**Sales Forecasting & AI Insights Application**

**1. Introduction & Overview**

Welcome to the Sales Forecasting & AI Insights Application! This tool is designed to empower users, from small business owners to data analysts, by providing accessible sales forecasting capabilities coupled with actionable business insights generated by artificial intelligence. In a dynamic market, understanding future sales trends is crucial for inventory management, resource allocation, marketing strategies, and overall business planning. This application aims to simplify this process.

**Key Features:**

* **Flexible Data Input:** Users can provide historical sales data either by uploading a CSV file or through manual comma-separated text input.
* **Powerful Forecasting Engine:** Utilizes Facebook's Prophet library, a robust time series forecasting model designed to handle common business data characteristics like trends, seasonality, and holidays (though holiday specification is a future development).
* **Customizable Data Visualization:** Forecasts and actual sales are presented on an interactive line chart using Chart.js. Users can choose to view historical actuals as:
  + Daily (smoothed with a 7-day Simple Moving Average)
  + Weekly Totals
  + Monthly Totals The prediction granularity automatically matches the selected actuals display.
* **Extended Forecast Horizon:** Predicts future sales for approximately 3 months (91 days) beyond the last provided data point.
* **AI-Generated Business Insights:** Leverages the DeepSeek AI API to analyze the forecast data and provide concise, actionable business insights related to trends, opportunities, and potential risks.
* **Interactive AI Chat:** Users can engage in a follow-up conversation with the AI directly on the results page to ask clarifying questions or delve deeper into the forecast and insights.
* **User-Friendly Web Interface:** Built with FastAPI for the backend and simple HTML, CSS, and JavaScript for the frontend, making it accessible via a web browser.

**Target Audience:**

This application is primarily aimed at:

* Small to medium-sized business owners seeking to understand future sales.
* Sales and marketing teams planning campaigns and inventory.
* Financial analysts needing quick sales projections.
* Students and individuals learning about time series forecasting and AI applications.

**Technology Stack:**

* **Backend:** Python 3.x, FastAPI (web framework), Uvicorn (ASGI server).
* **Forecasting:** Prophet (time series forecasting), Pandas (data manipulation and aggregation).
* **AI Integration:** DeepSeek API (for business insights and chat).
* **Frontend:** HTML5, CSS3, JavaScript.
* **Charting:** Chart.js (JavaScript charting library).
* **Environment Management:** python-dotenv (for API key management).

**2. Getting Started**

Follow these steps to set up and run the application on your local machine.

**Prerequisites**

* **Python:** Version 3.8 or higher installed on your system.
* **pip:** Python package installer (usually comes with Python).
* **Web Browser:** A modern web browser (e.g., Chrome, Firefox, Edge, Safari).
* **DeepSeek API Key:** You will need an API key from DeepSeek to use the AI insights and chat features. Visit <https://platform.deepseek.com/> to obtain one.

**Installation**

1. Project Files:

Ensure you have all the project files in a single directory on your computer:

* + sales\_forecasting\_api.py (the main Python backend script)
  + static/ (directory)
    - index.html (the main input page)
    - result.html (the page to display results)
  + requirements.txt (lists Python dependencies)
  + .env (you will create this for your API key)
  + sales\_data.csv (optional sample data for initial reference)

1. Create a Virtual Environment (Recommended):

It's good practice to create a virtual environment to manage project dependencies separately.

Open your terminal or command prompt, navigate to the project directory, and run:

Bash

python -m venv venv

Activate the virtual environment:

* + On1 macOS/Linux: source venv/bin/activate
  + On Windows: venv\Scripts\activate

1. Install Dependencies:2

With the virtual environment activated, install the required Python packages using pip and the requirements.txt file:

Bash

pip install -r requirements.txt

This will install FastAPI, Uvicorn, Prophet, Pandas, Requests, python-dotenv, Jinja2, and python-multipart.

1. Configure API Key:

Create a file named .env in the root of your project directory (the same level as sales\_forecasting\_api.py). Add your DeepSeek API key to this file:

Code snippet

DEEPSEEK\_API\_KEY=your\_actual\_deepseek\_api\_key\_here

Replace your\_actual\_deepseek\_api\_key\_here with the actual API key you obtained from DeepSeek.

**Initial Data (**sales\_data.csv**) - Optional Reference**

The application includes an optional sales\_data.csv file. This file was initially intended for training a base reference model. However, the current application logic primarily focuses on fitting a new Prophet model for each user request based on the data they provide in that session.

If you wish to include it for reference, ensure it has at least two columns: Date (in a recognizable date format like YYYY-MM-DD) and Sales\_Revenue (numeric values). The application will attempt to load this at startup but will function without it if it's not found or invalid.

**3. Running the Application**

Once the setup is complete, you can run the application using the Uvicorn ASGI server.

1. Start the Server:

Open your terminal or command prompt, ensure your virtual environment is activated, and navigate to the root directory of the project. Run the following command:

Bash

uvicorn sales\_forecasting\_api:app --reload --port 8000

* + sales\_forecasting\_api:app: Tells Uvicorn to find the FastAPI application instance named app inside the sales\_forecasting\_api.py file.
  + --reload: Enables auto-reloading, so the server will restart automatically if you make changes to the Python code (useful for development).
  + --port 8000: Specifies that the application should run on port 8000. You can change this if needed.

1. Access in Browser:

Once Uvicorn is running (you'll see output like INFO: Uvicorn running on http://127.0.0.1:8000), open your web browser and navigate to:

http://127.0.0.1:8000

You should now see the Sales Forecasting App's index page.

**4. User Guide**

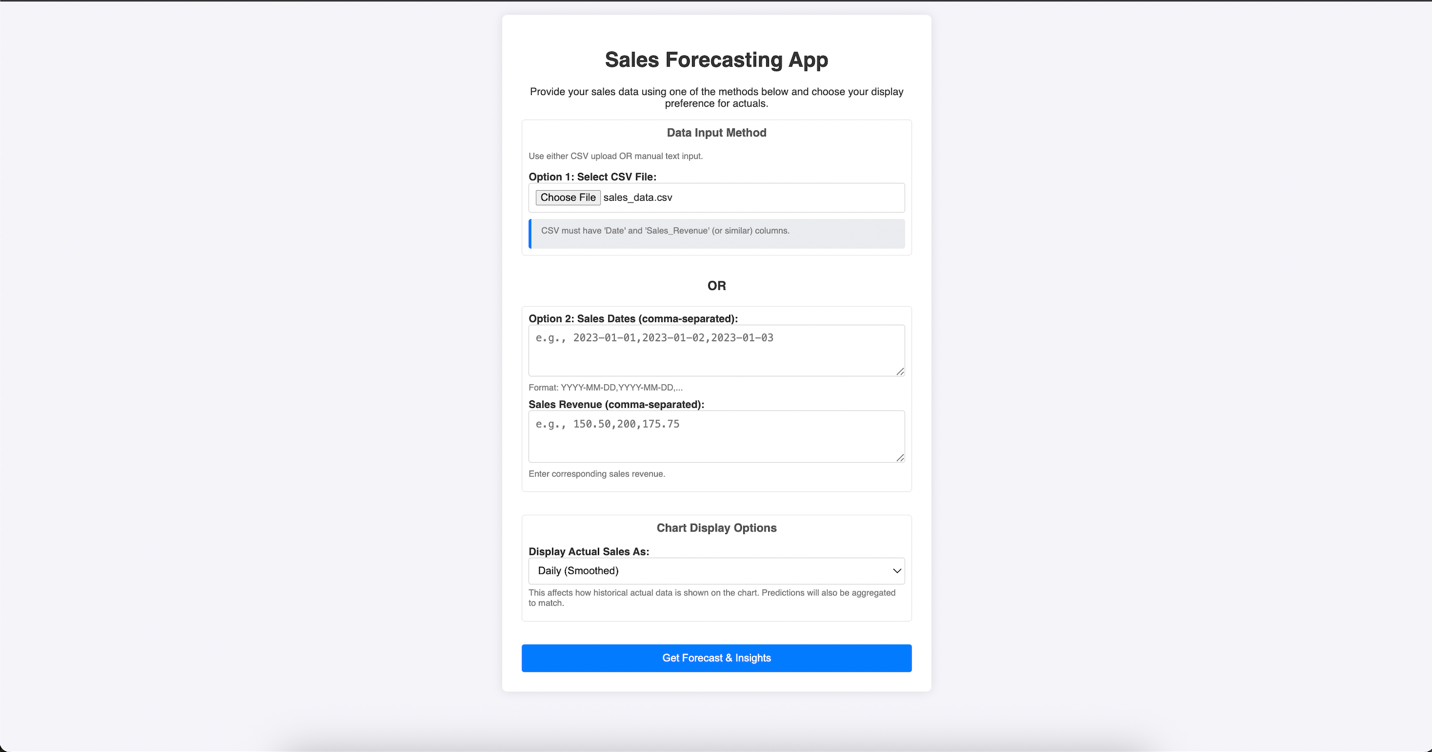
The application consists of two main pages: the input page (index.html) and the results page (result.html).

**Index Page (**index.html**)**

This is where you provide your historical sales data and choose display preferences.

* **Data Input Methods:** You have two options for providing data:
  + **Option 1: Select CSV File:**
    - Click the "Choose File" button to upload a CSV file from your computer.
    - **CSV Format Requirements:**
      * The file must be a valid .csv file.
      * It must contain a header row.
      * It needs at least two columns: one for dates and one for sales revenue.
      * Common column names like "Date", "date", "DS", "ds" for dates, and "Sales\_Revenue", "Sales Revenue", "Revenue", "Sales", "Y", "y", "Value" for sales figures are recognized.
      * Dates should be in a format that Pandas can parse (e.g., YYYY-MM-DD, MM/DD/YYYY).
  + **Option 2: Manual Text Input:**
    - **Sales Dates:** Enter dates in the "Sales Dates" text area, separated by commas. The recommended format is YYYY-MM-DD.
    - **Sales Revenue:** Enter the corresponding sales revenue figures in the "Sales Revenue" text area, also separated by commas. These should be numeric values.
    - Ensure the number of dates matches the number of sales revenue entries.
  + You should use *either* CSV upload *or* manual text input, not both simultaneously. If a CSV is uploaded, it will be prioritized.
* **Chart Display Options:**
  + **Display Actual Sales As:** This dropdown menu allows you to choose how your historical ("actual") sales data is presented on the forecast chart on the results page. The options are:
    - Daily (Smoothed): Shows daily actual sales data, smoothed with a 7-day Simple Moving Average (SMA) to reduce noise.
    - Weekly Totals: Aggregates your daily actual sales into weekly totals.
    - Monthly Totals: Aggregates your daily actual sales into monthly totals.
  + The "Predicted Future Sales" on the chart will also be aggregated to match the granularity you select here.
* **Submit:** Once you've provided your data and selected your display preference, click the "Get Forecast & Insights" button to generate the forecast.

(Placeholder for Image of Index Page - A view of the main data input page with options for CSV upload, manual entry, and display aggregation selection.)



**Results Page (**result.html**)**

After submitting your data, you'll be taken to the results page, which displays the forecast, AI insights, and an interactive chat.

* **Sales Performance & Forecast Chart:**
  + This chart visualizes your historical sales performance and the future forecast.
  + **Actual Sales Line:** Displayed in dark blue. The label will indicate if it's "Actual Sales (Smoothed)", "Actual Sales (Weekly Totals)", or "Actual Sales (Monthly Totals)" based on your selection on the index page.
  + **Predicted Future Sales Line:** Displayed in light blue/cyan. This line represents the sales forecast for approximately the next 3 months (91 days) from the end of your input data. Its label will also reflect the aggregation level.
  + **Connected Lines:** The predicted line starts from the end of the actuals period, using Prophet's in-sample fit for the last historical point to ensure a visual connection.
  + **Axes:** The X-axis represents "Date," and the Y-axis represents "Sales Revenue." The date ticks on the X-axis are adjusted based on the selected aggregation level to maintain clarity.
  + **Interactivity:** You can hover over the chart lines to see tooltips with specific date and sales values.
* **AI-Generated Business Insights:**
  + Below the chart, you'll find a section with 3 business insights generated by the DeepSeek AI.
  + These insights are derived from an analysis of the last 30 days of your forecast data (from the 91-day daily forecast).
  + The purpose is to highlight key trends, potential opportunities (e.g., upward trends, seasonal peaks), or risks (e.g., predicted downturns, high volatility) apparent in the forecast, offering actionable starting points for your business decisions.
* **Continue Chat with AI:**
  + This section allows you to ask follow-up questions about the forecast or the initial insights.
  + Type your question into the input field and click "Send."
  + The AI's response will appear in the chat display area.
  + The conversation history (your questions and AI responses for the current session) is maintained and sent with each new question to provide context to the AI.
* **Navigation:**
  + A "Back to Input New Data" button allows you to easily return to the index page to enter new data or change display options.

(Placeholder for Image of Results Page - Shows the forecast chart with smoothed/aggregated actuals, connected predictions, AI insights, and the chat interface. Consider one for daily and one for monthly view if different enough.)



**5. Technical Details & Architecture**

This section provides a brief overview of the application's internal workings.

**Backend (**sales\_forecasting\_api.py**)**

* **Framework:** Built using FastAPI, a modern, fast (high-performance) web framework for building APIs with Python.
* **Static Files & Templating:** FastAPI serves the HTML, CSS, and any client-side JavaScript from the static/ directory. Jinja2 is used for templating to inject dynamic data into result.html.
* **Endpoints:**
  + @app.get("/"):
    - Serves the index.html page.
  + @app.post("/result"):
    - This is the core endpoint for generating forecasts.
    - **Input Handling:** Accepts form data including manually entered dates/sales, an optional uploaded CSV file, and the display\_aggregation choice.
    - **Data Parsing & Validation:** Cleans and validates the input sales data, converting it into a Pandas DataFrame with 'ds' (datetime) and 'y' (numeric sales) columns. This raw daily data is sorted.
    - **Prophet Model Fitting:** An instance of the Prophet model is created and fitted using the *entire raw daily* user\_df. This ensures the model learns from the finest granularity of data available.
    - **Daily Forecast Generation:** Prophet generates a daily forecast for the next 91 days, including its fit for the historical period (include\_history=True).
    - **Actuals Data Preparation for Chart:**
      * Based on display\_aggregation:
        + daily\_smoothed: A 7-day Simple Moving Average (SMA) is applied to the raw daily 'y' values. The last MAX\_ACTUAL\_POINTS\_TO\_DISPLAY\_DAILY (e.g., 90) points are then selected.
        + weekly\_sum: Raw daily 'y' values are resampled to weekly sums. The last MAX\_ACTUAL\_POINTS\_TO\_DISPLAY\_WEEKLY (e.g., 52) weeks are selected.
        + monthly\_sum: Raw daily 'y' values are resampled to monthly sums. The last MAX\_ACTUAL\_POINTS\_TO\_DISPLAY\_MONTHLY (e.g., 24) months are selected.
      * This processed data becomes actual\_sales\_for\_chart.
    - **Predictions Data Preparation for Chart:**
      * The full daily forecast output (daily\_forecast\_output\_all) from Prophet is taken.
      * Predictions start from the last date of the raw actual data to ensure visual connection.
      * This daily prediction series is then aggregated (summed) to match the display\_aggregation choice (weekly or monthly totals), or kept daily if daily\_smoothed was chosen.
      * This processed data becomes future\_predictions\_for\_chart.
    - **DeepSeek API for Initial Insights:** The tail end of the daily forecast data (typically the last 30 daily points, from which the last 10 are then sampled) is sent to the DeepSeek API with a prompt asking for 3 actionable business insights.
    - **Template Rendering:** The processed actuals, predictions, AI insights, and aggregation choice are passed to the result.html template.
  + @app.post("/chat\_with\_ai"):
    - Handles AJAX requests from the chat interface on result.html.
    - Receives the current conversation history (an array of user and assistant messages).
    - Sends this history to the DeepSeek chat completions API.
    - Returns the AI's new response as JSON.

**Frontend (**static/ **directory)**

* index.html**:** A simple HTML form for data input and selecting display preferences.
* result.html**:**
  + Uses Jinja2 templating to display dynamic data passed from the backend.
  + **Chart.js:** Implements the "Sales Performance & Forecast" chart. JavaScript code within this file configures Chart.js, including setting up datasets for actuals and predictions, customizing colors, line styles (tension, point radius), and axes (time scale, dynamic unit based on aggregation, tick limits for clarity).
  + **JavaScript for AI Chat:** Handles capturing user input from the chat box, sending it via fetch (AJAX) to the /chat\_with\_ai backend endpoint, and appending both user messages and AI responses to the chat display area.

**Forecasting Model (Prophet)**

* Prophet is an open-source forecasting tool developed by Facebook, designed for business time series that often have trends, multiple seasonalities (yearly, weekly, daily), and holiday effects.
* It's generally robust to missing data and shifts in trends and can handle outliers well.
* In this application, it's fitted on the raw daily data provided by the user for each session to capture underlying patterns.
* The forecast horizon is currently set to 91 days (approximately 3 months).

**Data Handling**

* **Pandas:** The Pandas library is used extensively in the backend for:
  + Creating DataFrames from user input (CSV or text).
  + Data cleaning, type conversion (dates, numbers), and handling missing values.
  + Resampling daily data into weekly or monthly totals/sums.
  + Calculating Simple Moving Averages (SMA).

**6. Troubleshooting Common Issues**

* **Indentation Errors (Python):** Python relies strictly on indentation. If you get an IndentationError when starting the server, carefully check the lines mentioned in the traceback in sales\_forecasting\_api.py. Ensure consistent use of spaces (typically 4) for indentation and that code blocks under if, elif, else, def, class, try, except, etc., are correctly indented.
* **API Key Not Working:**
  + Ensure you have a .env file in the root project directory.
  + Verify that DEEPSEEK\_API\_KEY=your\_key is correctly entered in the .env file with your actual key.
  + Make sure python-dotenv is installed and load\_dotenv() is called at the beginning of sales\_forecasting\_api.py.
* **Data Format Errors:**
  + **CSV:** Ensure your CSV has a header and includes columns that can be recognized as dates and sales revenue. Dates should be in a consistent, parsable format.
  + **Manual Input:** Check for correct comma separation for both dates and sales. Ensure the number of date entries matches the number of sales entries. Use YYYY-MM-DD for dates.
* **Forecast Looks Unexpected (e.g., Flat, Very Low, or Lines Not Connecting):**
  + **Flat/Low Forecast:** As discussed, Prophet's forecast is heavily influenced by recent trends in your input data. A sharp, sustained drop at the end of your historical data will likely lead to a forecast of continued low values. Review your input data for accuracy, especially recent periods. Check the terminal logs for the actual yhat values predicted by the backend.
  + **Lines Not Connecting:** This was addressed by ensuring the prediction data passed to the chart starts from the same period as the end of the actuals. The current aggregation logic aims to maintain this.
* **Chart X-Axis Crowded:** If date labels are too close:
  + The maxTicksLimit in Chart.js options in result.html helps.
  + For very long time series displayed daily, consider choosing "Weekly Totals" or "Monthly Totals" on the index page, which will also change the chart's x-axis unit for better readability.
* **"Not Found" Error for Chat:** Ensure the @app.post("/chat\_with\_ai") endpoint is correctly defined in sales\_forecasting\_api.py and the server has been restarted after any changes.

**7. Future Developments**

This application provides a solid foundation for sales forecasting. Here are some potential areas for future enhancement:

**Application Enhancements**

* **User Accounts & Saved History:** Allow users to create accounts, save their uploaded datasets, and store past forecasts for comparison and tracking.
* **Advanced Prophet Configurations:** Expose more of Prophet's parameters to the user via the UI, such as:
  + Specifying yearly, weekly, and daily seasonality modes (additive/multiplicative).
  + Adjusting changepoint sensitivity or manually adding changepoints.
  + Adding country-specific holidays.
* **External Regressors:** Allow users to upload additional time series data (e.g., marketing spend, promotions, weather) to be used as regressors in the Prophet model for more nuanced forecasts.
* **Enhanced Error Handling & Feedback:** Provide more specific and user-friendly error messages for data input issues or API failures.
* **Download Options:** Allow users to download:
  + The forecast data (actuals, predictions, upper/lower bounds) as a CSV or Excel file.
  + The generated chart as an image (PNG/JPG).
* **Interactive Chart Features:**
  + Implement zoom and pan capabilities for the chart.
  + Allow users to toggle the visibility of actuals vs. predictions, or confidence intervals.
* **Customizable Smoothing/Aggregation:**
  + Let users choose the window for SMA if "Daily (Smoothed)" is selected.
  + Offer different aggregation methods (e.g., mean, median) in addition to sum for weekly/monthly views.
* **Direct Fitting on Aggregated Data:** Provide an option to fit the Prophet model directly on weekly or monthly aggregated data if the user prefers this approach (this would change how Prophet learns seasonality).
* **UI/UX Improvements:**
  + Implement a dark mode or theme selection.
  + Improve overall visual design and responsiveness.
* **Internationalization/Localization:** Support multiple languages for the UI.

**AI Enhancements**

* **Fine-Tuned AI Prompts:** Develop more sophisticated prompts for DeepSeek to generate even more targeted and context-aware business insights based on different forecast characteristics (e.g., high growth, decline, high uncertainty).
* **AI-Assisted Interpretation:** Allow users to ask the AI (via chat) specific questions about *why* the forecast looks a certain way, and have the AI provide explanations based on Prophet's components (trend, seasonality).
* **AI-Powered Data Cleaning Suggestions:** Explore using AI to identify potential outliers or anomalies in the uploaded historical data and suggest corrections.

**Deployment**

* **Containerization:** Package the application using Docker for easier deployment and scalability.
* **Cloud Deployment:** Deploy the application to cloud platforms such as Heroku, AWS (e.g., Elastic Beanstalk, Fargate), Google Cloud (e.g., App Engine, Cloud Run), or Azure App Service.

**Mobile App Development (Conceptual)**

A mobile application could provide on-the-go access to forecasting capabilities.

* **Platform Choice:**
  + **Native:** Swift for iOS, Kotlin for Android (best performance and platform integration).
  + **Cross-Platform:** React Native, Flutter (faster development for both platforms, single codebase).
* **Key Considerations:**
  + **Simplified UI/UX:** Design specifically for smaller screens and touch interaction.
  + **Offline Capabilities:** Potentially store the last generated forecast locally for offline viewing.
  + **API Reliance:** The mobile app would likely still rely on the Python backend for the heavy lifting of forecasting and AI calls.
  + **Data Input:** Mobile data input needs to be streamlined (e.g., simplified manual entry, linking to cloud storage like Google Drive/Dropbox for CSVs, or even voice input for quick notes).
* **Potential Mobile App Features:**
  + Dashboard for a quick overview of the latest forecast.
  + Viewing charts (optimized for mobile).
  + Reading AI insights.
  + Mobile-friendly AI chat interface.
  + Push notifications (e.g., if a scheduled forecast feature were added).
  + Ability to share forecast summaries or charts easily.

**Advanced Analytics & Reporting**

* **Forecast Comparison:** Allow users to generate multiple forecast scenarios (e.g., with different assumptions or regressors) and compare them side-by-side.
* **Forecast Accuracy Tracking:** If users can input actual sales figures after a forecast period, implement metrics (e.g., MAPE, MAE, RMSE) to track forecast accuracy over time.
* **Automated Reporting:** Generate periodic PDF reports summarizing forecasts and key insights.

**8. Conclusion**

The Sales Forecasting & AI Insights Application offers a powerful yet accessible way to peer into future sales trends and gain valuable business intelligence. By combining the statistical strength of Prophet with the analytical capabilities of modern AI, it provides users with not just a prediction, but also a starting point for strategic decision-making.

We encourage you to explore its features, provide your data, and see how it can benefit your planning processes. Future developments aim to make this tool even more robust, configurable, and integrated into various business workflows. Your feedback is invaluable in shaping its evolution.