r2bit Slotting Manager

A beginner-friendly Python application with a Streamlit interface for optimizing warehouse slotting operations using ABC classification and AI-powered analysis.

What is Warehouse Slotting?

Warehouse slotting is the process of determining the optimal storage location for each product (SKU) in a warehouse. Good slotting improves picking efficiency, reduces travel time, and maximizes warehouse space utilization.

Project Overview

This application helps warehouse managers:

1. **Classify products** into A, B, and C categories based on order frequency and unit volume
2. **Assign optimal locations** to products based on their classification
3. **Visualize results** with interactive charts and tables
4. **Get AI-powered insights** about slotting efficiency using OpenAI's language models

Project Structure

```
r2bit_Slotting/
├── src/
│   ├── __init__.py
│   ├── abc_classes.py     # ABC classification functionality
│   ├── slotting.py        # Core slotting functionality
│   └── llm_model_analysis.py # AI analysis using OpenAI
├── input/                 # Directory for input CSV files
├── outputs/               # Directory for output files
├── streamlit_app.py       # Streamlit web interface
├── requirements.txt       # Project dependencies
├── .env                   # Environment variables (API keys)
└── README.md              # This file
```

Input File Format

The application expects a CSV file (comma or semicolon separated) with the following columns:

Column	Description	Required
`sku`	Unique identifier for each product	Yes
`orders`	Number of orders for the SKU	Yes
`units`	Total units moved for the SKU	Yes

Column	Description	Required
`pre_slotting_location`	Current location of the SKU in format `DD.RR.BB.NN.AA` (deposit.row.block.level.apartment)	Yes
`description`	Product description	Optional

Example input file (semicolon separated):

```
sku;orders;units;pre_slotting_location;description
SKU001;120;500;01.01.01.01.01;High frequency item
SKU002;80;1200;01.01.01.01.02;High volume item
```

> **Note**: The `pre_slotting_location` column is used to track the original location of each SKU before optimization. This enables the AI analysis to compare the original and proposed locations to calculate movement efficiency improvements.

Installation Guide for Beginners

1. **Prerequisites**:
   - Python 3.8 or higher installed on your computer
   - Basic understanding of command line operations
2. **Clone or download the repository**:

```
git clone <repository-url>
cd r2bit_Slotting
```

3. **Set up a virtual environment** (recommended):

```
    # On Windows
python -m venv venv
venv\Scripts\activate

# On macOS/Linux
python -m venv venv
source venv/bin/activate
```

4. **Install the required dependencies**:

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

5. **Set up environment variables**:

- Create a file named `.env` in the project root directory
- Add your OpenAI API key to the file:

```
OPENAI_API_KEY=your_openai_api_key_here
```

- You can get an API key from OpenAI's website

How to Use the Application

Using the Web Interface

1. **Start the Streamlit application**:

```
streamlit run streamlit_app.py
```

2. **Access the web interface** in your browser (usually at http://localhost:8501)

3. **Upload your data**:

   - Prepare a CSV file with columns for SKU, order frequency, and unit volume
   - The application supports both comma "," and semicolon ";" as separators
   - Example format:

```
sku;orders;units;description
SKU001;120;500;"High frequency item"
SKU002;80;1200;"High volume item"
```

4. **Configure parameters**:

   - Set ABC classification parameters (weights and cutoffs)
   - Define warehouse layout (rows, blocks, levels, apartments)
   - Specify zone definitions

5. **Run the slotting process** and explore the results in the different tabs:

   - ABC Classification: View the distribution of SKUs across A, B, and C classes
   - Slotting Results: See where each SKU is assigned in the warehouse
   - Warehouse Statistics: Analyze zone utilization and other metrics
   - AI Analysis: Get AI-powered insights about your slotting strategy

Using the Command Line

You can also run the individual components from the command line:

1. **ABC Classification**:

```
python -c "from src.abc_classes import classify_abc;
classify_abc('input/your_file.csv', 'outputs/abc_results.csv', 'sku',
'orders', 'units')"
```

2. **Slotting**:

```
python -c "from src.slotting import perform_slotting;
perform_slotting('outputs/abc_results.csv', 'outputs/slotting_results.csv')"
```

3. **AI Analysis**:

```
python src/llm_model_analysis.py outputs/slotting_results.csv --abc-file
outputs/abc_results.csv
```

Key Features

1. ABC Classification

- **What it does**: Categorizes products based on their importance
  - **Class A**: High-value items (typically top 80% of value)
  - **Class B**: Medium-value items (typically next 15% of value)
  - **Class C**: Low-value items (typically remaining 5% of value)
- **Customizable parameters**: Adjust weights for order frequency and unit volume

2. Warehouse Slotting

- **What it does**: Assigns each SKU to an optimal warehouse location
- **Location format**: DD.RR.BB.NN.AA (Deposit.Row.Block.Level.Apartment)
- **Zone-based allocation**: Places Class A items in Zone A (closest to picking area), etc.
- **Customizable warehouse layout**: Define the size and structure of your warehouse

3. Interactive Visualization

- **Charts and tables**: Visualize ABC classification and zone distribution
- **Cross-tabulation**: Analyze the relationship between ABC classes and zones
- **Warehouse statistics**: View utilization metrics for each zone

4. AI-Powered Analysis

- **What it does**: Uses OpenAI's language models to analyze slotting results
- **Insights provided**: Efficiency improvements, optimization opportunities, and implementation recommendations
- **Language**: Analysis is provided in Portuguese

Troubleshooting

Common Issues

1. **OpenAI API Key Issues**:
    - Ensure your API key is correctly set in the `.env` file
    - The key should not have any spaces, quotes, or newlines
    - If using the web interface, you can also enter the API key directly
2. **File Format Issues**:
    - Make sure your CSV file has the required columns (SKU, orders, units)
    - Check that the separator is either comma "," or semicolon ";"
3. **Session State Issues**:
    - If the AI analysis disappears after running, this is a known issue that has been fixed
    - The analysis should now persist between page loads

For Developers

Core Components

1. **ABCClassifier** (`src/abc_classes.py`):
    - Classifies SKUs based on order frequency and unit volume
    - Calculates a combined score using weighted normalization
2. **SlottingManager** (`src/slotting.py`):
    - Manages warehouse locations and zone definitions
    - Assigns SKUs to locations based on their ABC classification
3. **SlottingAnalyzer** (`src/llm_model_analysis.py`):
    - Analyzes slotting results using OpenAI's language models
    - Generates insights and recommendations in Portuguese

Extending the Application

To add new features:

1. Modify the core classes in the `src` directory
2. Update the Streamlit interface in `streamlit_app.py` to expose the new features
3. Add any new dependencies to `requirements.txt`

License

Acknowledgments

- This project uses OpenAI's API for AI-powered analysis
- Built with Streamlit, Pandas, and other open-source libraries