National College of Ireland

Empirical Evaluation of an Integrated Forecast to Inventory Planning Tool for Retail Service Levels and Inventory Efficiency

MSc Research Practicum
MSc. Data Analytics

Indrajeet Diwane
Student ID: X23271434

School of Computing
National College of Ireland

Supervisor: Prof. Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Indrajeet Diwane

Student ID: X23271434

Programme: MSc. Data Analytics **Year:2025**

Module: MSc. Research Practicum

Lecturer: Prof. Vikas Sahni

Submission Due Date: 11Th Dec 2025

Project Title: Empirical Evaluation of an Integrated Forecast-to-Inventory Planning Tool for Retail Service Levels and Inventory Efficiency.

Word Count: 3359 **Page Count: 12**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Indrajeet Diwane

Date: 11th Dec 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Indrajeet Diwane
Student ID: x23271434

1 Introduction

The current configuration manual is a comprehensive guide to installation, configuration, and operationalization of the entire set of components of the MSc research project, entitled "Empirical Evaluation of an Integrated Forecast-to-Inventory Planning Tool to Retail Service Levels and Inventory Efficiency." The project applies an integrated workflow that integrates the demand forecasting and the inventory control simulation to assess the effects of better forecasts on the level of retail services (e.g., fill rate and cycle service level) as well as maximizes inventory efficiency. The system is built with Python scripts and data processing and forecasting with Jupyter Notebooks and interactive visualization of results with Microsoft power BI dashboard. With this manual, a user with sufficient technical literacy can replicate the entire end-to-end system on their own user machine - not just preparing the dataset, but also creating SKU-level forecasts and inventory policy metrics, and viewing the results on the Power BI dashboard.

2 System Requirements

This section contains the description of all hardware, software and data requirements of the project. The all-in-one character of the tool implies that a Python execution environment, as well as a Windows-based business intelligence environment, are required, and the corresponding retail dataset files. The suggested setup will also guarantee a seamless installation and maximum performance of both the forecasting and simulation routines, and the Power BI dashboard.

2.1 Hardware Requirements

Component	Minimum Requirement	Actual Environment Used	Recommended Requirement
CPU	Intel i5 (or AMD equivalent)	Intel Core™ i7-class processor used for development and testing	Intel Core™ i7-class (or equivalent) for smoother execution
RAM	8 GB	16 GB used	16 GB recommended, especially for larger datasets or running multiple forecasting models
Storage	1 GB free space	Project operated with sufficient local storage for dataset, intermediate files, outputs, and software	A few GB free space recommended to accommodate datasets, files, outputs, and software installations
Operating System	Windows 10 or Windows 11 (64-bit) required for Power BI Desktop	Windows 11(64-bit)	Windows 10/11 (64-bit) for full system (Python + Power BI) Windows VM required if using macOS/Linux for dashboard
Internet Connection	Required initially for installing Python, VS Code, Power BI and libraries	Stable internet used during setup and tooling installation	Not required for routine execution after setup Required to download tools and obtain the original dataset if regenerating samples

Table 1: Hardware requirements

2.2 Software Requirements

Component	Minimum Requirement	Actual Environment Used	Recommended Requirement
Python Environment	Python 3.14 installed	Python 3.14.0	Python 3.14 with a dedicated virtual environment to manage libraries
Jupyter Support	Jupyter Notebook capability available via installed tools	Jupyter Notebook execution via VS Code	Use Jupyter via VS Code for a consistent run workflow and smoother integration with the project structure
Development Tools (IDE)	Any Python IDE capable of running scripts/notebooks	VS Code with Python extension	VS Code + Python extension (latest version as of 2025)
Microsoft Power BI Desktop	Power BI Desktop installed on Windows (AUG2025)	Power BI Desktop (NOV2025)	Power BI Desktop (NOV2025)
Microsoft Excel (optional)	Excel Office 2016	Excel Office 2019	Excel (Office 2016+ or Excel 365)

Table 2: Software requirements

2.3 Python Libraries

The code requires a number of Python libraries to run. The key libraries and versions to be used in development are:

Libraries	Actual Version Used
pandas	version family: **2.3.3**
NumPy	version family: **2.3.4**
scikit-learn	version family: **1.7.2**
statsmodels	version family: **0.14.5**
Matplotlib	version family: **3.10.7**
Seaborn	version family: **0.13.2**
pmdarima (optional)	Optional dependency used for auto_arima functionality Required only if the implementation performs automated ARIMA order selection per SKU If not explicitly used, statsmodels alone is sufficient for ARIMA modelling
Standard library and other optional packages	Libraries such as pathlib (file paths) and math are part of Python's standard library and require no installation If referenced in the implementation, install additional packages as needed (e.g., openpyxl for Excel I/O or numpy-financial for EOQ-related calculations)

Table 3: Python libraries

Installation of all the necessary libraries is possible through pip. E.g. once the virtual environment of the project has been activated, run:

➔ *pip install pandas numpy scikit-learn statsmodels pmdarima matplotlib seaborn*

```
import os
import numpy as np
import pandas as pd

from statsmodels.tsa.arima.model import ARIMA
from sklearn.ensemble import RandomForestRegressor

print("Libraries imported successfully.")
```

Fig 1: Python libraries used in code

2.4 Data Requirements

Dataset File Name	Description
Retail Dataset (Instacart sample)	• A sample based on the Instacart Online Grocery Shopping Dataset 2017 is used in the analysis. • This publicly available dataset includes anonymized transactional information of more than 3 million grocery orders made by a US-based online retailer (Instacart). • The entire dataset is extensive, covering a number of several million rows in a number of CSV files. • A subset/sample is selected to ensure the project is manageable at a local environment.
orders_sample.csv	• A customer order sample (e.g., 120, 000 orders sampled out of the complete dataset). • A row is order and contains these fields like order ID, customer ID, order date time etc. • Sampling was done randomly using a fixed seed to make it reproducible.
order_products__prior_sample.csv	• The Instacart prior dataset is sampled on its order line items. • Every row is a product in an order (e.g., order ID, product ID, add-to-cart order, etc.) • Retrieves records of the sampled set of order_id. • Has a capacity of hundreds of thousands of rows, based on the quantity of items in sampled orders.
order_products__train_sample.csv	• The sampled orders of the Instacart "train" dataset have the order line items. • Instacart divides the data into prior, train, and test; in this project prior + train will be used as historical data. • On the existence of the original file, the order/products/train.csv is filtered with the script. • Selects all the product line items in which order_id is in orders_sample.csv.
Optional Full Dataset	• To regenerate the sample or test the tool on the full dataset, those users have to receive the original Instacart files (though Instacart official source or Kaggle; see References). • The initial dataset usually consists of: orders.csv, order_products_prior.csv, order_products_train.csv, as well as metadata files, i.e. products.csv, aisles.csv and departments.csv. • Such files are not necessary to get the project going with the given sample. • The Dataset_Samples.py script can be used to create a new sample, should one want to. • In order to make use of it, change the script settings, particularly, DATADIR (and possibly NO_RDERS) to the place where the raw dataset is located. • The sampling process (randomness (selecting) of around 120,000 orders) and filtering of the line items provide a coherent subset to be used in the experiment. • Working directly with the full dataset in forecasting will also consume a lot of runtime and resources, and thus it is advisable to initially work with the sample.

Table 4: Data requirements

3 Environment Setup

This part covers the process of setting up the local environment with the integrated forecast-to-inventory planning tool. It includes Python installation, project folder structure, sample data creation, and the proper sequence of execution of forecasting and inventory simulation to ensure the important output files are created and used in Power BI.

3.1 Python Environment and Project Folder Structure

3.1.1 Python Installation

Install **Python 3.14** on your system.
On Windows, use the official installer and ensure Python is added to **PATH.**
Confirm installation by running:
python --version (should show 3.14.x).

3.1.2 Virtual Environment Setup

Create an isolated environment inside the project directory:
python -m venv venv
Activate it:
Windows: .\venv\Scripts\activate
Linux/Mac: source venv/bin/activate
Check your terminal to see if it shows the environment prefix (e.g., (venv)).
Activate it:

3.1.3 Library Installation

Install required packages after activating the venv:
→ pip install pandas numpy scikit-learn statsmodels pmdarima matplotlib seaborn

Install the required libraries in the activated environment, as specified in the project setup notes.

3.1.4 Folder Structure Reference

Develop and sustain the suggested layout in order to have scripts and notebooks find the inputs and store the outputs uniformly.

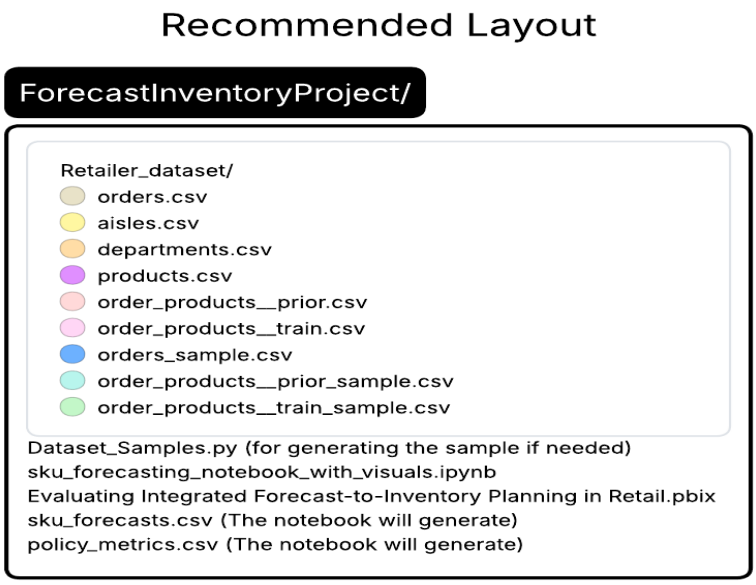

Fig 2: Folder structure layout

3.1.5 VS Code & Notebook Readiness

Open the project folder in **VS Code**.
Install the **Python extension** if prompted.
Select the **venv interpreter** in VS Code so execution uses the correct libraries.
Open the notebook and confirm it renders as a Jupyter notebook with runnable cells.

3.1.6 Critical Path Configuration

Update the DATA_DIR path in Dataset_Samples.py so it points to your local retailer_dataset folder.

➔ Example:
DATA_DIR =
Path(r"C:\Users\YourName\ForecastInventoryProject\retailer_dataset")

Optional improvement: use a relative path approach to avoid machine-specific directory issues.

3.2 Data Preparation Pipeline

Where this is required; The project applies a sample based on the Instacart data to make the computation accessible on a local computer. In the event that the sample files have been pre-prepared and stored properly in retailer_dataset/ you can go directly to the notebook of forecasting.

3.2.1 Sample Generation

If you need to regenerate the sample from the full dataset:

- Ensure the full raw files exist in the dataset directory:
  - orders.csv
  - orderproductsprior.csv
  - orderproducts_train.csv
- Run the sampling script from the project root:
  - python DatasetSamples.py

The script:

- Reads orders.csv

- Randomly samples 120,000 orders using a fixed seed for reproducibility
- Writes orderssample.csv
- Filters prior/train order-product tables using the sampled orderid
- Writes:
  - orderproductspriorsample.csv
  - orderproductstrainsample.csv

3.2.2 Verification Checks

- Confirm the three _sample.csv files exist in retailer_dataset/
- Check file sizes are substantially smaller than the full dataset.
- headers match the expected Instacart schema.

3.2.3 Performance Caution

- Sampling can be memory-intensive due to chunked reads.
- Default CHUNKSIZE is 500,000 rows.
- If memory errors occur:
  Reduce CHUNKSIZE
  Reduce N_ORDERS
- This step typically needs to be done only once.

3.3 Forecasting and Inventory Simulation Setup

Open sku_forecasting_notebook_with_visuals.ipynb in VS Code.
Confirm the notebook kernel is set to your **Python 3.14 virtual environment**.
Run the notebook cells in sequence to ensure state consistency.
The notebook is expected to:

- load the sample order and line-item data,
- merge and clean tables as required,
- construct a time index and aggregate demand into modelling-ready series,
- fit multiple candidate forecasting models per SKU (where implemented),
- select the best model based on forecast performance logic,
- generate forecasts for the test horizon, and
- calculate forecast uncertainty measures used by the inventory logic.

3.3.1 Forecasting Execution

- The notebook fits one or more models per SKU and selects the best model using error-based logic.
- Candidate approaches referenced include:
  ARIMA
  Random Forest or similar ML regression with lag features
- Forecasts are produced for the test horizon aligned with the simulation period.

3.3.2 Forecast output file

The notebook creates **sku_forecasts.csv**, which should include:

- SKU/product identifier,
- period/time index,
- actual units for the test window,
- best forecast values,
- selected model label, and
- error or uncertainty fields where applicable.

	A	B	C	D	E	F	G	H	I
1	product_id	Time_Idx	Period	Actual_Units	Best_Forecast	Best_Model	MAPE_ARIMA	MAPE_RF	Forecast_Error_SD
2	1	194	194	1	0.306170845	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
3	1	195	195	0	0.316882202	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
4	1	196	196	0	0.31554619	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
5	1	197	197	0	0.315712829	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
6	1	198	198	0	0.315692045	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
7	1	199	199	0	0.315694637	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
8	1	200	200	0	0.315694314	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
9	1	201	201	0	0.315694354	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
10	1	202	202	0	0.315694349	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
11	1	203	203	1	0.31569435	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
12	1	204	204	0	0.315694349	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782
13	1	205	205	0	0.315694349	ARIMA(1,0,1)	0.707535497	0.73496737	0.556710782

Fig 3: Forecast output file

3.3.3 Inventory Simulation Execution

- The notebook simulates two inventory regimes per SKU:
  Baseline policy
  Tool (forecast-driven) policy
- Assumptions referenced:
  Lead time and review period parameters are incorporated, with example values of LeadTime = 2 and ReviewPeriod = 1 in sample outputs.
- Tool policy uses forecasts to compute dynamic reorder logic, likely using expected lead-time demand plus safety stock aligned with a target CSL design.

3.3.4 Simulation output file

The notebook creates **policy_metrics.csv**, which should include:

- Baseline vs Tool labels,
- CSL, Fill Rate, and Average On-Hand Inventory,
- Lead time and review period fields (if included in the implementation),
- The target CSL design value used for the tool policy.
- Each SKU should appear twice (one row per policy)

	A	B	C	D	E	F	G	H	I	J
1	product_id	Policy	CSL	FillRate	AvgOnHand	LeadTime	ReviewPeriod	TargetCSLDesign		
2	1	Baseline	0.8625	0.817246835	0.996145833	2	1	0.95		
3	1	Tool	0.991666667	0.989204989	2.929243338	2	1	0.95		
4	10	Baseline	0.891666667	0.805257009	1.351979167	2	1	0.95		
5	10	Tool	1	1	3.324282369	2	1	0.95		
6	23	Baseline	0.9375	0.736111111	0.70234375	2	1	0.95		
7	23	Tool	0.979166667	0.926277126	1.224215603	2	1	0.95		
8	25	Baseline	0.891666667	0.824295775	0.87078125	2	1	0.95		
9	25	Tool	0.995833333	0.997431976	2.900980208	2	1	0.95		
10	32	Baseline	0.916666667	0.4625	0.386458333	2	1	0.95		
11	32	Tool	0.991666667	0.991171441	1.219573891	2	1	0.95		
12	34	Baseline	0.933333333	0.924404762	2.553541667	2	1	0.95		

Fig 4: Simulation output file

After these two output are created and demonstrate valid data, which are not empty, the Python pipeline is finished. In the following manual, the system can then be prepared to Power BI linking and dashboard refresh.

4 Power BI Dashboard Configuration

This chapter describes the process of adapting and using the Power BI Desktop report (Evaluating Integrated Forecast to Inventory Planning in Retail.pbix) which shows the results of the forecasting and inventory simulation. Opening the report on a different computer might make it suspect missing data sources since it keeps referring to the folder of the developer. Accept privacy warnings, dismiss any password requests as the sources of data are local CSV files.

4.1 Connecting to Output Tables

4.1.1 Opening the Power BI report

Launch Microsoft Power BI Desktop.

- Go to File → Open and select the .pbix from your project directory.
- Expect a yellow warning ribbon or pop-up regarding missing sources or refresh requirements because the file was previously saved in the developer's environment.
- Accept any normal privacy/security prompts.
- If an unexpected password prompt appears, cancel it and proceed to update sources manually.

4.1.2 Why source updating is required

- The .pbix is linked to CSV paths that may still reference the developer's local machine.
- You must update these paths to your own project folder so Power BI can locate the correct files.

4.1.3 Updating CSV paths in Power Query

Select **Transform Data** (or **Edit Queries**) to open **Power Query Editor**.

- In the Queries pane, locate queries for:
  - sku_forecasts
  - policy_metrics
- Select sku_forecasts:
  - Check the first step (Source).
  - If the path points to a developer directory, change it:
  - Use the gear icon beside Source or right-click Source → Edit Settings.
  - Browse and select your local sku_forecasts.csv.
  - Confirm the preview shows expected fields (e.g., product_id, Period, Actual_Units, Best_Forecast, etc.).
- Repeat the same process for:
  - policy_metrics.csv, confirming fields like product_id, Policy, CSL, FillRate.

4.1.4 Handling additional queries (if present)

If a **Products** (or similar metadata) query exists:

- Update its source path to your local products.csv if you have it.
- If you do not have it and product names are not essential:
- Consider removing the query or disabling load.
- Note: visuals that depend on product names may break if this table is removed.

4.1.5 Applying changes

- Click Close & Apply.
- Power BI will load data from your CSVs.
- If errors occur:
  - File not found: re-check paths and filenames.
  - Access denied: ensure CSVs are not open in Excel.
- After successful loading, confirm both tables appear in the Fields pane with their expected columns.

4.2 Report Pages, Slicers, and Refresh

The report is organized in a way that demonstrates accuracy of the forecasting and effects of the integrated tool on inventory performance usually in several pages (tabs) addressing various insights.

- Forecasting accuracy,
- Baseline vs tool inventory performance,
- The service–inventory trade-offs generated by the integrated pipeline.

4.2.1 Typical report pages

- Forecast vs Actual (SKU Detail)
  - Likely uses sku_forecasts.
  - Shows a line chart comparing forecasted demand to actual demand for a selected SKU.
  - May display model name and/or error indicators.
  - A product selector slicer may allow switching between SKUs.
- Service Level Improvement (Comparison)
  - Likely uses policy_metrics.
  - Compares Baseline vs Tool using:
  - bar charts/cards for fill rate, CSL, average inventory
    scatter plots showing baseline vs tool improvements
    tables listing SKU-level metrics.
- Trade-off Analysis / Summary
  - Provides higher-level comparisons (average service levels, inventory changes, stockout reductions).
  - May include scenario filters if multiple parameter runs exist.
- Data Tables / Logs (if included)
  - May provide raw table views for transparency and validation.
  - The exact pages may vary; users should confirm by inspecting tabs in the .pbix.
- Common slicers and interactions
  - Product selector: filters visuals to a chosen product_id (or name if metadata exists).
  - Policy selector: filters or toggles Baseline vs Tool views where applicable.
  - Scenario/Target CSL selector: only relevant if multiple scenarios are stored in the CSVs.
  - Users can click chart elements to cross-highlight related visuals.
- Refreshing behaviour
  - After rerunning the Python notebook and regenerating output CSVs, click Refresh in Power BI.
  - Refresh re-reads the local CSVs and updates visuals.
  - If files are moved or renamed, the Source paths must be updated again in Power Query.
- Diagnosing blank visuals
  - Check the Data view to confirm tables loaded and row counts are non-zero.
  - Verify the CSVs actually contain data.
  - Clear slicers that may be filtering out all records.
  - If a Products table is missing or mislinked, confirm model relationships are still valid.
- Saving configuration
  - After confirming correct paths and functioning visuals, save the .pbix.
  - This preserves your updated local data source settings for future use.

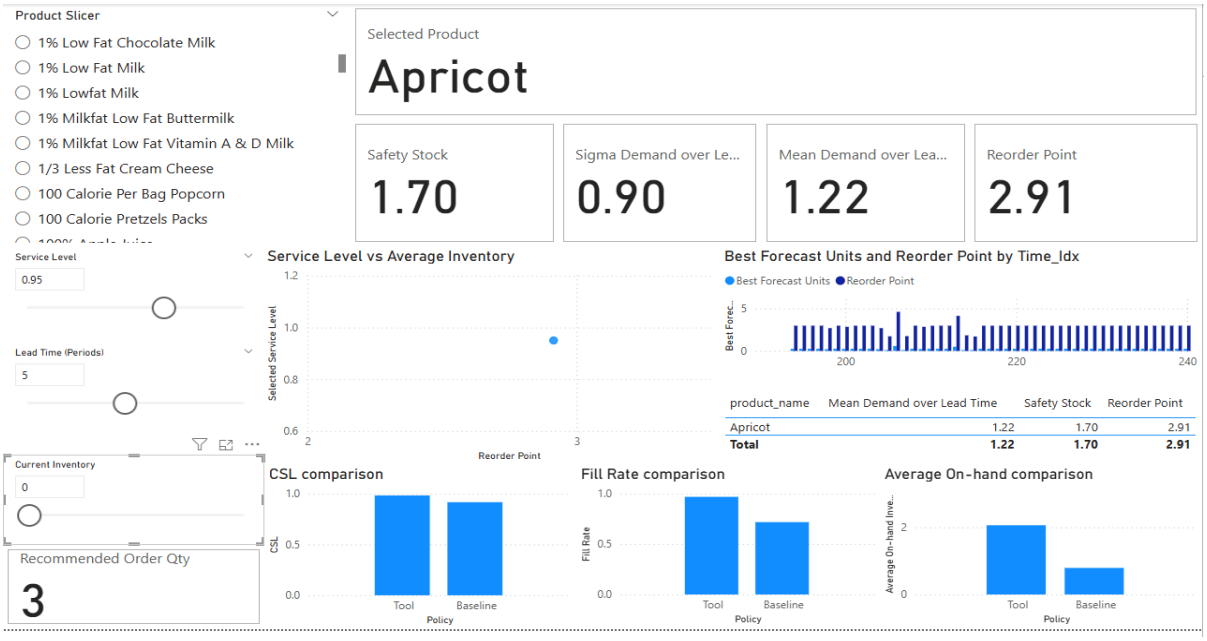

Fig 5: Power BI Dashboard

5 Logging, Output Files and Troubleshooting

This section provides a brief overview of the logging behaviour of the system, the most important output file generated by the Python pipeline, and some practical troubleshooting tips in the case of the most frequent configuration and execution problems. It allows users to confirm that they are successfully executed and fix issues without altering the logic of the integrated forecast-to-inventory tool.

5.1 Purpose of logging in this project

- The system primarily relies on console and notebook outputs to confirm correct execution.
- Logging supports quick validation of:
  - correct data sampling,
  - successful forecasting model runs, and
  - complete simulation output generation.

5.2 Logging and console output behaviour

- The sampling script prints progress messages showing:
  - files being read,
  - number of orders sampled,
  - chunk-based filtering progress, and
  - confirmation of saved sample files.
- The forecasting and simulation notebook prints:
  - model-fitting progress across SKUs,
  - intermediate warnings (e.g., ARIMA convergence),
  - and file-save confirmations at the end of execution.
- Warnings may appear during model training and are generally acceptable unless they stop execution.
- Because the notebook outputs logs **inline beneath cells**, users should:
  - run cells in order,
  - avoid skipping preprocessing steps, and
  - review the last executed cell if an error occurs.

There are no independent log files that are usually produced by the project, unless they are added to the code specifically; thus, the notebook output is the most important source of validation.

5.3 Key output files produced

5.3.1 sku_forecasts.csv

- Stores the forecast results per SKU across the forecast/test horizon.
- Includes forecast vs actual values, selected model information, and forecast error indicators used for evaluation and dashboard visuals.
- Used directly by Power BI to display forecast accuracy.
- Users should check:
  - that the file exists,
  - contains non-empty rows,
  - and does not have columns fully missing or null.

5.3.2 policy_metrics.csv

- Stores the inventory performance outputs under:
  - Baseline policy, and
  - Tool (integrated forecast-driven) policy.
- Contains the evaluation metrics central to the study:
  - **Cycle Service Level (CSL),**
  - **Fill Rate, and**
  - **Average On-Hand Inventory.**
- Each SKU should appear twice, once per policy.
- Users should verify:
  - CSL and Fill Rate are within 0–1,
  - AvgOnHand is non-negative.

5.4 Troubleshooting

5.4.1 Troubleshooting – Environment Issues

- Missing libraries
  - Errors like ModuleNotFoundError indicate packages were not installed in the active environment.
  - Fix by installing the missing package with pip, restarting the kernel, and rerunning the notebook.
- Kernel/interpreter mismatch
  - If execution stalls at “connecting to kernel” or runs without output:
  - Confirm that VS Code is using the correct virtual environment interpreter.
- Memory constraints

If the system slows significantly or raises MemoryError:

  - The dataset or the number of SKUs/models may be too large.
  - Reduce sampling size (N_ORDERS) or limit extremely low-selling SKUs.

5.4.2 Troubleshooting – Data and File Issues

- Incorrect file paths
  - If scripts cannot locate orders.csv, orders_sample.csv, or outputs:
  - Verify DATA_DIR, filenames, and working directory consistency.
- Delimiter/format changes
  - Ensure CSVs remain comma-delimited; regional settings or Excel resaving may change delimiters.
  - Regenerate files if corruption or formatting changes are suspected.
- Blank Power BI visuals

  Confirm:
  - CSVs exist,
  - contain data,

- Power BI queries point to the correct paths,
- slicers are not filtering out all rows

5.4.3 Troubleshooting – Power BI Specific

- Data source permissions
  - If Power BI prompts for privacy/permission:
  - Check Data source settings and set appropriate permissions for local files.
- Visual field mismatches
  - Red "X" errors may indicate that a column name changed in the CSV output.
  - Fix by mapping visuals to the updated field names.
- Performance degradation
  - If a larger dataset is loaded:
  - Consider reducing granularity and disabling auto date/time to prevent model bloat.

5.5 Reproducibility Best Practice

- To rerun experiments reliably:
  - Clear/rename old outputs,
  - Use Restart Kernel and Run All in the notebook,
  - Refresh Power BI after each new output generation.
- Comparing different scenarios may require separate output sets unless the system is extended to store multi-scenario data simultaneously.

To re-run a notebook with the same results, remove the old CSVs, re-run the notebook with a new kernel and re-open the dashboard. These are some of the practices that assist in maintaining and troubleshooting your forecasting and inventory system. These steps are to be followed so as not to cause problems.

References

Python 3.14 Documentation (n.d.). Python Software Foundation.
URL: https://docs.python.org/3.14/
Reference for the Python programming language environment used in this project.
Visual Studio Code Documentation (n.d.). Microsoft Docs.
URL: https://code.visualstudio.com/docs
Information on setting up and using Visual Studio Code, the integrated development environment used for running the code.
Microsoft Power BI Desktop (n.d.). Microsoft Docs.
URL: https://learn.microsoft.com/en-us/power-bi/
Official documentation for Power BI Desktop, the tool used to create and view the project's dashboards.
Instacart Online Grocery Shopping Dataset 2017 (2017). Instacart.
URL: https://www.instacart.com
Public retail transaction dataset used as the basis for the project's sample data. Originally released by Instacart in 2017, accessed via the official Instacart link. Contains orders and product data for an online retailer.
Kaggle – Instacart Market Basket Analysis Data (2017). Kaggle.
URL: https://www.kaggle.com/datasets/psparks/instacart-market-basket-analysis/data
Alternative access to the Instacart 2017 data through Kaggle. Includes the same orders and products files. Useful for obtaining the full dataset to generate new samples.