**🛍️ Sales Forecasting with ETL and Machine Learning**

**📖 Project Overview**

This project demonstrates an **end-to-end data pipeline** that covers:

* **ETL (Extract, Transform, Load)** operations to generate, clean, and load sales data into a database.
* **Sales forecasting** using **Machine Learning** (Facebook Prophet).
* **Orchestration** using **Apache Airflow**.

The project is applied to **perfume sales data** but can easily be extended to real-world e-commerce platforms like Shopify, WooCommerce, or Amazon Sellers.

**⚡ Real-World Problem Solved**

**Businesses need to predict future sales** to:

* Manage **inventory** and **avoid stockouts** or **overstocking**.
* **Optimize marketing strategies** and **seasonal promotions**.
* **Plan resources** like packaging, logistics, and staff.

Manual forecasting is error-prone and reactive. This project **automates sales forecasting** based on **historical data**, enabling **data-driven decision making** and **proactive planning**.

**🛠️ Project Structure**

| **File** | **Purpose** |
| --- | --- |
| perfumesales\_pipeline.py | **ETL Pipeline**: Generate (fake) sales data and load it into a SQL Server database. |
| perfumesales\_forecast.py | **ML Forecasting Pipeline**: Train a model and forecast future perfume sales. |
| perfume\_sales\_data.csv | Sample generated sales data. |

Both pipelines are orchestrated using **Apache Airflow DAGs**.

**🔥 Detailed Code Walkthrough**

**1. ETL Pipeline (perfumesales\_pipeline.py)**

**a. generate\_fake\_data()**

* **Goal**: Simulates **100 fake perfume sales records**.
* **Tools**: Uses the Faker library to generate random perfume names, sale dates, units sold, and revenues.
* **Output**: Saves the generated data into a CSV file perfume\_sales\_data.csv.

**b. create\_connection()**

* **Goal**: Establish a **connection to a Microsoft SQL Server** using pyodbc.

**c. load\_data\_into\_sql()**

* **Goal**: Load the generated CSV data into a **SQL Server table** named perfume\_sales.
* **Process**:
  + Read the CSV.
  + For each row, insert the record into SQL Server.

**d. Airflow DAG**

* **DAG Name**: perfume\_sales\_pipeline
* **Tasks**:
  1. **generate\_data** → Generate fake sales data.
  2. **load\_data** → Load the data into SQL Server.
* **Schedule**: Runs **once daily**.

**2. Forecasting Pipeline (perfumesales\_forecast.py)**

**a. load\_data\_smart()**

* **Goal**: Load and preprocess the sales CSV for modeling.
* **Features**:
  + Automatically detects date and numeric sales columns.
  + Renames columns to match Prophet’s expected format: ds (date) and y (value).
  + Also keeps the brand name for filtering.

**b. find\_best\_match(name, available\_brands)**

* **Goal**: Perform **fuzzy matching** to find the closest brand name entered by the user.

**c. train\_forecast\_model()**

* **Goal**: Train a **Facebook Prophet** model for the selected perfume brand.
* **Process**:
  1. User inputs the perfume name.
  2. Finds the closest matching brand.
  3. Trains a Prophet model.
  4. Forecasts sales for the next **60 days**.
  5. Saves forecast results and plot.

**d. Airflow DAG**

* **DAG Name**: perfume\_sales\_forecast
* **Tasks**:
  + **generate\_sales\_forecast** → Train model and generate future predictions.
* **Schedule**: Runs **daily at 2 AM**.

**📊 Sample Outputs**

* forecast\_result.csv: Contains predicted sales with confidence intervals.
* forecast\_plot.png: A line graph showing historical sales and future forecasts.

**📦 How To Run**

Ensure you have Airflow, Python 3.8+, and necessary libraries installed.

**1. Set up Airflow**

pip install apache-airflow

**2. Install project dependencies**

pip install pandas faker pyodbc matplotlib prophet

**3. Set up SQL Server**

* Install and configure SQL Server.
* Create a database named PerfumeSales.
* Create a table:

CREATE TABLE perfume\_sales (

PerfumeName VARCHAR(100),

SaleDate DATE,

UnitsSold INT,

Revenue FLOAT

);

**4. Place DAG files**

* Put perfumesales\_pipeline.py and perfumesales\_forecast.py in your Airflow dags/ folder.

**5. Run Airflow**

airflow standalone

Access Airflow UI at http://localhost:8080.

Trigger the two DAGs:

* perfume\_sales\_pipeline
* perfume\_sales\_forecast

**🔥 Real-World Extensions**

* Replace fake data with real sales from Shopify, Amazon, etc.
* Use cloud databases like Azure SQL or AWS RDS.
* Deploy forecasts to dashboards (PowerBI, Tableau, Flask apps).
* Automate inventory alerts.

**📢 Final Notes**

* ETL + Forecasting Pipelines are modular.
* Easily scalable for more brands and products.
* Provides critical insights for e-commerce and retail businesses.

**🚀 Let's Predict the Future, One Sale at a Time!**