National College of Ireland

Configuration Manual

Research Practicum Part 2
Msc Data Analytics

Kshiteej Chourasia
Student ID: X23271019

School of Computing
National College of Ireland

Supervisor: Harshani Nagahamulla

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Kshiteej Chourasia
Student ID:	X23271019
Programme:	Msc Data Analytics
Year:	2024-2025
Module:	Research Practicum Part 2
Supervisor:	Harshani Nagahamulla
Submission Due Date:	15/09/2025
Project Title:	Configuration Manual
Word Count:	1390
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:	Kshiteej Chourasia
Date:	15th September 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

Kshiteej Chourasia
X23271019

1 Requirements

1.1 Hardware Requirements

- **Processor (CPU):** AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx (2.30GHz)
- **RAM:** 16 GB
- **Storage:** 512 GB SSD
- **Graphics (GPU):** NVIDIA GeForce GTX

1.2 Software Environment

- **Operating System:** Windows 11 (64-bit)
- **Python version:** 3.12.8
- **Major packages/libraries used:**
  - `pandas` (pandas development team, 2024)
  - `scikit-learn` (Pedregosa, 2011)
  - `keras` (et al., 2015)
  - `statsmodels` (Seabold and Perktold, 2010)
  - `numpy` (Harris, 2020)
  - `seaborn` (Waskom, 2023)
  - `MinMaxScaler` (Pedregosa, 2011)
  - `SimpleImputer` (Pedregosa, 2011)
- **Notebook environment:** Visual Studio Code (Corporation, 2024)
- **Database:** MongoDB version 8.0 (MongoDB, 2024)
- **Cloud/Deployment platform:** Not used (local only)

2 Accounts, Cloud & Resource Setup

A local environment was used to create and implement the project without cloud services utilized as well. The development, testing, and execution stages of it were all carried out locally.

3 Data Source Setup

3.1 Data Acquisition

The data sources were downloaded from the Kaggle event of grocery sales forecasting (kag, 2017). The total size of the data set was about 480 MB in the CSV format.

- `train.csv`: Training data (date, store_nbr, item_nbr, unique id, unit_sales, negative values for returns, onpromotion feature)
- `test.csv`: Test data (date, store_nbr, item_nbr, onpromotion)
- `stores.csv`: Store metadata (city, state, type, cluster)
- `items.csv`: Item metadata (family, class, perishable flag)
- `transactions.csv`: Count of sales transactions for each date, store
- `oil.csv`: Daily oil price, for macroeconomic features

```python
def load_data():
    # Load core datasets
    train = pd.read_csv(r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\train.csv", parse_dates=['date'], dtype={'store_nbr': 'category', 'item_nbr': 'category'})
    items = pd.read_csv(r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\items.csv", dtype={'family': 'category', 'class': 'category'})
    stores = pd.read_csv(r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\stores.csv", dtype={'city': 'category', 'state': 'category', 'type': 'category'})
    oil = pd.read_csv(r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\oil.csv", parse_dates=['date'])
    holidays = pd.read_csv(r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\holidays_events.csv", parse_dates=['date'])
    transactions = pd.read_csv(r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\transactions.csv", parse_dates=['date'])

    return train, items, stores, oil, holidays, transactions
```

MagicPython

Figure 1: All CSV files and Data Loading

- **Holiday data:** Downloaded via the Calendarific API (Calendarific, 2024): `https://calendarific.com/api/v2/holidays`

```
curl 'https://calendarific.com/api/v2/holidays?&api_key=baa9dc110aa712sd3a9fa2a3dwb6c01d4c875950dc32vs&country=US&year=2019'
```

Below is a shortened API response that executes successfully


```
{
    "meta": {
        "code": 200
    },
    "response": {
        "holidays": [
            {
                "name": "Name of holiday goes here",
                "description": "Description of holiday goes here",
                "date": {
                    "iso": "2018-12-31",
                    "datetime": {
                        "year": 2018,
                        "month": 12,
                        "day": 31
                    }
                },
                "type": [
                    "Type of Observance goes here"
                ]
            }
        ]
    }
}
```


Figure 2: Calendarific API

- **Weather data:** Downloaded via the Open-Meteo API (Open-Meteo, 2024): `https://open-meteo.com/`

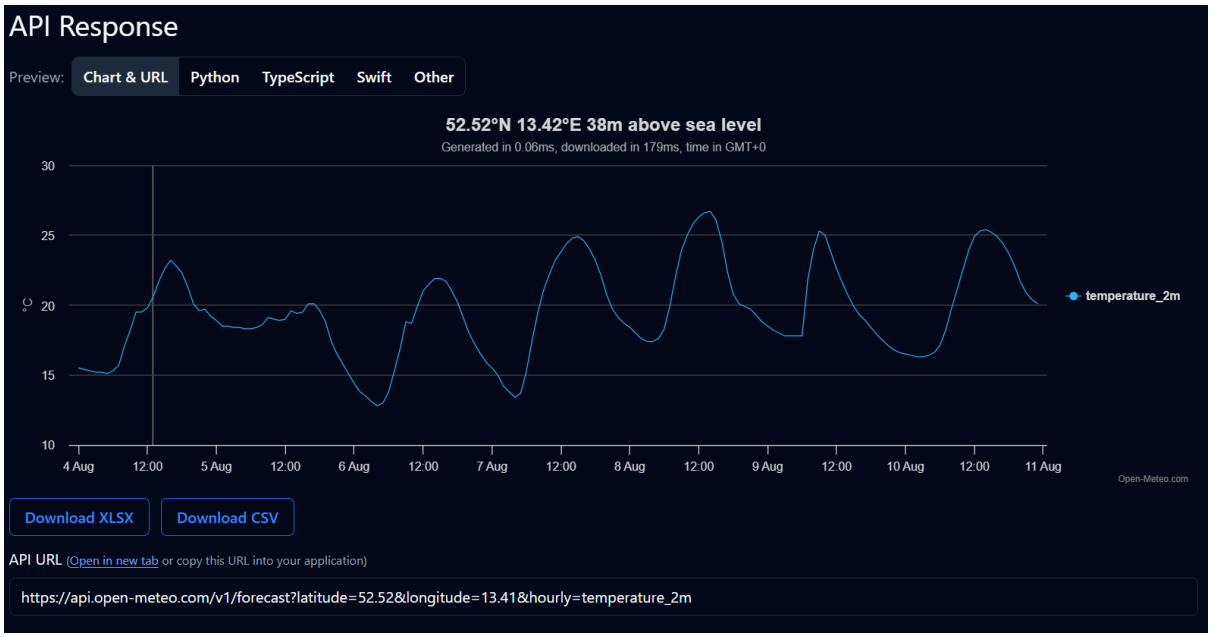

Figure 3: Open Meteo API

3.2 Data Storage Locations

- All CSV files (`train.csv`, `test.csv`, etc.) were stored in local folders.
- External API data (holiday/events and weather) was stored in MongoDB for efficient merging and querying with sales records (MongoDB, 2024).

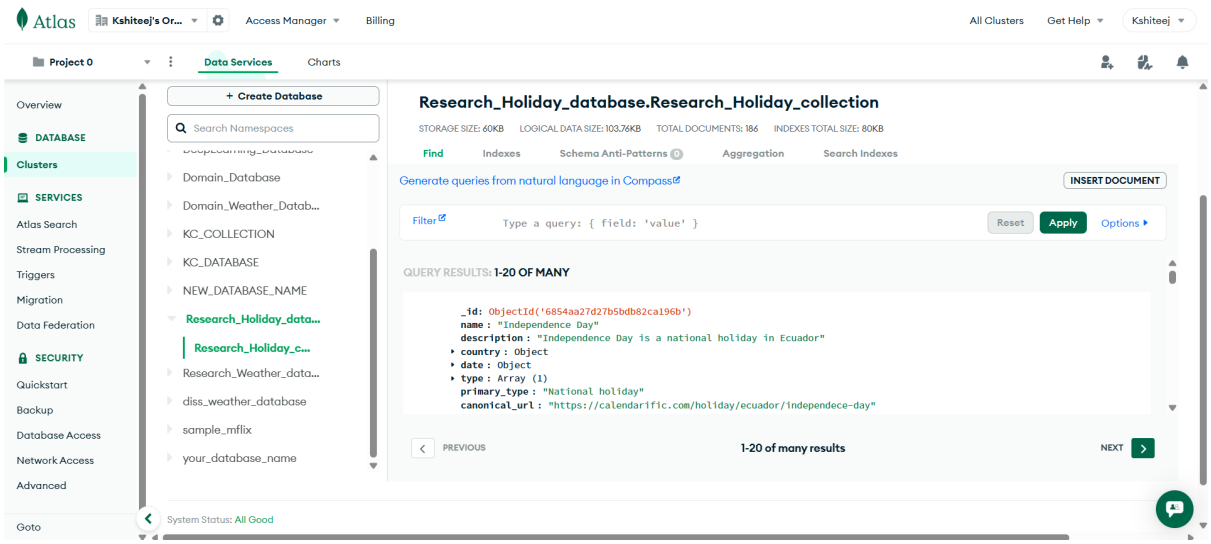

Figure 4: MongoDB Database: Holiday API

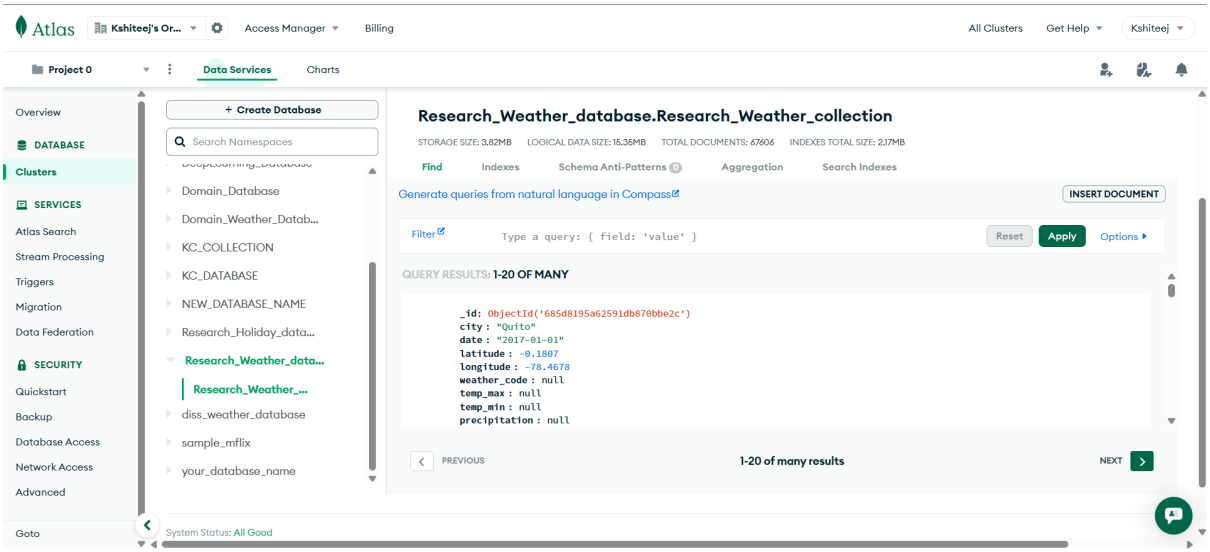

Figure 5: MongoDB Database: Weather API

4 Data Uploading and Preprocessing

4.1 Manual Data Upload Steps and Automation

The raw retail datasets which included sales transactions, holidays information and weather information were in the first place manually and automatically uploaded into local data stores and databases. Manual uploads meant that big data files of CSV and JSON were loaded into specific folders with a raw data named format. Where results on weather data in particular were concerned, APIs and MongoDB were connected with the aim towards automating frequent extraction and ingestion procedures.

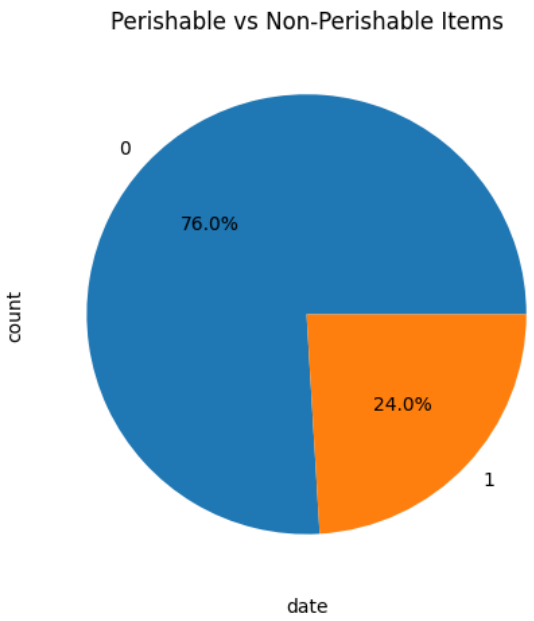

Figure 6: Perishable Vs. Non Perishable Items

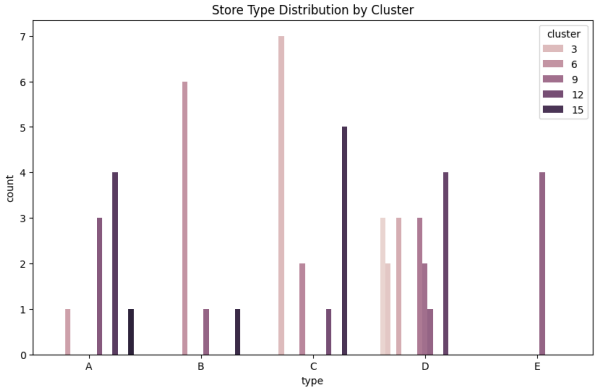

Figure 7: Store type Distribution by cluster

Big chunks of files were read and written by automated scripts in order to optimize both memory consumption and performance. These scripts made it possible to have data ingested effectively despite having to work with limited resources.

4.2 Example Scripts and Command Lines

Preprocessing and uploading of data was performed automatically using Python scripts. `pandas` (pandas development team, 2024) and `pymongo`. Examples of chunk based loading command lines are shown below:

```
import os

def merge_holiday_weather_chunked(input_path, output_path, holidays, weather, chunk_size=10000):
    if os.path.exists(output_path):
        os.remove(output_path)
    first = True
    for chunk in pd.read_csv(input_path, parse_dates=['date'], chunksize=chunk_size):
        chunk = chunk.merge(holidays, on='date', how='left')
        chunk = chunk.merge(weather, on=['date', 'city'], how='left')
        chunk['is_holiday'] = chunk['is_holiday'].fillna(0).astype('int8')
        chunk['holiday_type'] = chunk['holiday_type'].fillna('regular_day')
        chunk['temp_max'] = chunk['temp_max'].fillna(chunk['temp_max'].mean())
        chunk['temp_min'] = chunk['temp_min'].fillna(chunk['temp_min'].mean())
        chunk['precipitation'] = chunk['precipitation'].fillna(chunk['precipitation'].mean())
        chunk.to_csv(output_path, mode='a', index=False, header=first)
        first = False
        del chunk

# Merge for train
merge_holiday_weather_chunked(
    r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\train_top5_time_city.csv",
    r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\train_top5_final.csv",
    holidays_df,
    weather_df
)

# Merge for test
merge_holiday_weather_chunked(
    r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\test_top5_time_city.csv",
    r"C:\Users\91912\Desktop\NCI\Research Practicum\Practicum 2\test_top5_final.csv",
    holidays_df,
    weather_df
)
```

Figure 8: Chunk Based Loading and Merging of Datasets

The API data extraction and the MongoDB queries were also scripted in order to directly fetch the holiday and weather data into processing pipelines to ensure current and consistent external features.

4.3 Data Cleaning, Missing Data, and Merges

Data cleaning was mostly carried out in specialized preprocessing scripts, mainly removing the bulk of the HTML in the pages:

- Missing values in weather and holiday data were imputed with the help of mean or median values and deletion of rows was avoided, and data was not lost.
- Filtered sales data that contain only top 5 cities by sales volume in Ecuador making it simpler and narrowed the training of models.

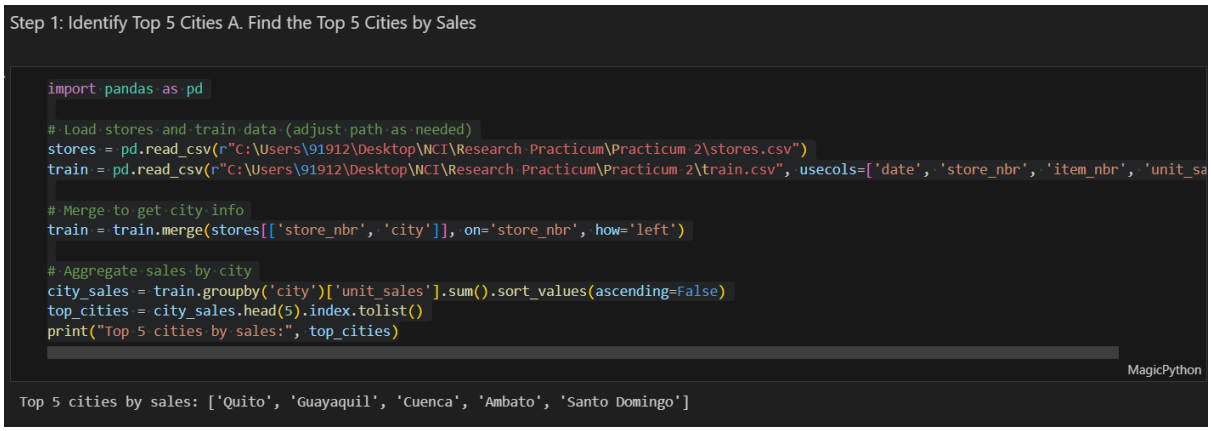

Figure 9: Top 5 City Dataset

- Made categorical variables as type data and Fixed list-like data conflicts as problematic data type.
- Holiday data based on keys such as a composite key of city and date (API and MongoDB) was combined with sales data using left joins which does not delete any sales details.

5 Feature Engineering Details

5.1 Derived and New Features

Using feature engineering, several new variables were also created to augment predictive abilities of the forecasting models that could not be provided by fitting the raw dataset variables alone. These included:

- **Temporal Features:** These are characteristics derived out of date columns e.g. day of week month, week and year to achieve seasonal and weekly sales trends.
- **Holiday Flags:** Binary flags and categorical flags of the type and date of public holidays which are moved in to place, as an integration of externally sourced holiday API and MongoDB.

- **Weather Features:** Including the maximum temperature variable, the minimum temperature variable, and the precipitation levels variable taken per city per date as environmental variables.

- **Lag Features:**Lag variables, or variables based on rolling windows like 14-day-rolling averages of the sales to put the variables in the context of time and identify trend.

- **Rolling Statistics:** More rolling statistics (e.g., rolling median, rolling standard deviation) probably calculated to summarize recent sales pattern within windows of predetermined time structure.

5.2 Scripts and Functions Used

- `Time Features:` Included with the `add_time_features(df)` command that pulled out `day_of_week`, `month`, and `year` out of `date` column of the training and test data sets.

- Chunk-Based processing: Your feature engineering process proposed to filter the top 5 cities, merge city information, and create new calendar features were all run in chunks so that the CSV files fitting in memory could be loaded.

- City, Holiday and Weather Feature Integration: City data was joined by using the function `merge_city_chunked`. The data was read into a MongoDB and using `get_holiday_data()` and `get_weather_data()`, features were fetched and then merged using `merge_holiday_weather_chunked` in bite-size chunks.

- Normalization and cleaning: Any columns that were categorical and thus be encoded (`city`, and `holiday_type` in the example) were pre-processed and included no list values, converting lists to strings where necessary.

- Missing Value dealing: Numeric columns (e.g. temperature) and average of the chunk. Categorical columns: empty strings like `regular_day` or `Unknown`.

- Encoding and Scaling: Operated on training set using scikit-learn `OneHotEncoder` and `MinMaxScaler`, fitting them to map some categorical features and scale numerical features, so the encoders and scaler can be used in the future (`ohe_encoder.pkl` and `scaler.pkl`).

- Sequence Creation: Neural models Final neural models were trained on windowed input sequences using a function such as `create_sequences(X y seq_len=14)`.

- Pipeline Management: These steps were choreographed with modular scripts which respectively saved intermediate result files (e.g., `ast_final.csv$`, `.npy` files), which was a robust and reproducible feature engineering.

These modular components ensured reusability and clarity, facilitating experimentation and tuning in downstream modeling steps.

6 Model Development

6.1 Model Development Pipeline Overview

Model development was in the form of a structured pipeline which included data preparation, model design, training, hyperparameter optimization and evaluation. The major steps were:

1. **Data Preparation:** Temporal Models Construction of rolling 14-day sequences of input data.
2. **Model Definition:** Hybrid CNN-GRU and LSTM Model run with configurable numbers of layers and parameters.
3. **Training:** Model training and early stopping to avoid overfitting and checking of the model to save the best weights.
4. **Hyperparameter Tuning:** keras-tuner to search systematically hyperparameter space in order to find best model configurations.
5. **Evaluation and Saving:** Measurement of performance of models with various measures and preservation of trained artifacts of models.

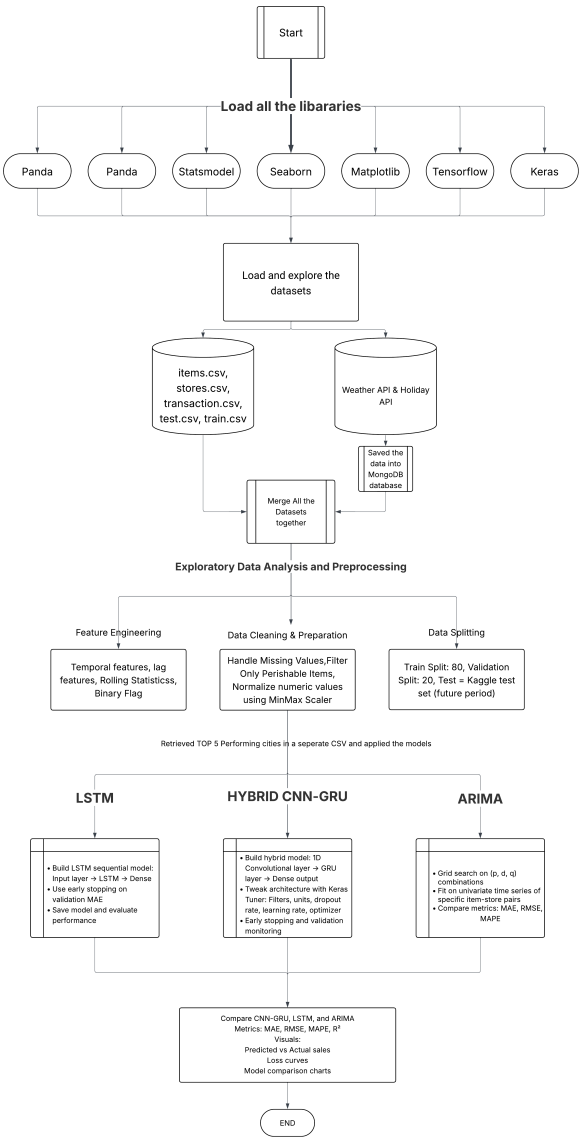

Figure 10: Flowchart of Model Development Pipeline

6.2 Neural Network Architectures and Hyperparameters

The model development phase focused on building and tuning neural network architectures suitable for time series forecasting with retail sales data:

- **Hybrid CNN-GRU** : Coupling convolutional layers (captures local temporal pattern) and GRU layers (captures long term dependency).
- **LSTM**: standard sequential benchmark LSTM.
- **Baseline ARIMA model**: stat standard time serie model for each store item per

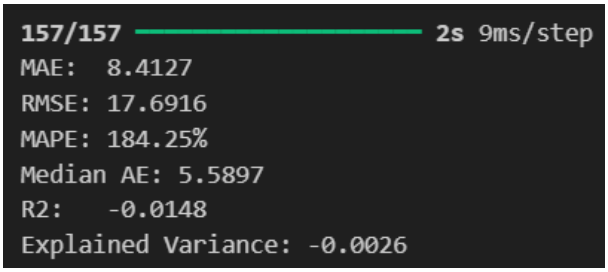

Figure 11: Results via different metrics

7 Model Evaluation

7.1 Evaluation Process

The assessment of the models was done through manual and automated methods since both methods are used to evaluate thoroughly the ability of the forecast:

- **Automated Evaluation:** Automated Assessment: Batch assessments that use scripts ran through validation and assessment data following learning of the model. Accuracy in the model was measured as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Median Absolute Error, R2 score, and Explained Variance programmatically (Pedregosa, 2011).
- **Manual Review:** The model outputs were visually analyzed by checking the graphs of actual sales verses model predicted sales, loss and validation loss curves and comparing charts of the model performance to determine any patterns of trends or overfitting/underfitting.
- The top-performing cities and the key product items were then re-evaluated so as to produce sufficiently granular insights and the applicability of the models in varying contexts.
- **Predictions:** Forecasted outputs of sales were stored as CSV files in a specified predictions/ sub directory and categorized by the type of model and the city/item name as a traceability measure.

8 Visualization

8.1 Visualization Tools and Methods

A combination of industry-standard Python visualization libraries and external dashboarding was used:

- `matplotlib` and `seaborn` (Waskom, 2023): For exploratory data analysis, loss curves, and actual vs. predicted plots in scripts and notebooks.

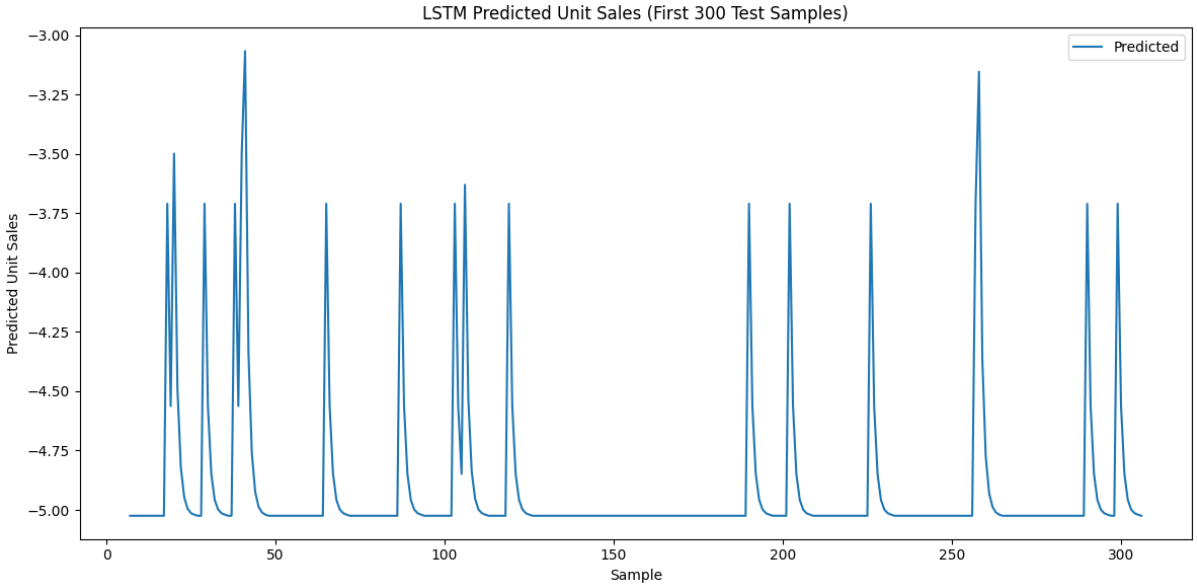

Figure 12: MatplotLib andSeaborn Plot 1

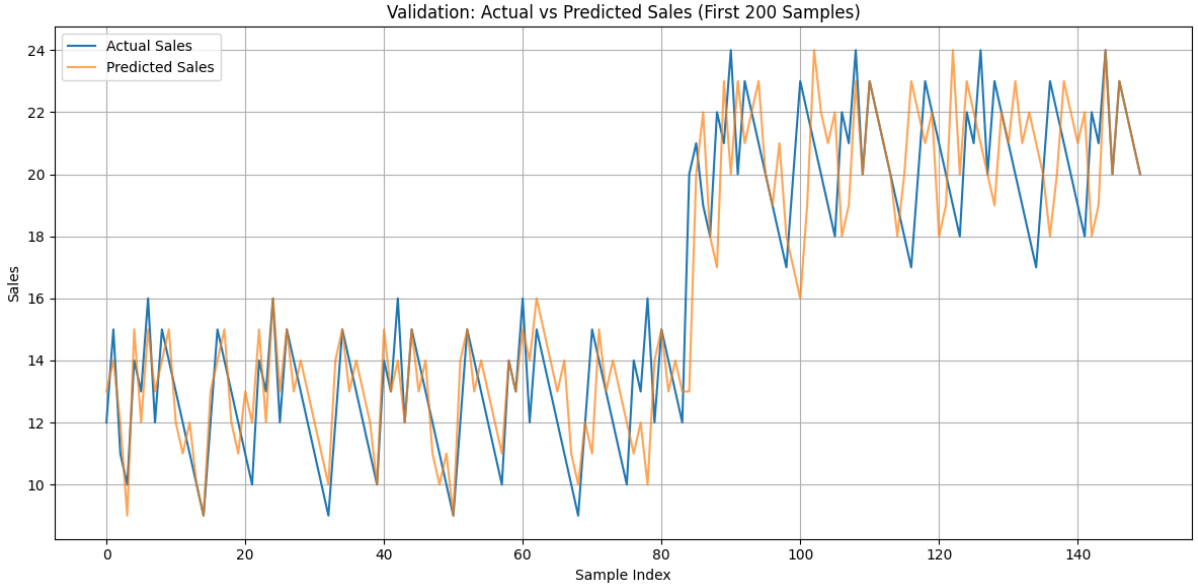

Figure 13: MatplotLib andSeaborn Plot 2

- `TensorBoard`: To monitor neural network training progress and performance metrics.

References

(2017). Corporación favorita grocery sales forecasting - kaggle. `https://www.kaggle.com/competitions/favorita-grocery-sales-forecasting`.

Calendarific (2024). Calendarific api documentation. Available at `https://calendarific.com/api-documentation`.

Corporation, M. (2024). Visual studio code. Available at https://code.visualstudio.com/.

et al., F. C. (2015). Keras. Available at https://keras.io/.

Harris, C. R. e. a. (2020). Array programming with numpy.

MongoDB, I. (2024). Mongodb database. Available at https://www.mongodb.com/.

Open-Meteo (2024). Open-meteo api documentation. Available at https://open-meteo.com/.

pandas development team, T. (2024). pandas - python data analysis library. Available at https://pandas.pydata.org/.

Pedregosa, F. e. a. (2011). Scikit-learn: Machine learning in python. Available at https://scikit-learn.org/.

Seabold, S. and Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. Proceedings of the 9th Python in Science Conference, 2010.

Waskom, M. (2023). Seaborn: statistical data visualization. Available at https://seaborn.pydata.org/.