinction
- **Status Messages**: Color-coded success, warning, and error indicators

4.4.2 Responsive Design

Multi-Device Compatibility

- **Desktop Optimization**: Full-width layouts with multi-column sections
- **Tablet Adaptation**: Responsive column stacking and button sizing
- **Mobile Support**: Single-column layouts with touch-friendly controls

4.4.3 Accessibility Features

Inclusive Design Elements

- **Color Contrast**: High contrast ratios for text readability
- **Alternative Text**: Descriptive alt text for all images and charts
- **Keyboard Navigation**: Full keyboard accessibility for all controls
- **Screen Reader Support**: Semantic HTML structure for assistive technologies

4.5 Interactive Features

4.5.1 Real-Time Feedback

Dynamic User Experience

- **Instant Validation**: Real-time file format checking
- **Progress Indicators**: Visual feedback during long operations
- **Status Updates**: Immediate confirmation of user actions
- **Error Prevention**: Input validation before processing

4.5.2 Chart Interactivity

Advanced Visualization Controls

- **Hover Information**: Detailed data points on mouse over
- **Zoom Controls**: Mouse wheel and toolbar zoom functionality
- **Pan Navigation**: Click and drag chart movement
- **Reset View**: One-click return to default chart view

4.6 Interface Customization

4.6.1 Theme and Styling

Visual Customization Options

- **Light Theme**: Default high-contrast theme for most users
- **Dark Theme**: Available through browser/OS dark mode settings
- **Font Scaling**: Browser zoom support for accessibility
- **Color Preferences**: System-level color scheme integration

4.6.2 Layout Preferences

User-Controlled Interface Elements

- **Section Expansion**: Expandable sections for focused workflow
- **Tab Memory**: Interface remembers last selected tabs
- **Window Sizing**: Automatic adaptation to browser window size

4.7 Interface Best Practices

4.7.1 Optimal Usage Patterns

Recommended Workflow Navigation

- **Top-to-Bottom**: Follow the natural page flow for optimal experience
- **Tab Completion**: Complete all inputs in a tab before switching
- **Progressive Disclosure**: Use advanced features only after mastering basics
- **Regular Saves**: Download results frequently to prevent data loss

4.7.2 Performance Optimization

Interface Responsiveness Tips

- **Browser Choice**: Latest version of Chrome or Firefox recommended for best performance
- **Tab Management**: Avoid having too many browser tabs open
- **Cache Clearing**: Clear browser cache if interface becomes sluggish
- **Network Stability**: Ensure stable internet connection for cloud usage

This comprehensive interface documentation provides the foundation for effective system utilization. The next chapter will detail the specific functions and features available in each application.

5 Application Functions and Features

5.1 Comprehensive Feature Overview

The Walmart Sales Forecasting System provides a complete ecosystem for time series forecasting, encompassing model development, training, evaluation, and production deployment. This chapter details every feature and function available across both applications.

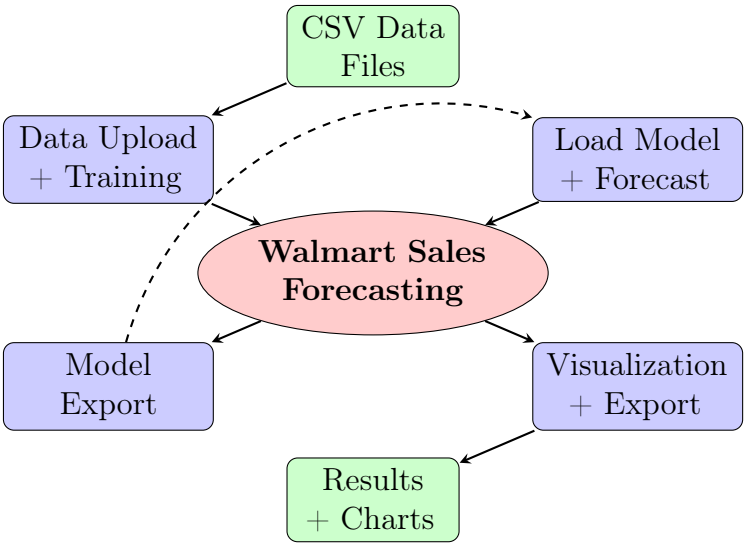


Figure 5.1: Complete Feature Ecosystem Overview

5.2 Training Application Functions

5.2.1 Data Management Features

Multi-File Data Upload

The Training Application supports comprehensive data ingestion:
Supported File Formats

- **CSV Files**: Primary format for all datasets
- **Encoding**: UTF-8 encoding support

- **Size Limits**: Up to 200MB per file (cloud), unlimited (local)
- **Validation**: Automatic schema validation upon upload

Required Dataset Structure

1. **train.csv**: Historical sales data with Store, Date, Weekly_Sales, IsHoliday
2. **features.csv**: External factors with Store, Date, Temperature, Fuel_Price, etc.
3. **stores.csv**: Store metadata with Store, Type, Size information

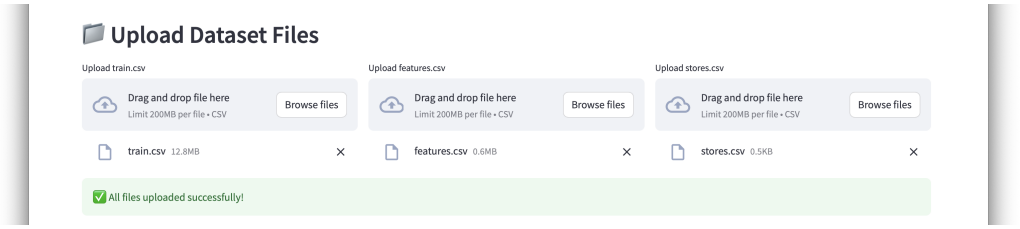


Figure 5.2: Multi-File Data Upload Interface

Automated Data Preprocessing

Data Cleaning Pipeline

- **Missing Value Imputation**: Automatic filling of missing markdown values with zeros
- **Outlier Detection**: Removal of negative or invalid sales records
- **Date Standardization**: Automatic conversion to standard date formats
- **Data Type Validation**: Ensures numeric columns contain valid numbers

Feature Engineering

- **Holiday Indicators**: Automatic creation of binary holiday features
- **Seasonal Components**: Detection and encoding of seasonal patterns
- **Trend Extraction**: Identification of long-term trends in data
- **Merge Operations**: Intelligent joining of train, features, and stores data

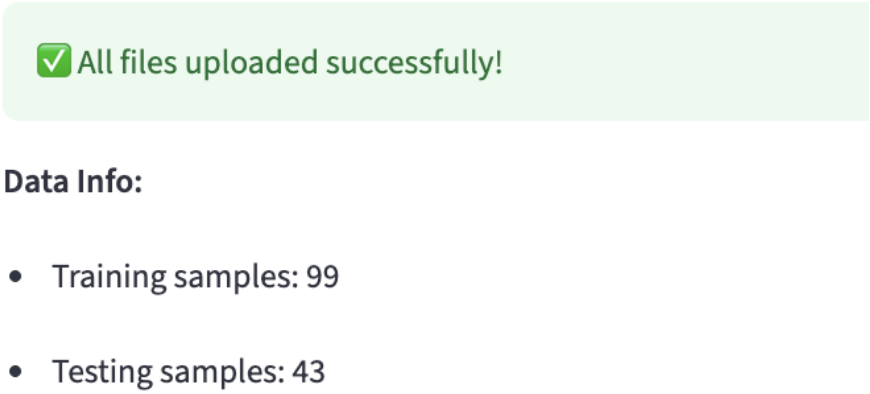


Figure 5.3: Data Preprocessing Pipeline Results

5.2.2 Model Training Functions

Auto ARIMA Training

Automated Parameter Selection

- **Grid Search**: Comprehensive search across parameter space
- **Information Criteria**: AIC-based model selection
- **Seasonal Detection**: Automatic identification of seasonal patterns
- **Differencing**: Automatic determination of differencing order

Configurable Hyperparameters

- **AR Terms**: start_p (0-10), max_p (1-30)
- **MA Terms**: start_q (0-10), max_q (1-30)
- **Seasonal AR**: start_P (0-10), max_P (1-30)
- **Seasonal MA**: start_Q (0-10), max_Q (1-30)

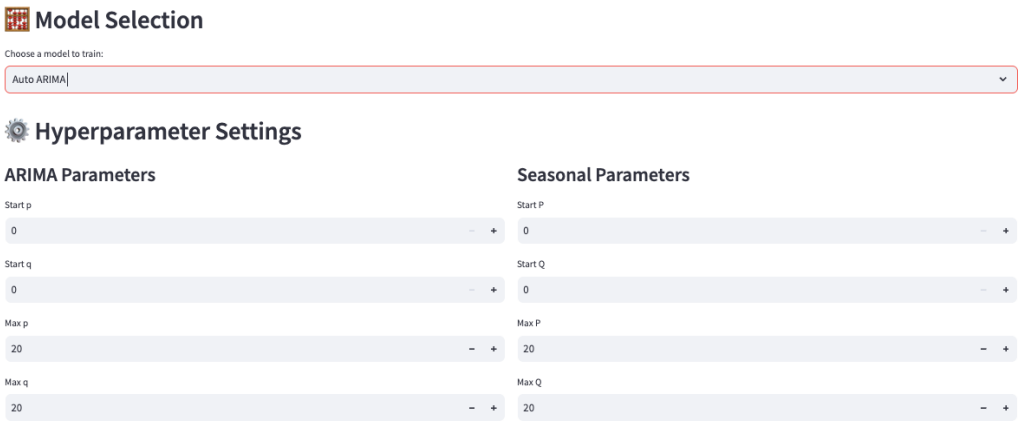


Figure 5.4: Auto ARIMA Training Configuration

Exponential Smoothing Training

Holt-Winters Implementation

- **Triple Smoothing**: Level, trend, and seasonal components
- **Additive/Multiplicative**: Choice of seasonal component type
- **Damped Trend**: Optional trend damping for stability
- **Optimization**: Automatic parameter optimization

Seasonal Configuration

- **Seasonal Periods**: Configurable cycle length (1-52 weeks)
- **Seasonal Type**: Additive or multiplicative seasonality
- **Trend Type**: Additive, multiplicative, or none
- **Initialization**: Automatic initialization of smoothing parameters

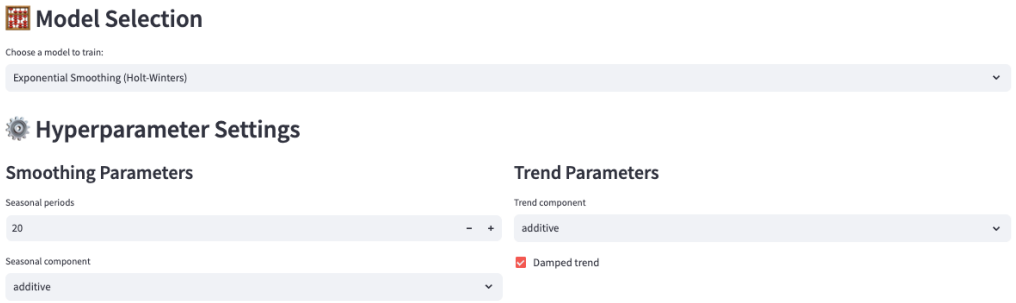


Figure 5.5: Exponential Smoothing Training Configuration

5.2.3 Model Evaluation Features

Performance Metrics

WMAE Calculation

- **Absolute WMAE**: Raw error metric in sales dollars
- **Normalized WMAE**: Percentage error relative to total sales
- **Interpretive Guidance**: Automatic performance category assignment
- **Benchmark Comparison**: Performance relative to industry standards

Performance Categories

- **Excellent**: $< 5\%$ normalized WMAE (Green indicator)
- **Acceptable**: 5-15% normalized WMAE (Yellow indicator)
- **Poor**: $> 15\%$ normalized WMAE (Red indicator)

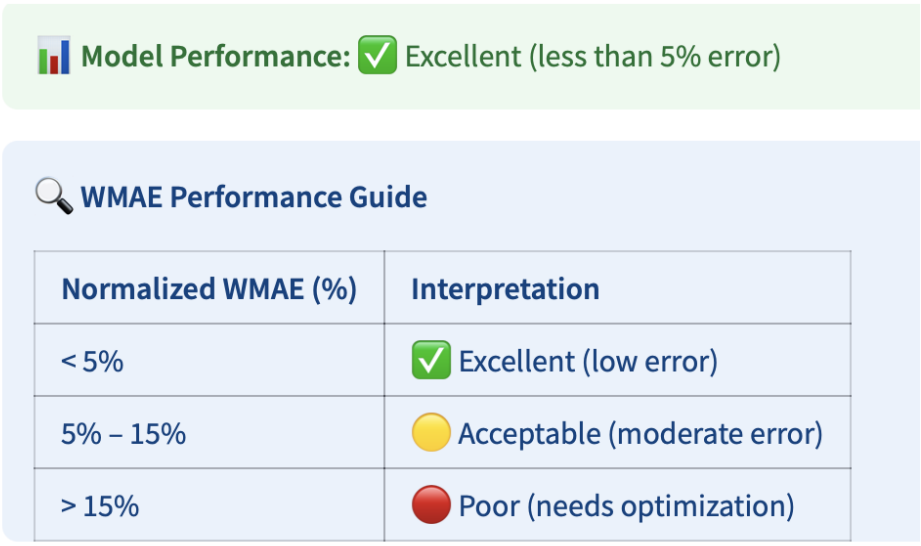


Figure 5.6: Model Performance Evaluation Display

Diagnostic Visualization

Training vs Testing Plots

- **Time Series Plot**: Historical data with train/test split visualization
- **Prediction Overlay**: Model predictions overlaid on actual test data

- **Residual Analysis**: Visual assessment of prediction errors
- **Interactive Charts**: Zoom, pan, and hover functionality

Model Diagnostics

- **Fit Quality**: Visual assessment of model fit to training data
- **Prediction Accuracy**: Comparison of predictions to actual values
- **Trend Capture**: Evaluation of trend and seasonal pattern capture
- **Error Distribution**: Analysis of prediction error patterns

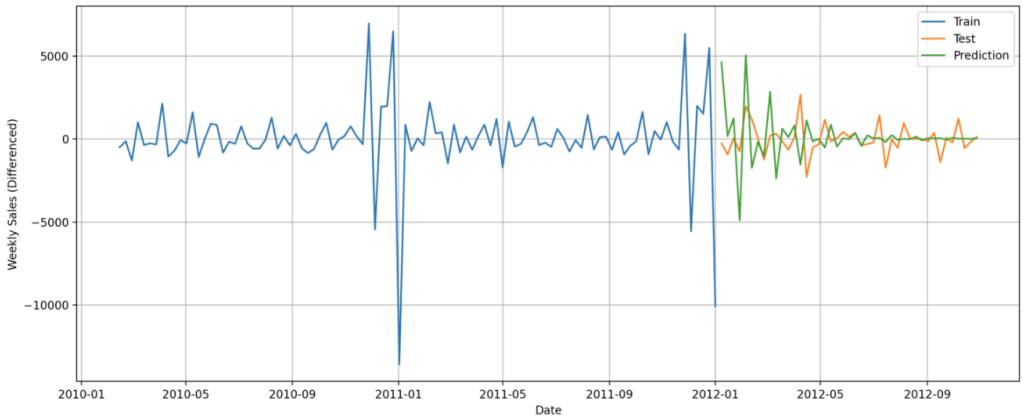


Figure 5.7: Comprehensive Diagnostic Visualization

5.2.4 Model Export and Management

Model Serialization

Export Formats

- **Primary Format**: Joblib .pkl files for cross-platform compatibility
- **Backup Format**: Pickle serialization as fallback option
- **Compression**: Efficient model compression for storage
- **Metadata**: Model type and performance information embedded

File Management

- **Automatic Naming**: Descriptive filenames based on model type
- **Local Storage**: Models saved to models/default/ directory
- **Download Option**: Direct download for external use

- **Version Control**: Timestamp-based version management

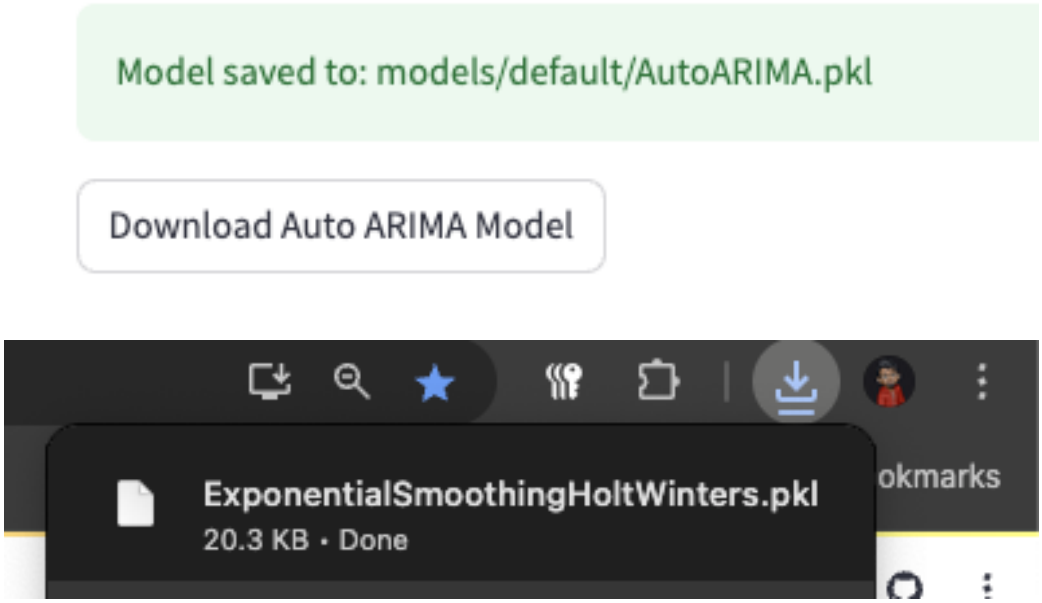


Figure 5.8: Model Export and Download Interface

5.3 Prediction Application Functions

5.3.1 Model Loading Features

Default Model Access

Pre-trained Model Library

- **Exponential Smoothing**: High-performance default model (3.58% WMAE)
- **Performance Verification**: Pre-validated on standard datasets
- **Immediate Access**: One-click loading without configuration
- **Reliability**: Thoroughly tested and optimized parameters

Model Specifications

- **Algorithm**: Holt-Winters Triple Exponential Smoothing
- **Training Data**: Comprehensive Walmart sales dataset
- **Seasonal Periods**: 20-week seasonal cycle optimization
- **Performance**: 3.58% normalized WMAE (excellent accuracy)

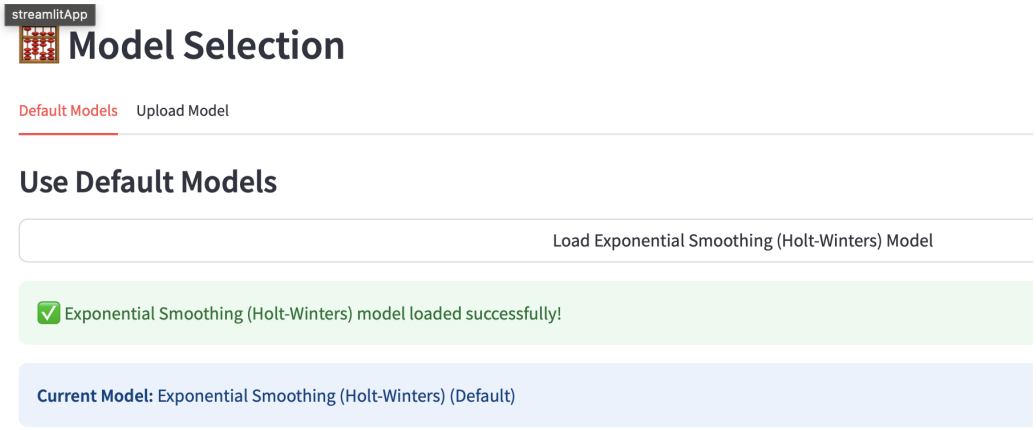


Figure 5.9: Default Model Loading Interface

Custom Model Upload

Model Upload Capabilities

- **File Support**: .pkl files from Training Application or external sources
- **Model Types**: Auto ARIMA and Exponential Smoothing support
- **Validation**: Automatic model format and compatibility checking
- **Error Handling**: Clear error messages for invalid models

Upload Process

1. Select model type from dropdown (Auto ARIMA or Exponential Smoothing)
2. Upload .pkl file using drag-and-drop or file browser
3. Click "Load Uploaded Model" to validate and activate
4. Receive confirmation message upon successful loading

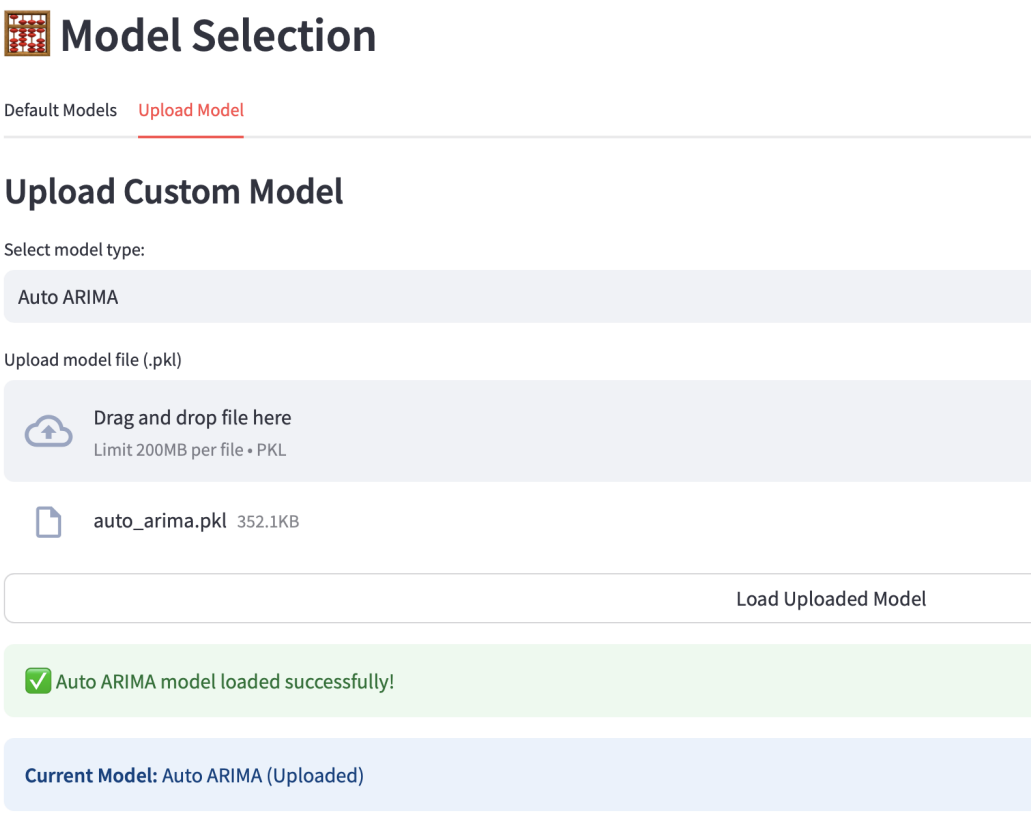


Figure 5.10: Custom Model Upload Interface

5.3.2 Forecasting Functions

Prediction Generation

4-Week Forecast Engine

- **Forecast Horizon**: Fixed 4-week prediction period
- **Output Type**: Week-over-week sales changes (not absolute values)
- **Processing Speed**: Sub-second prediction generation
- **Consistency**: Reproducible results for same model and date

Prediction Algorithm Routing

- **Auto ARIMA**: Uses .predict() method with n_periods parameter
- **Exponential Smoothing**: Uses .forecast() method with step count
- **Date Generation**: Automatic next-4-weeks date calculation
- **Error Handling**: Graceful failure with descriptive error messages

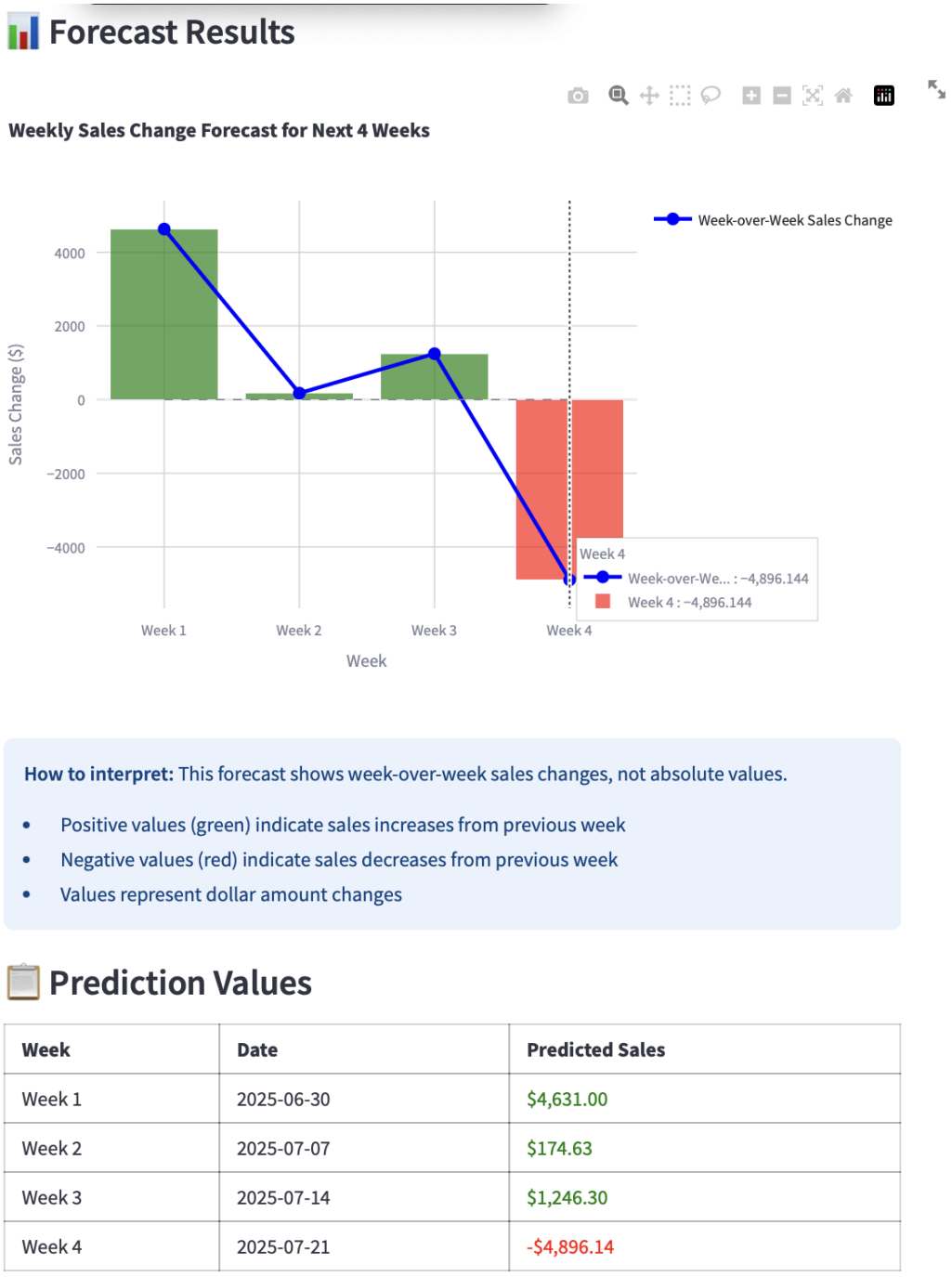


Figure 5.11: Prediction Generation Process Interface

Results Interpretation

Sales Change Analysis

- **Positive Values**: Green indicators for sales increases
- **Negative Values**: Red indicators for sales decreases

- **Magnitude Interpretation**: Dollar amounts represent change size
- **Baseline Reference**: Zero line for positive/negative comparison

Business Context

- **Week-over-Week Changes**: Comparison to previous week's performance
- **Seasonal Patterns**: Recognition of cyclical business patterns
- **Trend Analysis**: Long-term direction assessment
- **Decision Support**: Actionable insights for business planning

5.3.3 Visualization Features

Interactive Chart Display

Plotly-Based Visualization

- **Interactive Elements**: Hover tooltips with detailed information
- **Zoom Controls**: Mouse wheel and toolbar zoom functionality
- **Pan Navigation**: Click-and-drag chart movement
- **Reset View**: One-click return to default view

Visual Design Elements

- **Color Coding**: Green bars for positive changes, red for negative
- **Reference Line**: Horizontal zero line for baseline comparison
- **Grid Lines**: Background grid for value estimation
- **Professional Styling**: Clean, business-appropriate chart design

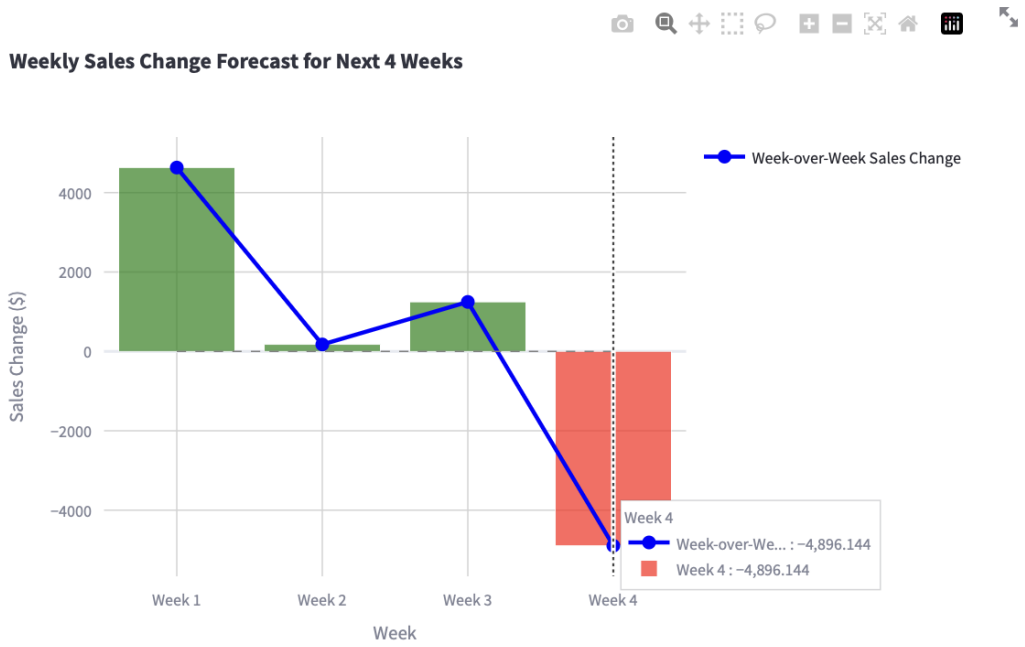


Figure 5.12: Interactive Chart Visualization with Controls

Data Table Presentation

Formatted Results Display

- **Color-Coded Values**: Green text for positive, red for negative changes
- **Currency Formatting**: Proper dollar signs and decimal precision
- **Date Display**: Clear date formatting (YYYY-MM-DD)
- **Week Labels**: Sequential week numbering for easy reference

Table Features

- **Responsive Design**: Adapts to different screen sizes
- **Clean Layout**: Minimal design focusing on data clarity
- **Easy Scanning**: Clear column headers and row delineation
- **Copy-Friendly**: Table format suitable for copying to other applications

📋 Prediction Values

Week	Date	Predicted Sales
Week 1	2025-06-30	$4,631.00
Week 2	2025-07-07	$174.63
Week 3	2025-07-14	$1,246.30
Week 4	2025-07-21	-$4,896.14

Figure 5.13: Formatted Data Table with Color Coding

5.3.4 Export and Analysis Features

Multi-Format Export

CSV Export

- **Standard Format**: Comma-separated values for universal compatibility
- **Excel Integration**: Direct opening in Microsoft Excel
- **Data Analysis**: Compatible with Python pandas, R, and other tools
- **Automatic Naming**: Descriptive filename with timestamp

JSON Export

- **API Integration**: Machine-readable format for applications
- **Structured Data**: Nested format preserving data relationships
- **Programming Integration**: Easy import into programming environments
- **Web Applications**: Direct integration with web services

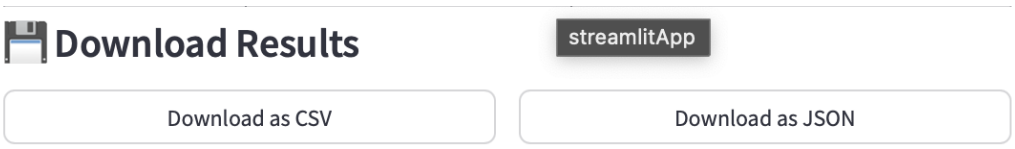


Figure 5.14: Multi-Format Export Options Interface

Summary Statistics

Performance Metrics Display

- **Cumulative Impact**: Total sales change across all predicted weeks
- **Growth Week Count**: Number of weeks with positive sales changes
- **Best/Worst Weeks**: Identification of strongest and weakest periods
- **Trend Direction**: Overall forecast trend assessment

Business Intelligence

- **Quick Insights**: At-a-glance performance summary
- **Decision Metrics**: Key indicators for business planning
- **Comparative Analysis**: Week-to-week performance comparison
- **Risk Assessment**: Identification of potential decline periods

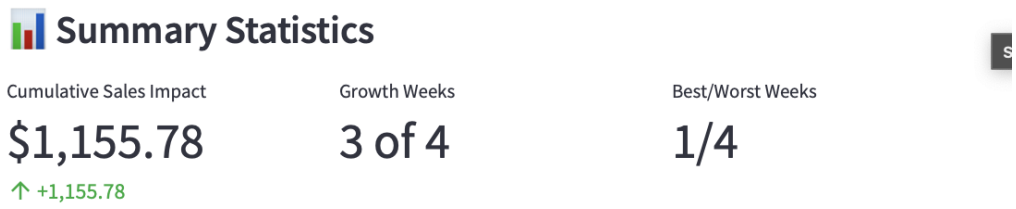


Figure 5.15: Summary Statistics and Business Intelligence Display

5.4 Cross-Application Integration

5.4.1 Model Transfer Workflow

Training to Prediction Pipeline

1. Train model in Training Application with custom dataset
2. Download trained model in .pkl format
3. Upload downloaded model to Prediction Application
4. Generate forecasts using custom-trained model

File Management

- **Local Installation**: Direct file copy between applications

- **Cloud Deployment**: Manual download/upload process
- **Format Consistency**: Unified .pkl format across applications
- **Compatibility Checking**: Automatic validation of model compatibility

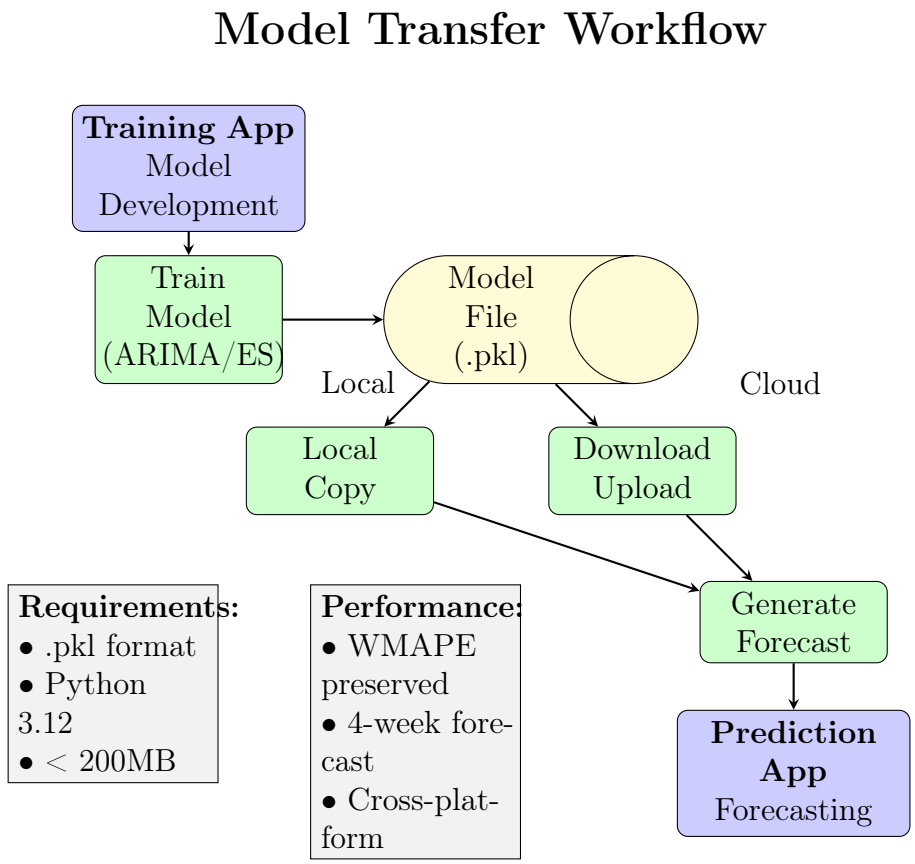


Figure 5.16: Model Transfer Workflow Between Applications

5.4.2 Workflow Optimization

Efficient Usage Patterns

- **Development Cycle**: Train → Evaluate → Export → Deploy → Predict
- **Iterative Improvement**: Compare multiple models before deployment
- **Performance Monitoring**: Regular retraining with new data
- **Version Management**: Maintain multiple model versions for comparison

Best Practices

- **Model Documentation**: Record hyperparameters and performance metrics

- **Regular Updates**: Retrain models with fresh data periodically
- **A/B Testing**: Compare different models on same prediction tasks
- **Backup Strategy**: Maintain copies of well-performing models

5.5 Advanced Features

5.5.1 Error Handling and Validation

Input Validation

- **File Format Checking**: Automatic validation of uploaded files
- **Data Schema Validation**: Verification of required columns and data types
- **Model Compatibility**: Checking model format and algorithm compatibility
- **Parameter Validation**: Range checking for hyperparameter inputs

Error Recovery

- **Graceful Degradation**: Continued operation despite minor errors
- **Clear Error Messages**: Descriptive explanations of problems
- **Recovery Suggestions**: Specific steps to resolve issues
- **Fallback Options**: Alternative approaches when primary methods fail

5.5.2 Performance Optimization

Computational Efficiency

- **Caching**: Intelligent caching of intermediate results
- **Lazy Loading**: Loading data and models only when needed
- **Memory Management**: Efficient memory usage for large datasets
- **Processing Optimization**: Streamlined algorithms for faster execution

User Experience Enhancement

- **Progress Indicators**: Real-time feedback during long operations
- **Asynchronous Processing**: Non-blocking user interface during computation
- **Session Management**: Maintaining state across user interactions
- **Responsive Design**: Adaptive interface for different devices

5.6 Feature Comparison Matrix

Feature	Training App	Prediction App
Data Upload	Yes (Multi-file)	No
Model Training	Yes (ARIMA, ES)	No
Custom Models	Yes (Export)	Yes (Import)
Default Models	No	Yes
Hyperparameter Tuning	Yes	No
Performance Evaluation	Yes (WMAE)	No
Forecasting	No	Yes (4-week)
Visualization	Yes (Diagnostics)	Yes (Interactive)
Export Options	Yes (Models)	Yes (Results)
Cloud Access	Yes	Yes
Local Installation	Yes	Yes

Table 5.1: Comprehensive Feature Comparison Between Applications

This comprehensive feature overview demonstrates the system's capabilities for complete time series forecasting workflows. The next chapter will detail the technical specifications and requirements for optimal system performance.

6 System Specifications

6.1 Technical Requirements Overview

The Walmart Sales Forecasting System is designed to operate across multiple deployment environments while maintaining consistent performance and functionality. This chapter provides comprehensive technical specifications for both local installation and cloud deployment scenarios.

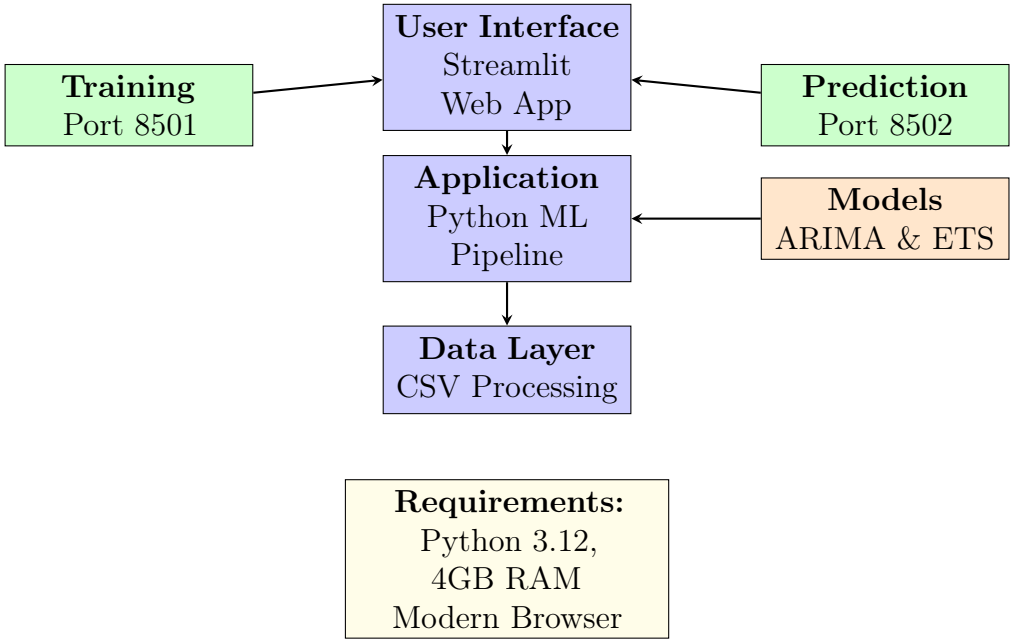


Figure 6.1: System Technical Architecture and Requirements

6.2 Local Installation Requirements

6.2.1 Core System Requirements

Python Environment

- **Python Version**: Exactly Python 3.12.x (3.12.0, 3.12.1, 3.12.2, etc.)
- **Version Restriction**: NOT compatible with Python 3.13+ or Python 3.11 and below

- **Architecture**: 64-bit Python installation required

- **Installation Source**: Official Python.org distribution recommended

Operating System Support

- **Windows**: Windows 10 (version 1903+) or Windows 11

- **macOS**: macOS 10.15 (Catalina) or newer

- **Linux**: Ubuntu 18.04+, CentOS 7+, or equivalent distributions

- **Architecture**: x86_64 (AMD64) processor architecture

Component	Minimum	Recommended
CPU	Dual-core 2.0 GHz	Quad-core 3.0 GHz+
RAM	4 GB	8 GB or more
Storage	2 GB free space	5 GB free space
Network	1 Mbps	10 Mbps+

Table 6.1: Hardware Requirements for Local Installation

6.2.2 Software Dependencies

Required Python Packages

The system depends on specific package versions for stability and compatibility:

Package	Version	Purpose
streamlit	1.32.0	Web application framework
pandas	2.2.2	Data manipulation and analysis
numpy	1.26.4	Numerical computing foundation
matplotlib	3.8.4	Static plotting and visualization
seaborn	0.13.2	Statistical data visualization
plotly	5.24.1	Interactive web-based plotting
joblib	1.4.2	Model serialization and loading
statsmodels	0.14.2	Statistical modeling library
pmdarima	2.0.4	Auto ARIMA implementation
pytest	7.4.4	Testing framework for validation

Table 6.2: Required Python Package Dependencies

Virtual Environment Configuration

```
# Virtual environment creation
python3.12 -m venv walmart_forecast_env

# Environment activation (Linux/macOS)
source walmart_forecast_env/bin/activate

# Environment activation (Windows)
walmart_forecast_env\Scripts\activate

# Package installation
pip install -r requirements.txt
```

6.2.3 Port and Network Configuration

Default Port Assignments

- **Training Application**: Port 8501 (Streamlit default)
- **Prediction Application**: Port 8502 (manually configured)

- **Protocol**: HTTP (local development only)
- **Firewall**: No special firewall configuration required for local use

Network Requirements

- **Internet Access**: Required for initial package installation
- **Bandwidth**: Minimal during operation (interface assets only)
- **Latency**: Not critical for local deployment
- **Proxy**: Compatible with corporate proxy configurations

6.3 Cloud Deployment Specifications

6.3.1 Streamlit Cloud Infrastructure

Platform Specifications

- **Hosting Provider**: Streamlit Community Cloud
- **Runtime Environment**: Containerized Python 3.12 environment
- **Resource Allocation**: Shared compute resources with automatic scaling
- **Geographic Distribution**: Global CDN for optimized access speeds

Performance Characteristics

- **CPU**: Shared vCPU with fair-use allocation
- **Memory**: 1GB RAM per application instance
- **Storage**: Ephemeral file system with session-based persistence
- **Bandwidth**: Unlimited for normal usage patterns

Metric	Training App	Prediction App
Cold Start Time	15-45 seconds	10-30 seconds
Warm Response Time	$<$ 2 seconds	$<$ 1 second
Session Timeout	30 minutes idle	30 minutes idle
File Upload Limit	200 MB	200 MB
Concurrent Users	50+ supported	100+ supported

Table 6.3: Cloud Deployment Performance Specifications

6.3.2 Browser Compatibility

Supported Browsers

- **Google Chrome**: Version 90+ (Recommended)
- **Mozilla Firefox**: Version 88+
- **Microsoft Edge**: Version 90+
- **Safari**: Version 14+ (macOS/iOS)

Browser Feature Requirements

- **JavaScript**: ES6+ support required
- **WebSockets**: For real-time application updates
- **Local Storage**: For session state management
- **File API**: For drag-and-drop file uploads

6.4 Data Format Specifications

6.4.1 Input Data Requirements

CSV File Structure

The system expects three specific CSV files with predefined schemas:

train.csv Requirements

- **Required Columns**: Store, Date, Weekly_Sales, IsHoliday
- **Store Column**: Integer values (1, 2, 3, ...)
- **Date Column**: YYYY-MM-DD format (e.g., 2010-02-05)
- **Weekly_Sales**: Float values representing sales in dollars
- **IsHoliday**: Boolean values (True/False or 1/0)

features.csv Requirements

- **Required Columns**: Store, Date, Temperature, Fuel_Price, MarkDown1-5, CPI, Unemployment, IsHoliday
- **Store/Date**: Same format as train.csv for proper merging
- **Temperature**: Float values in Fahrenheit

- **Fuel_Price**: Float values in dollars per gallon

- **MarkDown1-5**: Float values (may contain NaN for missing markdowns)

- **CPI/Unemployment**: Float values for economic indicators

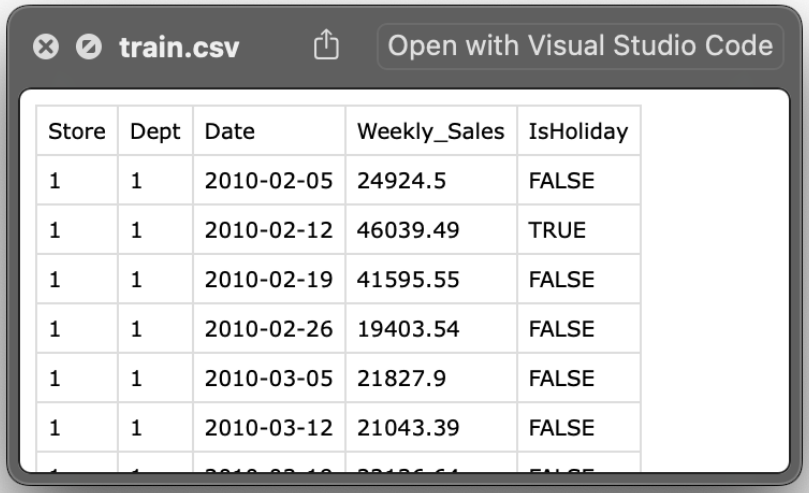
stores.csv Requirements

- **Required Columns**: Store, Type, Size

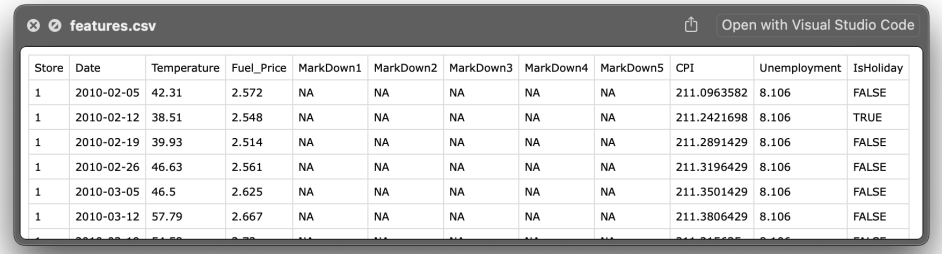
- **Store**: Integer values matching train.csv and features.csv

- **Type**: String values (A, B, or C)

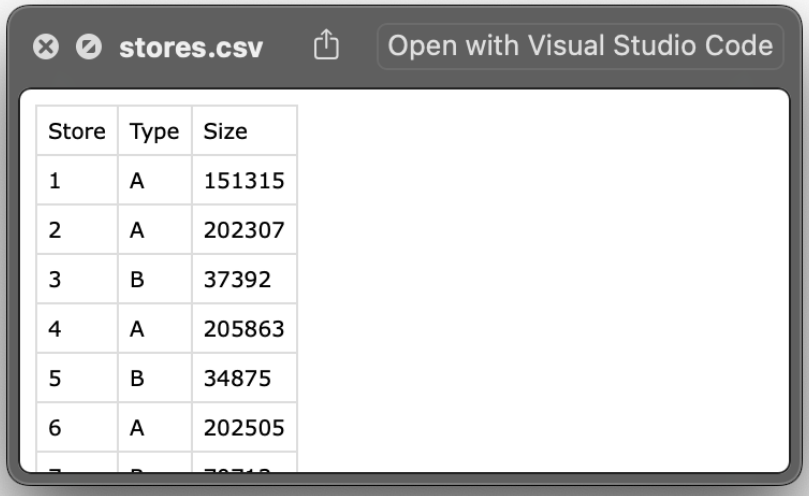
- **Size**: Integer values representing store square footage



Store	Dept	Date	Weekly_Sales	IsHoliday
1	1	2010-02-05	24924.5	FALSE
1	1	2010-02-12	46039.49	TRUE
1	1	2010-02-19	41595.55	FALSE
1	1	2010-02-26	19403.54	FALSE
1	1	2010-03-05	21827.9	FALSE
1	1	2010-03-12	21043.39	FALSE



Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
1	2010-02-05	42.31	2.572	NA	NA	NA	NA	NA	211.0963582	8.106	FALSE
1	2010-02-12	38.51	2.548	NA	NA	NA	NA	NA	211.2421698	8.106	TRUE
1	2010-02-19	39.93	2.514	NA	NA	NA	NA	NA	211.2891429	8.106	FALSE
1	2010-02-26	46.63	2.561	NA	NA	NA	NA	NA	211.3196429	8.106	FALSE
1	2010-03-05	46.5	2.625	NA	NA	NA	NA	NA	211.3501429	8.106	FALSE
1	2010-03-12	57.79	2.667	NA	NA	NA	NA	NA	211.3806429	8.106	FALSE



Store	Type	Size
1	A	151315
2	A	202307
3	B	37392
4	A	205863
5	B	34875
6	A	202505

Figure 6.2: Example Data Schema and Format Requirements

6.4.2 Model File Specifications

Model Serialization Format

- **Primary Format**: Joblib .pkl files
- **Compression**: Optional gzip compression supported
- **Python Version**: Must be serialized with Python 3.12
- **Cross-Platform**: Compatible across Windows, macOS, and Linux

Model File Structure

- **File Extension**: .pkl (required)
- **Naming Convention**: ModelType.pkl (e.g., AutoARIMA.pkl)
- **Size Limits**: Typically 1-50 MB depending on model complexity
- **Metadata**: Model parameters and performance metrics embedded

6.4.3 Output Format Specifications

CSV Export Format

- **Columns**: Week, Date, Predicted_Sales
- **Encoding**: UTF-8 character encoding
- **Delimiter**: Comma-separated values
- **Header Row**: Column names included in first row

JSON Export Format

- **Structure**: Array of objects with week, date, and prediction fields
- **Date Format**: ISO 8601 format (YYYY-MM-DD)
- **Numeric Precision**: 2 decimal places for currency values
- **Encoding**: UTF-8 JSON encoding

6.5 Performance Specifications

6.5.1 Model Training Performance

Training Time Expectations

- **Auto ARIMA**: 2-10 minutes depending on dataset size and parameters
- **Exponential Smoothing**: 10-60 seconds for typical datasets
- **Dataset Size Impact**: Linear scaling with number of data points
- **Hardware Impact**: Significant performance improvement with more CPU cores

Memory Usage

- **Base Application**: 200-500 MB RAM
- **Training Process**: Additional 500-2000 MB depending on dataset
- **Model Storage**: 1-50 MB per trained model
- **Peak Usage**: May spike during parameter optimization

6.5.2 Prediction Performance

Forecast Generation Speed

- **Prediction Time**: $<$ 1 second for 4-week forecasts
- **Model Loading**: 1-5 seconds depending on model complexity
- **Visualization Rendering**: 1-3 seconds for interactive charts
- **Export Generation**: $<$ 1 second for CSV/JSON files

Scalability Characteristics

- **Concurrent Users**: Local installation supports 1 user, cloud supports 50+
- **Session Management**: Independent user sessions with isolated state
- **Resource Sharing**: Efficient memory sharing for identical models
- **Load Balancing**: Automatic scaling in cloud deployment

6.6 Security and Compliance

6.6.1 Data Security

Local Installation Security

- **Data Storage**: All data remains on local machine
- **Network Traffic**: No external data transmission
- **User Authentication**: No authentication required for local use
- **File Permissions**: Standard OS file permission controls

Cloud Deployment Security

- **Data Transmission**: HTTPS encryption for all communications
- **Session Isolation**: Each user session is completely isolated
- **Data Persistence**: No permanent storage of uploaded data
- **Platform Security**: Streamlit Cloud security infrastructure

6.6.2 Privacy Considerations

Data Handling

- **Upload Privacy**: Uploaded files are not permanently stored in cloud
- **Session Cleanup**: All data deleted when session ends
- **No Logging**: User data is not logged or monitored
- **Compliance**: Suitable for internal business data analysis

6.7 Limitations and Constraints

6.7.1 Technical Limitations

Dataset Size Constraints

- **Cloud Upload**: 200 MB maximum file size per upload
- **Local Installation**: Limited by available system memory
- **Processing Time**: Large datasets may require extended processing time
- **Memory Requirements**: 8GB+ RAM recommended for datasets $>$ 1M rows

Model Limitations

- **Forecast Horizon**: Fixed 4-week prediction period
- **Algorithm Support**: Limited to Auto ARIMA and Exponential Smoothing
- **Update Frequency**: Models require retraining for updated forecasts
- **Seasonal Patterns**: Best performance with datasets containing seasonal cycles

6.7.2 Platform Constraints

Cloud Platform Limitations

- **Session Timeout**: 30-minute idle timeout
- **Shared Resources**: Performance may vary with platform load
- **Internet Dependency**: Requires stable internet connection
- **Browser Compatibility**: Modern browser required for full functionality

Local Installation Constraints

- **Python Version**: Strict requirement for Python 3.12.x
- **Single User**: Designed for single-user operation
- **Port Conflicts**: May conflict with other applications using same ports
- **Manual Updates**: Requires manual updates for new features

This comprehensive specification provides all technical requirements for successful system deployment and operation. The next chapter will cover maintenance procedures and best practices for optimal system performance.

7 Maintenance and Best Practices

7.1 System Maintenance Overview

Effective maintenance of the Walmart Sales Forecasting System ensures optimal performance, data quality, and reliable forecasting results. This chapter provides comprehensive guidance for regular maintenance procedures, performance optimization, and operational best practices.

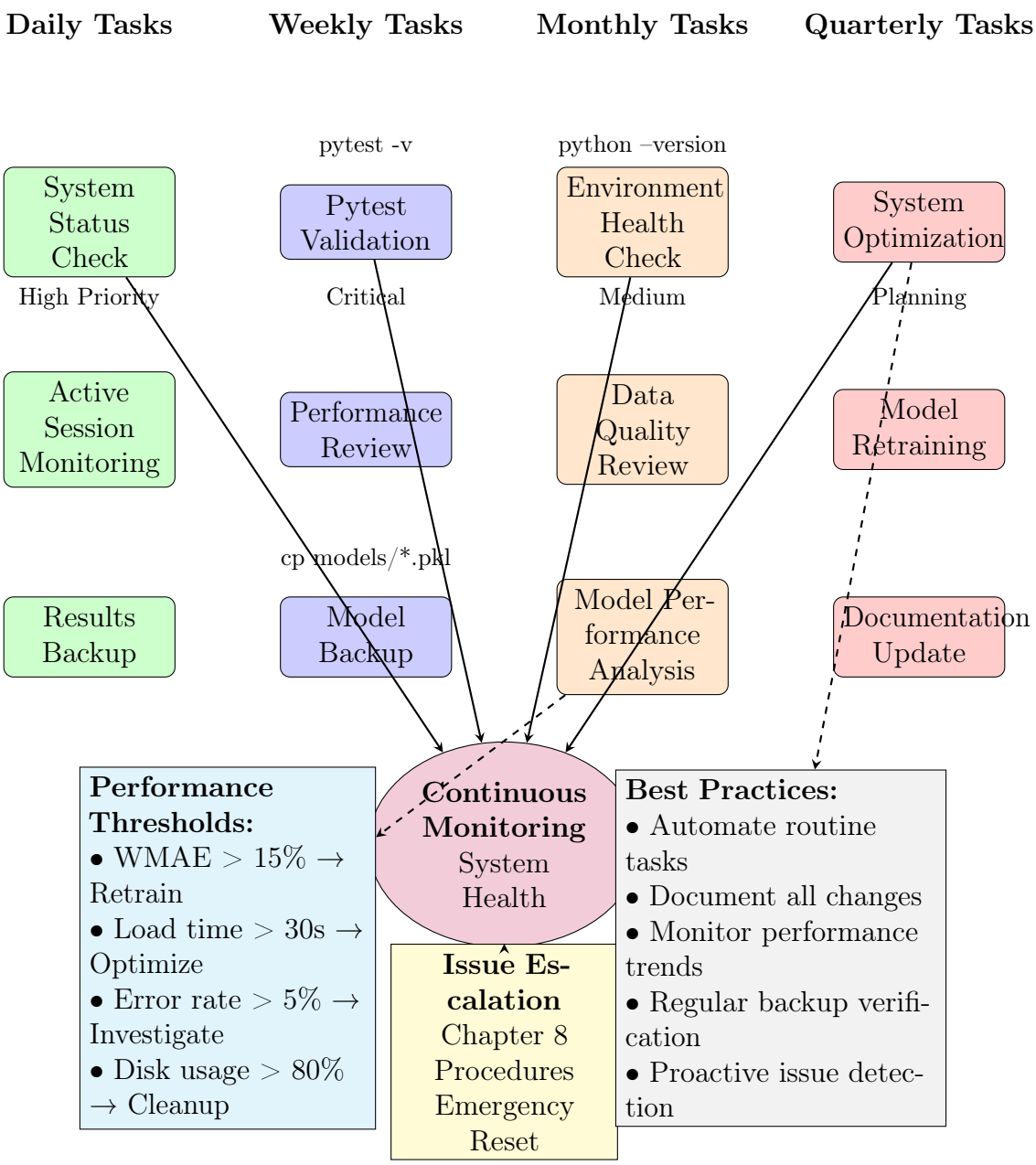


Figure 7.1: System Maintenance and Best Practices Workflow

7.2 Regular Maintenance Procedures

7.2.1 Local Installation Maintenance

Virtual Environment Management

Environment Health Checks

- **Monthly Verification**: Confirm Python 3.12.x version remains active
- **Package Integrity**: Run dependency checks to ensure no conflicts
- **Environment Activation**: Verify virtual environment activates correctly
- **Path Verification**: Ensure correct Python executable is being used

Environment Maintenance Commands

```
# Verify active environment and Python version
python --version
which python  # or where python on Windows

# Check installed packages for conflicts
pip check

# Update pip package manager
pip install --upgrade pip

# Recreate environment if issues persist
deactivate
rm -rf walmart_forecast_env
python3.12 -m venv walmart_forecast_env
source walmart_forecast_env/bin/activate
pip install -r requirements.txt
```

Application Health Monitoring

Regular Health Checks

- **Weekly Testing**: Run pytest validation suite to verify functionality
- **Interface Testing**: Launch both applications to check loading times
- **Model Loading**: Test default model loading and prediction generation
- **File Operations**: Verify upload/download functionality works correctly

Health Check Procedures

```
# Comprehensive test suite execution
pytest testWalmartSalesTraining.py -v
pytest testWalmartSalesPrediction.py -v

# Application startup verification
streamlit run walmartSalesTrainingApp.py --server.headless=true
streamlit run walmartSalesPredictionApp.py --server.port=8502 -
```

```
# Clean shutdown after testing
pkill -f streamlit
```

7.2.2 Model Management

Model Version Control

Model Backup Strategy

- **Regular Backups**: Create weekly backups of trained models
- **Version Naming**: Use timestamp-based naming for model versions
- **Performance Tracking**: Document WMAE scores for each model version
- **Storage Organization**: Maintain organized directory structure

Model Backup Procedure

```
# Create backup directory with timestamp
backup_date=$(date +%Y%m%d_%H%M%S)
mkdir -p models/backups/$backup_date

# Copy current models to backup location
cp models/default/*.pkl models/backups/$backup_date/

# Document model performance
echo "Backup created: $backup_date" >> models/backup_log.txt
echo "ExponentialSmoothingHoltWinters.pkl - WMAE: 3.58
        %" >> models/backup_log.txt
```

Model Performance Monitoring

Performance Tracking

- **WMAE Monitoring**: Track model accuracy over time
- **Prediction Consistency**: Compare forecasts across time periods
- **Seasonal Adaptation**: Monitor model performance during different seasons
- **Data Drift Detection**: Identify when model retraining is needed

Date	Model	WMAE	Status
2025-06-01	Exponential Smoothing	3.58%	Excellent
2025-05-15	Auto ARIMA	4.12%	Excellent
2025-05-01	Exponential Smoothing	6.23%	Acceptable
2025-04-15	Auto ARIMA	8.45%	Acceptable

Table 7.1: Model Performance Tracking Log Example

7.3 Data Quality Management

7.3.1 Input Data Validation

Data Quality Checks

- **Schema Validation**: Verify required columns are present
- **Data Type Checking**: Ensure numeric columns contain valid numbers
- **Date Format Validation**: Confirm dates follow YYYY-MM-DD format
- **Missing Value Assessment**: Identify and handle missing data appropriately

Data Quality Checklist

1. Check for required columns in all three CSV files
2. Verify date ranges are consistent across files
3. Ensure no negative sales values in training data
4. Validate store IDs match across all datasets
5. Confirm no duplicate date-store combinations

7.3.2 Data Refresh Procedures

Regular Data Updates

- **Monthly Updates**: Incorporate new sales data for improved accuracy
- **Seasonal Adjustments**: Update seasonal parameters based on recent patterns
- **Feature Updates**: Add new external factors that may affect sales
- **Historical Validation**: Verify updated data maintains historical consistency

Data Update Workflow

1. Export new data in required CSV format
2. Validate data quality using established checks
3. Backup existing models before retraining
4. Retrain models with updated dataset
5. Compare new model performance to previous versions
6. Deploy new models if performance improves

7.4 Performance Optimization

7.4.1 System Performance Tuning

Memory Optimization

Memory Management Strategies

- **Data Chunking**: Process large datasets in smaller chunks
- **Memory Monitoring**: Track memory usage during training and prediction
- **Garbage Collection**: Force garbage collection after large operations
- **Model Caching**: Implement intelligent model caching for repeated use

Memory Optimization Commands

```
import gc
import psutil

# Monitor memory usage
process = psutil.Process()
memory_info = process.memory_info()
print(f"Memory usage: {memory_info.rss / 1024 / 1024:.2f} MB")

# Force garbage collection
gc.collect()

# Clear large variables when done
del large_dataset
gc.collect()
```

Processing Speed Optimization

Performance Enhancement Techniques

- **Parallel Processing**: Utilize multiple CPU cores for training
- **Algorithm Tuning**: Optimize hyperparameter search ranges
- **Caching**: Cache intermediate results to avoid recomputation
- **Efficient Data Structures**: Use optimized pandas operations

7.4.2 Cloud Usage Optimization

Efficient Cloud Operations

Resource Management

- **Session Management**: Close idle sessions to free resources
- **File Size Optimization**: Compress large datasets before upload
- **Batch Operations**: Group similar operations to maximize efficiency
- **Off-Peak Usage**: Use cloud resources during off-peak hours when possible

Cloud Best Practices

- **Regular Saves**: Download results frequently to prevent data loss
- **Browser Management**: Use dedicated browser tabs for each application
- **Network Stability**: Ensure stable internet connection for large operations
- **Timeout Awareness**: Be aware of 30-minute idle timeout limits

7.5 Security and Access Management

7.5.1 Local Installation Security

Security Best Practices

- **Environment Isolation**: Keep forecasting environment separate from other projects
- **File Permissions**: Set appropriate permissions on model and data files
- **Network Security**: Disable external network access if not needed
- **Regular Updates**: Keep Python and packages updated for security patches

Access Control

- **User Accounts**: Use dedicated user accounts for forecasting operations
- **Directory Permissions**: Restrict access to model and data directories
- **Backup Security**: Secure backup files with appropriate permissions
- **Audit Logging**: Maintain logs of model training and prediction activities

7.5.2 Data Privacy Protection

Sensitive Data Handling

- **Data Anonymization**: Remove or mask personally identifiable information
- **Secure Transmission**: Use secure methods for data transfer
- **Storage Encryption**: Encrypt sensitive data files at rest
- **Access Logging**: Log access to sensitive forecasting data

Compliance Considerations

- **Data Retention**: Establish data retention policies for training data
- **Access Controls**: Implement role-based access to forecasting system
- **Audit Trails**: Maintain audit trails for compliance requirements
- **Privacy Policies**: Follow organizational privacy policies for data use

7.6 Operational Best Practices

7.6.1 Workflow Standardization

Training Workflow Standards

Standardized Training Process

1. **Data Preparation**: Validate data quality before training
2. **Parameter Documentation**: Record all hyperparameter settings
3. **Performance Baseline**: Compare new models to established baselines
4. **Model Documentation**: Document model purpose and performance
5. **Deployment Testing**: Test models thoroughly before production use

Training Documentation Template

- **Model Name**: Descriptive name with version number
- **Training Date**: Date and time of model training
- **Dataset Version**: Version or date of training data used
- **Hyperparameters**: Complete list of model parameters
- **Performance Metrics**: WMAE scores and interpretation
- **Notes**: Any special considerations or observations

Prediction Workflow Standards

Standardized Prediction Process

1. **Model Verification**: Confirm correct model is loaded
2. **Prediction Generation**: Generate forecasts using standard procedure
3. **Result Validation**: Review forecasts for reasonableness
4. **Documentation**: Record prediction parameters and results
5. **Distribution**: Share results with appropriate stakeholders

7.6.2 Quality Assurance

Prediction Quality Control

Forecast Validation Procedures

- **Reasonableness Checks**: Verify forecasts fall within expected ranges
- **Trend Analysis**: Ensure forecasted trends align with business expectations
- **Seasonal Patterns**: Confirm seasonal patterns are appropriately captured
- **Comparative Analysis**: Compare forecasts across different models

Quality Metrics

- **Consistency**: Forecasts should be consistent across similar time periods
- **Stability**: Small data changes should not cause dramatic forecast changes
- **Business Alignment**: Forecasts should align with business knowledge
- **Historical Performance**: Models should perform well on historical data

Continuous Improvement

Performance Monitoring

- **Regular Evaluation**: Assess model performance on new data
- **Comparative Studies**: Compare different models and approaches
- **Parameter Optimization**: Continuously refine hyperparameters
- **Feature Engineering**: Explore new features to improve accuracy

Improvement Documentation

- **Change Log**: Document all system and model changes
- **Performance Tracking**: Maintain historical performance records
- **Lessons Learned**: Document insights and best practices discovered
- **Recommendations**: Provide recommendations for future improvements

7.7 Maintenance Schedule

7.7.1 Daily Operations

Daily Tasks

- **System Status Check**: Verify applications are running correctly
- **Active Session Monitoring**: Monitor for any error messages or warnings
- **Results Backup**: Save important prediction results
- **Usage Documentation**: Record significant forecasting activities

7.7.2 Weekly Maintenance

Weekly Tasks

- **Test Suite Execution**: Run complete pytest validation
- **Performance Review**: Analyze system performance metrics
- **Model Backup**: Create backup copies of all models
- **Documentation Update**: Update maintenance logs and documentation

7.7.3 Monthly Maintenance

Monthly Tasks

- **Environment Health Check**: Comprehensive virtual environment validation
- **Data Quality Review**: Analyze data quality and identify improvements
- **Model Performance Analysis**: Detailed analysis of model accuracy trends
- **Security Review**: Review access logs and security configurations

7.7.4 Quarterly Maintenance

Quarterly Tasks

- **System Optimization**: Comprehensive performance optimization review
- **Model Retraining**: Consider retraining models with updated data
- **Documentation Review**: Update all documentation and procedures
- **Capacity Planning**: Assess future resource and capability needs

Frequency	Task	Duration	Priority
Daily	Status Check	5 minutes	High
Weekly	Test Suite	15 minutes	High
Monthly	Health Check	30 minutes	Medium
Quarterly	Full Review	2 hours	Medium

Table 7.2: Maintenance Schedule Summary

This comprehensive maintenance guide ensures reliable system operation and optimal forecasting performance. The next chapter will address common problems and their solutions through detailed troubleshooting procedures.

8 Troubleshooting

8.1 Troubleshooting Framework

This chapter provides comprehensive problem resolution procedures for the Walmart Sales Forecasting System. It covers common issues, diagnostic procedures, and systematic approaches to resolving problems in both local and cloud deployments.

8.2 Installation and Setup Issues

8.2.1 Python Version Problems

Wrong Python Version

Problem: System reports Python version other than 3.12.x

Symptoms:

- Package installation failures
- Import errors when running applications
- Virtual environment creation errors
- Dependency conflict messages

Diagnostic Steps:

```
# Check current Python version
python --version
python3 --version

# Check which Python executable is being used
which python
which python3

# List all available Python versions
ls /usr/bin/python*  # Linux/macOS
```

Solution:

1. Download Python 3.12.x from `https://www.python.org/downloads/`
2. Install the correct version

3. Use explicit version commands:

```
# Use specific Python version
python3.12 -m venv walmart_forecast_env
python3.12 -m pip install -r requirements.txt
```

4. Update system PATH if necessary

Multiple Python Installations

Problem: Conflicts between different Python installations
Symptoms:

- Inconsistent behavior between terminals
- Package installed but not found
- Virtual environment using wrong Python version

Solution:

```
# Explicitly specify Python executable
/usr/bin/python3.12 -m venv walmart_forecast_env

# Or use pyenv for version management (recommended)
pyenv install 3.12.0
pyenv local 3.12.0
python -m venv walmart_forecast_env
```

8.2.2 Virtual Environment Issues

Environment Activation Failures

Problem: Virtual environment won't activate
Symptoms:

- "No such file or directory" errors
- Activation script not found
- Permission denied errors

Diagnostic Steps:

```
# Check if virtual environment directory exists
ls -la walmart_forecast_env/

# Check activation script
ls -la walmart_forecast_env/bin/activate  # Linux/macOS
```

```
ls -la walmart_forecast_env\Scripts\activate.bat  # Windows

# Check permissions
ls -la walmart_forecast_env/bin/
```

Solution:

```
# Recreate virtual environment
rm -rf walmart_forecast_env
python3.12 -m venv walmart_forecast_env

# Fix permissions if needed (Linux/macOS)
chmod +x walmart_forecast_env/bin/activate

# Use alternative activation method
source walmart_forecast_env/bin/activate
# or
. walmart_forecast_env/bin/activate
```

8.2.3 Dependency Installation Problems

Package Installation Failures

Problem: pip install fails for required packages

Symptoms:

- Build errors during installation
- "No module named" errors after installation
- Version conflict messages
- SSL certificate errors

Diagnostic Steps:

```
# Check pip version
pip --version

# Verify virtual environment is active
which pip
which python

# Check for conflicting packages
pip check

# Try manual installation of problematic package
pip install -v package_name
```

Solution:

```
# Upgrade pip first
pip install --upgrade pip

# Clear pip cache
pip cache purge

# Install with specific options
pip install --no-cache-dir -r requirements.txt

# For SSL issues
pip install --trusted-host pypi.org --trusted-host pypi.python.org

# Install packages one by one to identify problem
pip install streamlit==1.32.0
pip install pandas==2.2.2
# ... continue with each package
```

8.3 Application Launch Issues

8.3.1 Streamlit Startup Problems

Port Already in Use

Problem: "Address already in use" error when launching

Symptoms:

- Error: "OSError: [Errno 48] Address already in use"
- Application won't start
- Previous sessions still running

Diagnostic Steps:

```
# Check what's using the port
lsof -i :8501  # Linux/macOS
netstat -ano | findstr :8501  # Windows

# List all Streamlit processes
ps aux | grep streamlit  # Linux/macOS
tasklist | findstr streamlit  # Windows
```

Solution:

```
# Kill existing Streamlit processes
pkill -f streamlit  # Linux/macOS
taskkill /F /IM python.exe  # Windows (be careful!)

# Or use different port
streamlit run walmartSalesTrainingApp.py --server.port=8503
streamlit run walmartSalesPredictionApp.py --server.port=8504

# Or specify port in command
streamlit run app.py --server.port=8502
```

Import Errors on Startup

Problem: Module import errors when launching applications

Symptoms:

- "ModuleNotFoundError" messages
- "ImportError" for specific packages
- Application crashes immediately

Diagnostic Steps:

```
# Test imports manually
python -c "import streamlit; print('Streamlit OK')"
python -c "import pandas; print('Pandas OK')"
python -c "import plotly; print('Plotly OK')"

# Check installed packages
pip list | grep streamlit
pip show streamlit
```

Solution:

```
# Reinstall problematic packages
pip uninstall streamlit
pip install streamlit==1.32.0

# Verify virtual environment is active
echo $VIRTUAL_ENV

# Reinstall all dependencies
pip install --force-reinstall -r requirements.txt
```

8.4 Cloud Application Issues

8.4.1 Cloud Access Problems

Application Won't Load

Problem: Cloud application shows loading screen indefinitely
Symptoms:

- Endless "Please wait..." message
- White screen after initial load
- Browser shows "Connection timed out"

Diagnostic Steps:

1. Check internet connection stability
2. Try different browser (Chrome recommended)
3. Check browser console for errors (F12)
4. Test with incognito/private browsing mode

Solution:

1. Refresh the page (Ctrl+F5 for hard refresh)
2. Clear browser cache and cookies
3. Disable browser extensions temporarily
4. Try different network connection
5. Wait 5-10 minutes and retry (server may be busy)

Session Timeout Issues

Problem: Application resets unexpectedly
Symptoms:

- Loaded models disappear
- Interface resets to initial state
- "Please rerun" messages appear

Solution:

- Be aware of 30-minute idle timeout

- Save/download results frequently
- Keep browser tab active
- Refresh page to restart session
- Use local installation for long training sessions

8.4.2 File Upload Problems

Upload Failures

Problem: File uploads fail or hang
Symptoms:

- Upload progress bar stuck
- "File too large" errors
- Upload completes but file not recognized

Diagnostic Steps:

1. Check file size (must be $<$ 200MB for cloud)
2. Verify file format (.csv for data, .pkl for models)
3. Test with smaller sample file
4. Check network connection stability

Solution:

1. Compress large files or reduce dataset size
2. Ensure files are saved in correct format
3. Try upload during off-peak hours
4. Use local installation for large files
5. Split large datasets into smaller chunks

8.5 Data-Related Issues

8.5.1 Data Format Problems

CSV Schema Errors

Problem: Data files don't meet required schema

Symptoms:

- "Missing required column" errors
- "Data type mismatch" warnings
- Processing fails after upload

Required Schema Validation:

vs

Solution:

1. Verify all required columns are present
2. Check column names for exact spelling and capitalization
3. Ensure date format is YYYY-MM-DD
4. Validate numeric columns contain only numbers
5. Remove any extra columns not required by schema

Figure 8.1: Data Schema Validation Error Resolution

Data Quality Issues

Problem: Data contains invalid or inconsistent values

Symptoms:

- Training fails with "Invalid data" errors
- Negative sales values causing rejection
- Date parsing errors
- Store ID mismatches between files

Data Quality Validation:

```
# Check for negative sales
negative_sales = train_df[train_df['Weekly_Sales'] < 0]
print(f"Negative sales records: {len(negative_sales)}")

# Check date format consistency
try:
pd.to_datetime(train_df['Date'])
print("Date format OK")
except:
print("Date format issues detected")

# Check Store ID consistency
train_stores = set(train_df['Store'].unique())
features_stores = set(features_df['Store'].unique())
stores_stores = set(stores_df['Store'].unique())
print("Store ID mismatches:", train_stores.symmetric_difference
(features_stores))
```

Figure 8.2: Data Schema Validation Error Resolution

Solution:

1. Remove or correct negative sales values
2. Standardize date formats to YYYY-MM-DD
3. Ensure Store IDs match across all files
4. Fill missing values appropriately
5. Validate data ranges are reasonable

8.5.2 Model Compatibility Issues

Model Loading Failures

Problem: Custom models fail to load

Symptoms:

- "Invalid model file" errors
- "Model format not supported" messages
- Application crashes when loading model

Diagnostic Steps:

```
import joblib
import pickle

# Test model file integrity
```

```
try:
model = joblib.load('model.pkl')
print("Joblib loading successful")
except Exception as e:
print(f"Joblib error: {e}")

try:
with open('model.pkl', 'rb') as f:
model = pickle.load(f)
print("Pickle loading successful")
except Exception as e:
print(f"Pickle error: {e}")
```

Solution:

1. Ensure model was trained with Python 3.12
2. Verify model type matches selection (ARIMA vs Exponential Smoothing)
3. Retrain model if cross-platform compatibility issues
4. Check file wasn't corrupted during transfer
5. Use same package versions for training and prediction

8.6 Performance Issues

8.6.1 Slow Training Performance

Training Takes Too Long

Problem: Model training exceeds expected time
Symptoms:

- Auto ARIMA takes hours instead of minutes
- Application becomes unresponsive
- Memory usage continuously increases

Diagnostic Steps:

```
import psutil
import time

# Monitor system resources
def monitor_resources():
process = psutil.Process()
```

```
while True:
cpu_percent = process.cpu_percent()
memory_mb = process.memory_info().rss / 1024 / 1024
print(f"CPU: {cpu_percent}%, Memory: {memory_mb:.1f}MB")
time.sleep(10)
```

Solution:

1. Reduce hyperparameter search space:

```
# Use smaller parameter ranges
max_p = 5  # instead of 20
max_q = 5  # instead of 20
```

2. Use smaller dataset for initial testing
3. Increase system RAM if possible
4. Use stepwise ARIMA search instead of exhaustive
5. Consider Exponential Smoothing for faster training

8.6.2 Memory Issues

Out of Memory Errors

Problem: Application runs out of memory during training

Symptoms:

- "MemoryError" exceptions
- System becomes slow and unresponsive
- Training process killed by OS

Solution:

```
# Implement data chunking
def process_in_chunks(data, chunk_size=10000):
for i in range(0, len(data), chunk_size):
chunk = data[i:i+chunk_size]
# Process chunk
yield chunk

# Force garbage collection
import gc
gc.collect()

# Use memory-efficient data types
```

```
df = df.astype({
        'Store': 'int16',
        'Weekly_Sales': 'float32'
})
```

```python
import gc

def process_in_chunks(data, chunk_size=10000):
    """
    Generator to process data in chunks.
    """
    for i in range(0, len(data), chunk_size):
        chunk = data.iloc[i:i+chunk_size]
        # Example: convert to memory-efficient types
        chunk = chunk.astype({
            'Store': 'int16',
            'Weekly_Sales': 'float32'
        }, errors='ignore')
        yield chunk
        gc.collect()

# Example usage:
for chunk in process_in_chunks(train_df):
    # Replace with your processing logic
    print(f"Processing chunk with {len(chunk)} rows")
```

[8] ✓ 1.2s

```
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
Processing chunk with 10000 rows
```

Figure 8.3: Memory Management and Optimization Interface

8.7 Prediction Issues

8.7.1 Prediction Generation Failures

Forecast Generation Errors

Problem: Prediction generation fails with loaded model
Symptoms:

- "Error generating predictions" messages
- NaN values in forecast results
- Unrealistic forecast values

Diagnostic Steps:

```
# Test model prediction manually
try:
if model_type == "Auto_ARIMA":
predictions = model.predict(n_periods=4)
else:
predictions = model.forecast(4)
print("Predictions:", predictions)
except Exception as e:
print(f"Prediction_error:_{e}")
```

Solution:

1. Verify model was trained successfully
2. Check model type matches prediction method
3. Ensure model has sufficient training data
4. Retrain model if necessary
5. Check for data preprocessing consistency

8.7.2 Result Interpretation Issues

Unexpected Forecast Values

Problem: Forecasts seem unrealistic or inconsistent
Symptoms:

- All negative predictions
- Extremely large prediction values
- No seasonal patterns visible

Analysis Steps:

- Remember predictions are week-over-week changes, not absolute sales
- Check training data for similar patterns
- Compare with historical sales changes
- Verify model performance metrics (WMAE)

Solution:

1. Review model training performance
2. Consider retraining with different parameters
3. Validate training data quality
4. Compare multiple model types
5. Consult domain expertise for reasonableness

8.8 Test Suite Validation

8.8.1 Using Pytest for Troubleshooting

Running Diagnostic Tests

Comprehensive Test Execution:

```
# Run all tests with verbose output
pytest testWalmartSalesTraining.py -v
pytest testWalmartSalesPrediction.py -v

# Run specific test categories
pytest -k "test_load" -v  # Only model loading tests
pytest -k "test_predict" -v  # Only prediction tests

# Run tests with detailed output
pytest --tb=long -v  # Long traceback format
pytest --tb=short -v  # Short traceback format

# Run tests and stop on first failure
pytest -x -v

# Generate test coverage report
pytest --cov=walmartSalesTrainingCore --cov=walmartSalesPredictionC
```

Interpreting Test Results:

- **All tests PASSED**: System is functioning correctly
- **Specific test FAILED**: Identifies exact problem area
- **Import errors**: Package installation issues
- **Timeout errors**: Performance or resource issues

Figure 8.4: Pytest Validation and Diagnostic Results

Custom Diagnostic Tests

Create Custom Test Scripts:

```
import sys
import subprocess

def test_python_version():
    version = sys.version_info
    assert version.major == 3 and version.minor == 12, f"Wrong Python version: {version}"
    print("Python version correct")

def test_package_imports():
    try:
        import streamlit, pandas, numpy, plotly
        print("All packages import successfully")
    except ImportError as e:
        print(f"Import error: {e}")

def test_application_startup():
    try:
        result = subprocess.run(['streamlit', '--version'],
                                capture_output=True, text=True, timeout=10)
        assert result.returncode == 0
        print("Streamlit is working")
    except Exception as e:
        print(f"Streamlit test failed: {e}")

if __name__ == "__main__":
    test_python_version()
    test_package_imports()
    test_application_startup()
```

```
[9]  ✓ 0.8s

...  Python version correct
     All packages import successfully
     Streamlit is working
```

Figure 8.5: Pytest Validation and Diagnostic Results

8.9 Advanced Troubleshooting

8.9.1 Log Analysis

Enabling Debug Logging

Enhanced Logging Configuration:

```
import logging
import streamlit as st

# Configure detailed logging
logging.basicConfig(
    level=logging.DEBUG,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    filename='forecast_debug.log'
)

# Add console handler
console = logging.StreamHandler()
console.setLevel(logging.INFO)
logging.getLogger().addHandler(console)

# Log application events
logger = logging.getLogger(__name__)
logger.info("Application starting")
logger.debug("Debug information")
```

```
[10]  ✓ 0.0s

...   Application starting
```

Figure 8.6: Pytest Validation and Diagnostic Results

8.9.2 Performance Profiling

8.10 Emergency Recovery Procedures

8.10.1 System Recovery

Complete Environment Reset

Nuclear Option - Full Reset:

```
# Backup important files first
cp -r models/default/ backup_models/

# Remove everything
rm -rf walmart_forecast_env/
pkill -f streamlit

# Fresh installation
python3.12 -m venv walmart_forecast_env
source walmart_forecast_env/bin/activate
pip install --upgrade pip
pip install -r requirements.txt

# Verify installation
pytest testWalmartSalesTraining.py -v
pytest testWalmartSalesPrediction.py -v

# Restore models
cp backup_models/* models/default/
```

Figure 8.7: Pytest Validation and Diagnostic code

8.10.2 Data Recovery

Recovering Lost Work

Model Recovery Steps:

1. Check models/default/ directory for saved models
2. Look in backup directories created by maintenance procedures
3. Re-download models from cloud applications if available
4. Retrain models using original training data
5. Document lessons learned to prevent recurrence

8.11 Getting Additional Help

8.11.1 Documentation Resources

Internal Documentation:

- This user manual for comprehensive guidance
- Code comments in application files
- Test files for expected behavior examples
- README files for quick reference

External Resources:

- Streamlit documentation: `https://docs.streamlit.io/`
- pandas documentation: `https://pandas.pydata.org/docs/`
- pmdarima documentation: `https://alkaline-ml.com/pmdarima/`
- statsmodels documentation: `https://www.statsmodels.org/`

8.11.2 Community Support

Online Communities:

- Stack Overflow for programming questions
- Streamlit Community Forum
- Reddit r/MachineLearning and r/Python
- GitHub Issues for package-specific problems

8.12 Troubleshooting Quick Reference

Problem	Quick Check	Quick Fix
App won't start	Python version	python3.12 -m venv env
Import errors	Virtual env active	pip install -r requirements.txt
Port in use	Check processes	pkill -f streamlit
Upload fails	File size/format	Check file $<$ 200MB, .csv/.pkl
Model won't load	File integrity	Retrain and re-export model
Slow training	Resource usage	Reduce parameter ranges
Cloud timeout	Session length	Download results frequently
Tests fail	Environment	Run pytest -v for details

Table 8.1: Troubleshooting Quick Reference Guide

This comprehensive troubleshooting guide addresses the most common issues users encounter with the Walmart Sales Forecasting System. The next chapter provides additional help resources and support information.

9 Help and Support

9.1 Support Resources Overview

This chapter provides comprehensive guidance for obtaining additional assistance with the Walmart Sales Forecasting System. It includes frequently asked questions, user support resources, and pathways for getting help when standard troubleshooting procedures don't resolve issues.

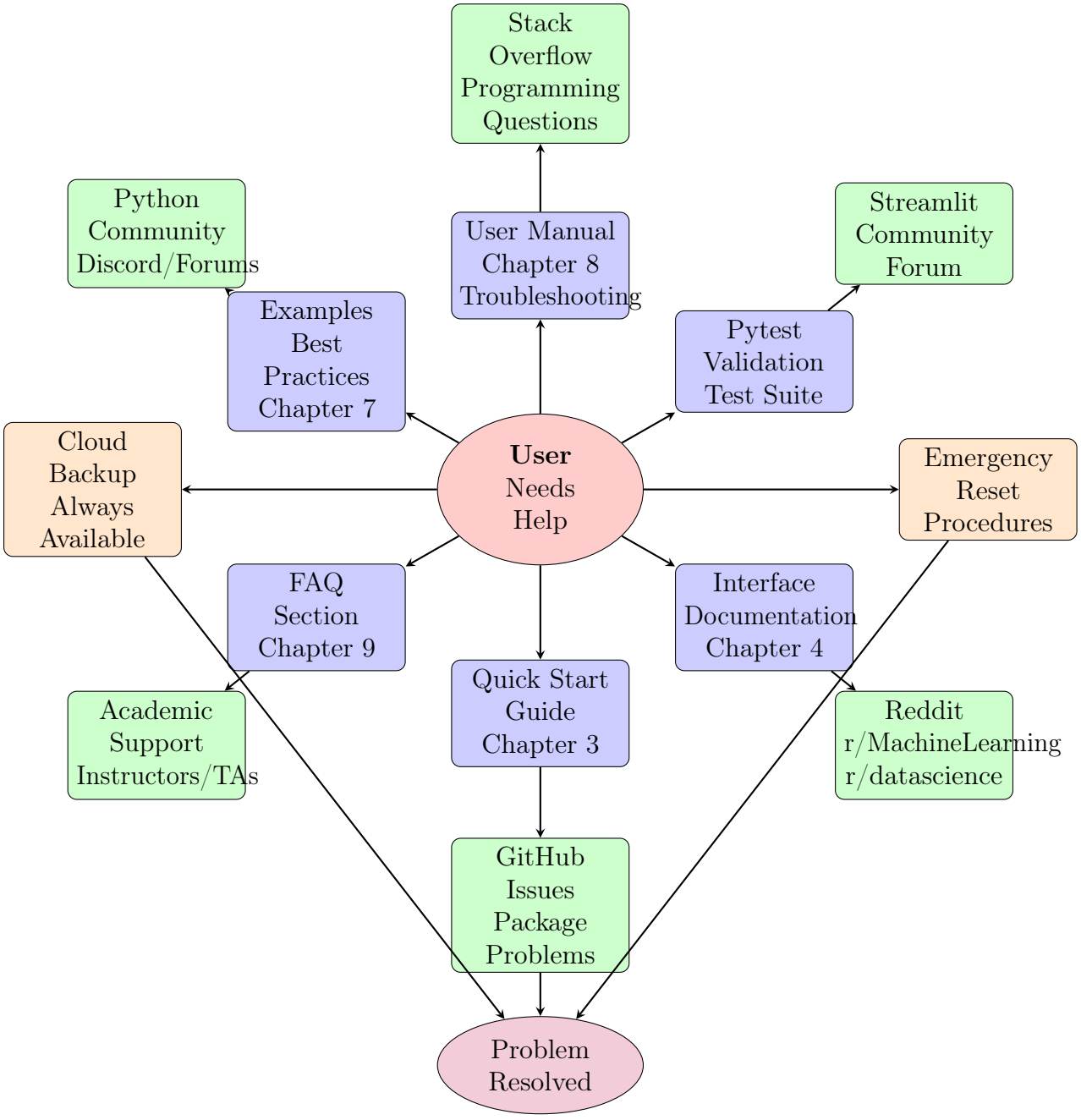


Figure 9.1: Comprehensive Support Ecosystem

9.2 Frequently Asked Questions

9.2.1 General System Questions

Getting Started

Q: Which deployment option should I choose?
A: For quick evaluation and occasional use, choose cloud deployment. For regular use, large datasets, or organizational requirements, choose local installation.

Q: Do I need programming experience to use the system?
A: No programming experience is required for basic usage. The web interface handles all technical aspects. However, understanding time series concepts is helpful for interpreting results.

Q: How accurate are the forecasts?
A: The default Exponential Smoothing model achieves 3.58% normalized WMAE, which is considered excellent for business forecasting. Custom models may achieve different accuracy levels depending on data quality and parameters.

Q: Can I use this system for non-retail forecasting?
A: While designed for retail sales, the system can potentially work with other time series data that follows similar patterns. However, optimal performance is achieved with sales-type data containing seasonal patterns.

Data Requirements

Q: What if I don't have all three required CSV files?
A: All three files (train.csv, features.csv, stores.csv) are required for training. If you're missing features or stores data, you can create minimal versions with basic information or use the default models for prediction only.

Q: How much historical data do I need?
A: Minimum 2 years of weekly data is recommended for seasonal pattern detection. More data (3-5 years) typically improves model accuracy, especially for capturing seasonal variations.

Q: Can I use daily or monthly data instead of weekly?
A: The system is specifically designed for weekly sales data. Using other frequencies would require code modifications and retraining of models.

9.2.2 Technical Questions

Installation and Setup

Q: Why specifically Python 3.12 and not newer versions?
A: The system was developed and tested with Python 3.12. Package dependencies are validated for this version. Newer Python versions may have compatibility issues with specific package versions.

Q: Can I run both applications simultaneously?
A: Yes, the applications are designed to run on different ports (8501 and 8502). This allows simultaneous training and prediction operations.

Q: What happens if my local installation stops working?
A: Use the troubleshooting procedures in Chapter 8, particularly the pytest validation tests. As a backup, you can always use the cloud versions while fixing local issues.

Model and Predictions

Q: Why do I get negative forecast values?
A: Negative values indicate predicted sales decreases from the previous week. This is normal business behavior and doesn't indicate errors. The system forecasts changes, not absolute values.

Q: Can I forecast beyond 4 weeks?
A: The current system is optimized for 4-week forecasts. Longer horizons typically require model modifications and retraining for accuracy.

Q: How often should I retrain models?
A: Retrain quarterly or when performance degrades. If you notice forecasts becoming less accurate, it's time to retrain with fresh data.

9.2.3 Advanced Usage Questions

Model Customization

Q: How do I choose between Auto ARIMA and Exponential Smoothing?
A: Exponential Smoothing is generally faster and works well with clear seasonal patterns. Auto ARIMA is more flexible but takes longer to train. Try both and compare WMAE scores.

Q: What hyperparameters should I adjust first?
A: For Auto ARIMA, start with max_p and max_q values (try 5-10). For Exponential Smoothing, adjust seasonal_periods to match your business cycle.

Q: Can I combine forecasts from multiple models?
A: The current system doesn't support ensemble methods, but you can manually combine forecasts from different models exported as CSV files.

Data and Performance

Q: My training takes very long. How can I speed it up?
A: Reduce hyperparameter search ranges, use smaller datasets for testing, or switch to Exponential Smoothing. Consider upgrading hardware for large datasets.

Q: What's the maximum dataset size I can use?
A: Cloud deployment supports up to 200MB files. Local installation is limited by available RAM. For large datasets, consider data sampling or chunking.

9.3 User Support Resources

9.3.1 Self-Help Resources

Documentation Hierarchy

1. **This User Manual**: Comprehensive coverage of all system aspects
2. **Quick Start Guide**: Chapter 3 for immediate usage
3. **Troubleshooting Guide**: Chapter 8 for problem resolution
4. **Interface Documentation**: Chapter 4 for detailed UI guidance

Hands-On Learning

Recommended Learning Path:

1. Start with cloud Prediction Application using default model
2. Generate several forecasts to understand output format
3. Try local installation following Chapter 2 procedures
4. Experiment with Training Application using sample data
5. Practice model transfer between applications
6. Explore hyperparameter tuning for optimization

Practice Exercises:

- **Exercise 1**: Generate forecasts with default model and compare results
- **Exercise 2**: Train custom models with different hyperparameters
- **Exercise 3**: Upload custom model and generate predictions
- **Exercise 4**: Analyze forecast trends and seasonal patterns

9.3.2 Community Resources

Online Communities

Time Series Forecasting Communities:

- **Reddit r/MachineLearning**: General ML discussions and help
- **Reddit r/datascience**: Data science community with forecasting expertise
- **Stack Overflow**: Technical programming questions

- **Cross Validated**: Statistical modeling and forecasting theory

Python and Streamlit Communities:

- **Streamlit Community Forum**: `https://discuss.streamlit.io/`
- **Python Discord**: Real-time help and discussion
- **Python Reddit**: r/Python for general Python questions
- **GitHub Issues**: Package-specific problems and bug reports

Educational Resources

Time Series Learning Resources:

- **Online Courses**: Coursera, edX time series courses
- **Books**: "Forecasting: Principles and Practice" by Hyndman and Athanasopoulos
- **Tutorials**: Towards Data Science articles on time series
- **Videos**: YouTube tutorials on ARIMA and Exponential Smoothing

Technical Documentation:

- **pmdarima**: `https://alkaline-ml.com/pmdarima/`
- **statsmodels**: `https://www.statsmodels.org/`
- **pandas**: `https://pandas.pydata.org/docs/`
- **Streamlit**: `https://docs.streamlit.io/`

9.4 Technical Support

9.4.1 Issue Reporting

Before Reporting Issues

Pre-Report Checklist:

1. Review relevant sections of this manual
2. Follow troubleshooting procedures from Chapter 8
3. Run pytest validation tests and note results
4. Try reproducing the issue with minimal data
5. Document exact error messages and steps to reproduce

Information to Include

Essential Information for Support:

- **System Information**: OS, Python version, package versions
- **Deployment Type**: Local installation or cloud usage
- **Exact Error Messages**: Copy full error text
- **Steps to Reproduce**: Detailed procedure that causes the issue
- **Expected vs Actual Behavior**: What should happen vs what happens
- **Screenshots**: Visual evidence of the problem

System Information Script:

```
import sys
import platform
import pandas as pd
import streamlit as st
import numpy as np

print("=== System Information ===")
print(f"Python Version: {sys.version}")
print(f"Platform: {platform.platform()}")
print(f"Architecture: {platform.architecture()}")
print("\n=== Package Versions ===")
print(f"Streamlit: {st.__version__}")
print(f"Pandas: {pd.__version__}")
print(f"NumPy: {np.__version__}")

# Add more packages as needed
```

9.4.2 Escalation Procedures

When to Escalate

Escalation Criteria:

- Standard troubleshooting procedures don't resolve the issue
- System security or data integrity concerns
- Critical functionality completely unavailable
- Suspected bugs in the application code

Academic Context Support

For Academic Environments:

- **Instructor Assistance**: Consult course instructor for assignment-related questions
- **Teaching Assistants**: Get help with technical implementation details
- **Lab Resources**: Use computer lab resources for installation issues
- **Peer Study Groups**: Collaborate with classmates on understanding concepts

9.5 Quick Reference Guides

9.5.1 Command Quick Reference

Essential Commands

Installation Commands:

```
# Create virtual environment
python3.12 -m venv walmart_forecast_env

# Activate environment
source walmart_forecast_env/bin/activate  # Linux/macOS
walmart_forecast_env\Scripts\activate     # Windows

# Install dependencies
pip install -r requirements.txt

# Run applications
streamlit run walmartSalesTrainingApp.py
streamlit run walmartSalesPredictionApp.py --server.port=8502
```

Troubleshooting Commands:

```
# Validate installation
pytest testWalmartSalesTraining.py -v
pytest testWalmartSalesPrediction.py -v

# Check system status
python --version
pip check
pip list

# Kill stuck processes
pkill -f streamlit  # Linux/macOS
```

```
taskkill /F /IM python.exe  # Windows
```

9.5.2 URL Quick Reference

Cloud Applications:

- **Training App**: `https://walmart-sales-training-app-py.streamlit.app/`
- **Prediction App**: `https://walmart-sales-prediction-app-py.streamlit.app/`

Local Applications:

- **Training App**: `http://localhost:8501`
- **Prediction App**: `http://localhost:8502`

Documentation Resources:

- **Streamlit Docs**: `https://docs.streamlit.io/`
- **pmdarima Docs**: `https://alkaline-ml.com/pmdarima/`
- **Python Download**: `https://www.python.org/downloads/`

9.6 Feedback and Improvement

9.6.1 User Feedback

Providing Feedback

Types of Valuable Feedback:

- **Usability Issues**: Interface confusion or difficulty
- **Documentation Gaps**: Missing or unclear information
- **Feature Requests**: Suggestions for new functionality
- **Performance Issues**: Slow operations or resource problems

Feedback Collection Methods:

- Direct communication with system administrators
- User surveys and questionnaires
- Issue tracking systems
- Community forums and discussion boards

9.6.2 Continuous Improvement

Version Updates

Update Schedule:

- **Major Updates**: Annual releases with new features
- **Minor Updates**: Quarterly bug fixes and improvements
- **Patch Updates**: Monthly security and critical fixes
- **Documentation Updates**: As needed based on user feedback

Feature Development

Future Enhancements:

- Enhanced model ensemble capabilities
- Real-time data integration
- Advanced visualization features
- Mobile application support