

The Influence of Weather on Fashion Retail: Developing a Sustainable and Interpretable Forecasting Model

MSc Research Project
Artificial Intelligence for Business

César Antonio Ledesma González
Student ID: 23425016

School of Computing
National College of Ireland

Supervisor: Victor del Rosal

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: César Antonio Ledesma González

 23425016
Student ID:
Programme: MSc AI for Business **Year:** 2024
 Practicum 2: MSc Research Project
Module:
 Victor del Rosal
Lecturer:
Submission Due Date: 11/08/2025
Project Title: "The Influence of Weather on Fashion Retail: Developing a Sustainable and Interpretable Forecasting Model" // Configuration Manual

 750 10 pages
Word Count: **Page Count:**

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: César Antonio Ledesma González

 11/08/2025
Date:

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

César Antonio Ledesma González
Student ID: 23425016

1 Introduction

This Configuration Manual details the environment setup as well as the execution steps for the forecasting model developed during the MSc project “The Influence of Weather on Fashion Retail: Developing a Sustainable and Interpretable Forecasting Model.” It contains all details regarding the environment setup, and steps needed to reproduce experiments and results.

2 Hardware Requirements

The system setup for this research is described in Figure 1 along with the software specs used.

System Software Overview:	Hardware Overview:
System Version: macOS 15.5 (24F74)	Model Name: MacBook Pro
Kernel Version: Darwin 24.5.0	Model Identifier: Mac15,3
Boot Volume: Macintosh HD	Model Number: MTL73B/A
Boot Mode: Normal	Chip: Apple M3
Computer Name: MacBook Pro de César	Total Number of Cores: 8 (4 performance and 4 efficiency)
Username: César Ledesma (cesarledesma)	Memory: 8 GB
Secure Virtual Memory: Enabled	System Firmware Version: 11881.121.1
System Integrity Protection: Enabled	OS Loader Version: 11881.121.1
	Serial Number (system): K521Y46K39
	Hardware UUID: B4B5D60A-3F3B-5F64-ABF9-0E126EE1731E
	Provisioning UDID: 00008122-0001581C2180001C
	Activation Lock Status: Disabled

Fig 1. System Configuration

To effectively implement the forecasting model, the following specifications are recommended:

Minimum:

- CPU: Intel® Core™ i5/ Apple M1
- RAM: 8 GB
- Disk: 500 MB Available

Recommended:

- RAM: 16 GB (for faster SHAP/Prophet processing)

3 Environment Setup

Operation System: macOS Sequoia 15.5

Development tools:

- Visual Studio Code, Version: 1.102.2 (Universal)
- Python Virtual Environment (venv)
- Terminal (macOS default for iTerm2)

Programming Language:

- Python 3.12.11

4 Required Libraries and Dependencies

Prior to conduction any data analysis, must ensure that required libraries and packages that facilitate data cleaning, processing, modelling, and evaluation are installed. The critical libraries that have been references in thesis are:

- **Package Manager:** pip
- **Installation File:** requirement.txt

Requirements.txt

```
txt
#Core libraries
pandas==2.2.2
numpy==1.26.4
matplotlib==3.8.4
seaborn==0.13.2
scikit-learn==1.5.0

#Machine learning models
xgboost==2.0.3
lightgbm==4.3.0
statsmodels==0.14.2

#Time series and forecasting
prophet==1.1.5
pmdarima==2.0.4

#Model interpretation
shap==0.45.0

#Clustering and dimensionality reduction
scipy==1.13.0
umap-learn==0.5.6

#CmdStan backend for Prophet (required if using
Prophet with Stan)
cmdstanpy==1.2.1
```

Install with:

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

5 System Configuration

The development of this research project was carried using Visual Studio Code
Virtual Environment Setup:

```
bash

Python3 -m venv .venv
source .venv/bin/activate #macOS/Linux
```

Directory Structure:

```
/Modelling
├── prepared_sales_weather_dataset.csv
├── shap_feature_analysis.py
├── Prophet_forecast_comparison.py
├── clustering_and_forecasting_sub_category.py
└── outputs/ (plots & logs)
```

Environment Variables: None required explicitly.

This setup can also be adapted to Windows with Anaconda or Google Colab if desired.

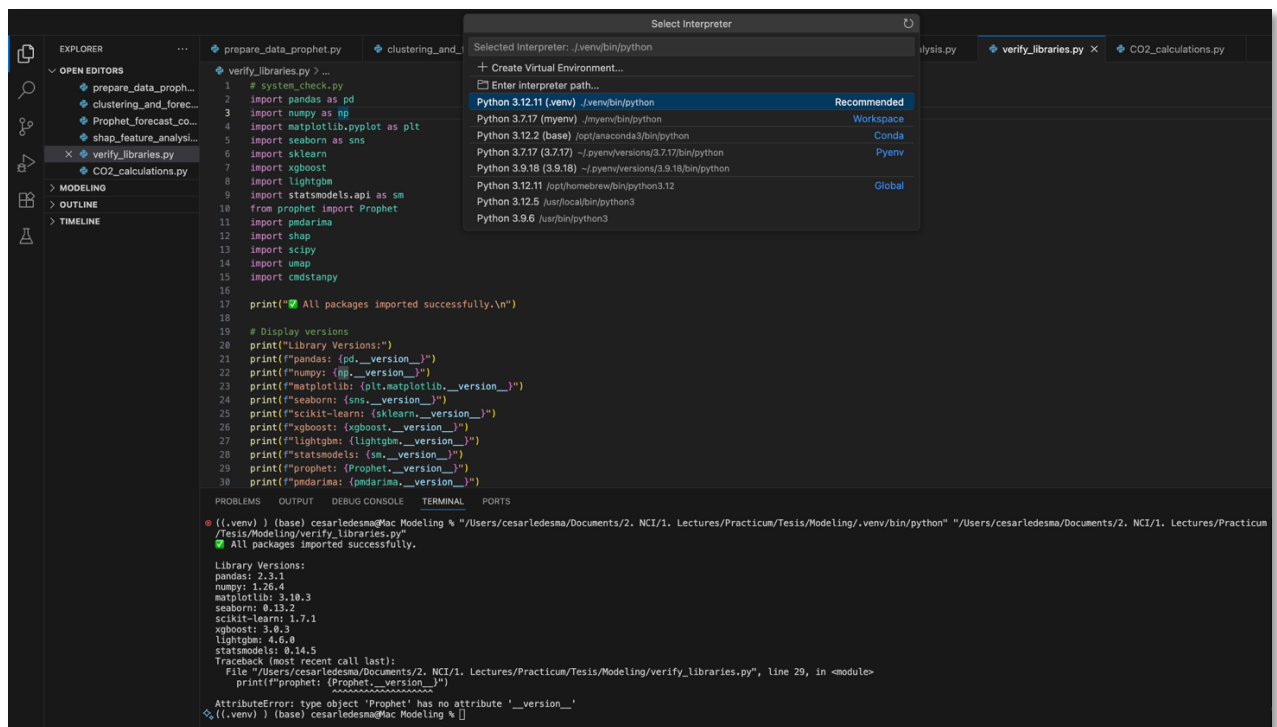
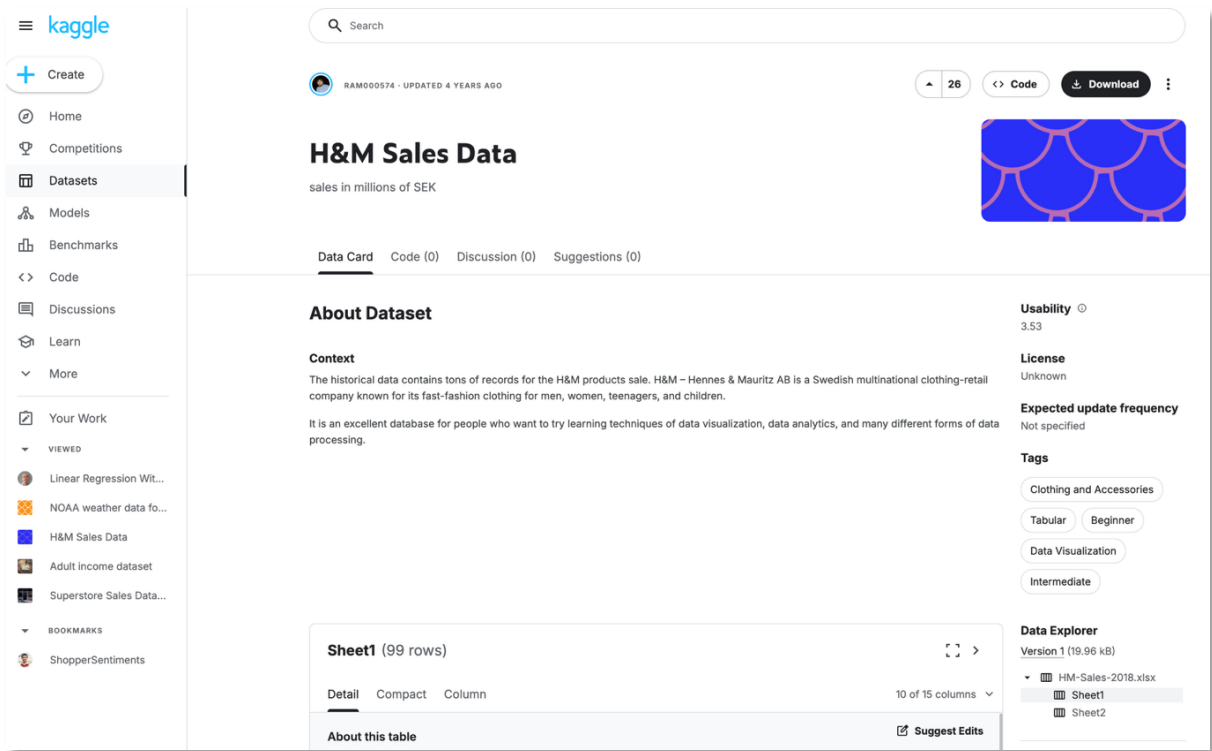


Fig 2. Environment Setup Visual Studio Code®

6 Data Collection

For this project, a public dataset titled “H&M Sales Data” was used, which contains historical sales records from the retail chain. This dataset is suitable for time series forecasting and demand analysis in the fashion retail domain. It can be accessed from Kaggle: <https://www.kaggle.com/datasets/tulasiram574/hm-sales-data>



The image shows a screenshot of the Kaggle website interface for the 'H&M Sales Data' dataset. The page layout includes a left-hand navigation menu with options like 'Create', 'Home', 'Competitions', 'Datasets', 'Models', 'Benchmarks', 'Code', 'Discussions', 'Learn', 'More', 'Your Work', and 'ShopperSentiments'. The main content area features a search bar, the dataset title 'H&M Sales Data' with a subtitle 'sales in millions of SEK', and a blue decorative graphic. Below the title, there are tabs for 'Data Card', 'Code (0)', 'Discussion (0)', and 'Suggestions (0)'. The 'About Dataset' section provides context, stating that the data contains records for H&M products sale and is suitable for learning techniques of data visualization and analytics. On the right side, there are sections for 'Usability' (3.53), 'License' (Unknown), 'Expected update frequency' (Not specified), and 'Tags' (Clothing and Accessories, Tabular, Beginner, Data Visualization, Intermediate). At the bottom, a 'Data Explorer' section shows a preview of 'Sheet1' with 99 rows and 10 of 15 columns visible.

Fig 3. Dataset

7 Data preprocessing

The initial data preparation involved addressing incomplete records by imputing missing values and excluding features such as store identifiers. Temporal attributes were included into a standard data time format before joining the dataset with external weather data. To further refine the dataset and emphasize underlying patterns. The resultant dataset was then exported and saved as *prepared_sales_weather_dataset.csv*.

```

Weather.py > ...
1 from meteostat import Stations, Daily
2 from datetime import datetime
3 import pandas as pd
4
5 cities = [
6     ("Birmingham", "AL"), ("Phoenix", "AZ"), ("Los Angeles", "CA"),
7     ("Wilmington", "DE"), ("Miami", "FL"), ("Chicago", "IL"),
8     ("Indianapolis", "IN"), ("Louisville", "KY"), ("Detroit", "MI"),
9     ("Minneapolis", "MN"), ("Omaha", "NE"), ("New York", "NY"),
10    ("Charlotte", "NC"), ("Portland", "OR"), ("Philadelphia", "PA"),
11    ("Charleston", "SC"), ("Nashville", "TN"), ("Houston", "TX"),
12    ("Salt Lake City", "UT"), ("Richmond", "VA"), ("Seattle", "WA"),
13    ("Milwaukee", "WI")
14 ]
15
16 start = datetime(2018, 1, 1)
17 end = datetime(2018, 12, 31)
18
19 city_weather_data = {}
20
21 for city, state in cities:
22     try:
23         stations = Stations().region('US', state)
24         station_list = stations.fetch(10)
25
26         if not station_list.empty:
27             station_id = station_list.index[0]
28             data = Daily(station_id, start, end).fetch()
29             data['City'] = city
30             city_weather_data[city] = data
31             print(f"✓ Fetched for {city}")
32         else:
33             print(f"⚠ No station found for {city}")
34
35     except Exception as e:
36         print(f"✖ Error for {city}: {str(e)}")
37
38 if city_weather_data:
39     combined_data = pd.concat(city_weather_data.values())
40     combined_data.to_csv("weather_data_2018_selected_us_cities.csv")
41     print(f"📄 CSV saved: weather_data_2018_selected_us_cities.csv")
42 else:
43     print(f"⚠ No data collected.")
44
45 import pandas as pd
46
47 # Load the merged dataset
48 df = pd.read_csv("synthetic_sales_weather_with_real_weather.csv")
49
50 # List of weather-related columns
51 weather_cols = ['tavg', 'tmin', 'tmax', 'prcp', 'snow', 'wdir', 'wspd', 'wpgt', 'pres', 'tsun']
52
53 # Fill missing values: list by region mean, then overall mean
54 for col in weather_cols:
55     if col in df.columns:
56         # Fill with region-wise mean

```

```

fetch_weather_from_sales.py > ...
1 from meteostat import Daily, Stations
2 from datetime import datetime
3 import pandas as pd
4 from geopy.geocoders import Nominatin
5 from time import sleep
6
7 # Fallback coordinates for cities with known geocoding issues
8 fallback_coors = {
9     "Chicago": (41.8781, -87.6298),
10    "Los Angeles": (34.0522, -118.2437),
11    "Seattle": (47.6062, -122.3321),
12    "San Francisco": (37.7749, -122.4194),
13    "Minneapolis": (44.9778, -93.2650),
14    "Houston": (29.7684, -95.3698),
15    "Philadelphia": (39.9526, -75.1652),
16    "Madison": (43.0731, -89.4012),
17    "Fort Worth": (32.7555, -97.3308),
18    "New York City": (40.7128, -74.0068),
19    "Portland": (45.5152, -122.6784),
20    "Memphis": (35.1495, -90.0498),
21    "Rochester": (43.1566, -77.6088),
22    "Dover": (39.1582, -75.5244)
23 }
24
25
26 # Initialize geolocator
27 geolocator = Nominatin(user_agent="weather-data-collector")
28
29 # Load your sales data
30 sales_df = pd.read_excel("HW-Sales-2018.xlsx")
31
32 # Unique city names
33 unique_cities = sales_df['City'].dropna().str.strip().str.title().unique()
34
35 # Date range
36 start = datetime(2018, 1, 1)
37 end = datetime(2018, 12, 31)
38
39 city_weather_data = {}
40
41 for city in unique_cities:
42     try:
43         # First try geocoding
44         location = None
45         try:
46             location = geolocator.geocode(f"{city}, USA", timeout=10)
47         except Exception as geocode_error:
48             print(f"⚠ Geocoding timeout for {city}: {geocode_error}")
49
50         if not location and city in fallback_coors:
51             lat, lon = fallback_coors[city]
52             print(f"📍 Using fallback coordinates for {city}")
53         elif location:
54             lat, lon = location.latitude, location.longitude
55         else:
56             print(f"✖ Skipping {city}: no coordinates")

```

Fig. 4 Data preprocessing pipeline overview

8 Instructions to run the Model

4.1 SHAP Feature Importance

The SHAP analysis was performed to interpret the feature influence on the sales prediction model. The model was fitted to the pre-processed dataset and SHAP values were generated to assess the effect of variables including “*Temperature*”, “*Humidity*” and “*Day of the Week*” upon the sales forecast. The output includes the summary plot, which orders the features according to their marginal contribution, this clarifying and enhancing interpretability of the model and facilitating informed choices regarding feature retention and actionable insights for business strategies.

- SHAP Feature Importance Analysis:

```

bash
python shap_feature_analysis.py

```

```

shap_feature_analysis.py > ...
1 import pandas as pd
2 import shap
3 import xgboost as xgb
4 import matplotlib.pyplot as plt
5 from sklearn.model_selection import train_test_split
6
7 # Load dataset
8 df = pd.read_csv("prepared_sales_weather_dataset.csv")
9
10 # Prepare data
11 features = ['tavg_normalized', 'prcp_normalized', 'wspd_normalized', 'Month']
12 df = pd.get_dummies(df, columns=["Sub-Category"], drop_first=True)
13 X = df[features + [col for col in df.columns if "Sub-Category_" in col]]
14 y = df["Sales"]
15
16 # Train model
17 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
18 model = xgb.XGBRegressor(n_estimators=100, random_state=42)
19 model.fit(X_train, y_train)
20
21 # SHAP analysis
22 explainer = shap.Explainer(model)
23 shap_values = explainer(X_train)
24
25 # Plot
26 shap.summary_plot(shap_values, X_train, show=False)
27 plt.tight_layout()
28 plt.savefig("shap_summary_plot.png")
29 plt.close()
30

```

Fig 5. SHAP feature analysis

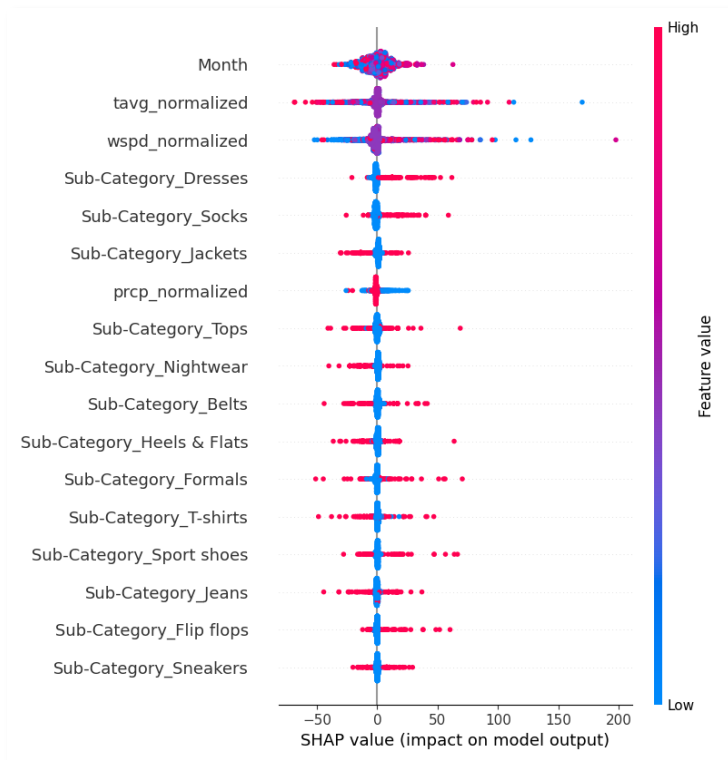


Fig 6. SHAP expected output plot

4.2 Forecasting and Clustering Prophet

This step focuses on builds a robust forecasting model using Prophet both at an aggregate and subcategory level. It includes applying clustering techniques to group similar subcategories, which allows more specific forecasts. Seasonal patterns are also examined to enhance temporal understanding and model granularity.

```
bash
```

```
python Prophet_forecast_comparison.py
```

Expected Output:

- **Forecast plots:** store in outputs/
- **SHAP summary plot:** shap_summary_plot.png
- Forecast RMSE logs printed to console

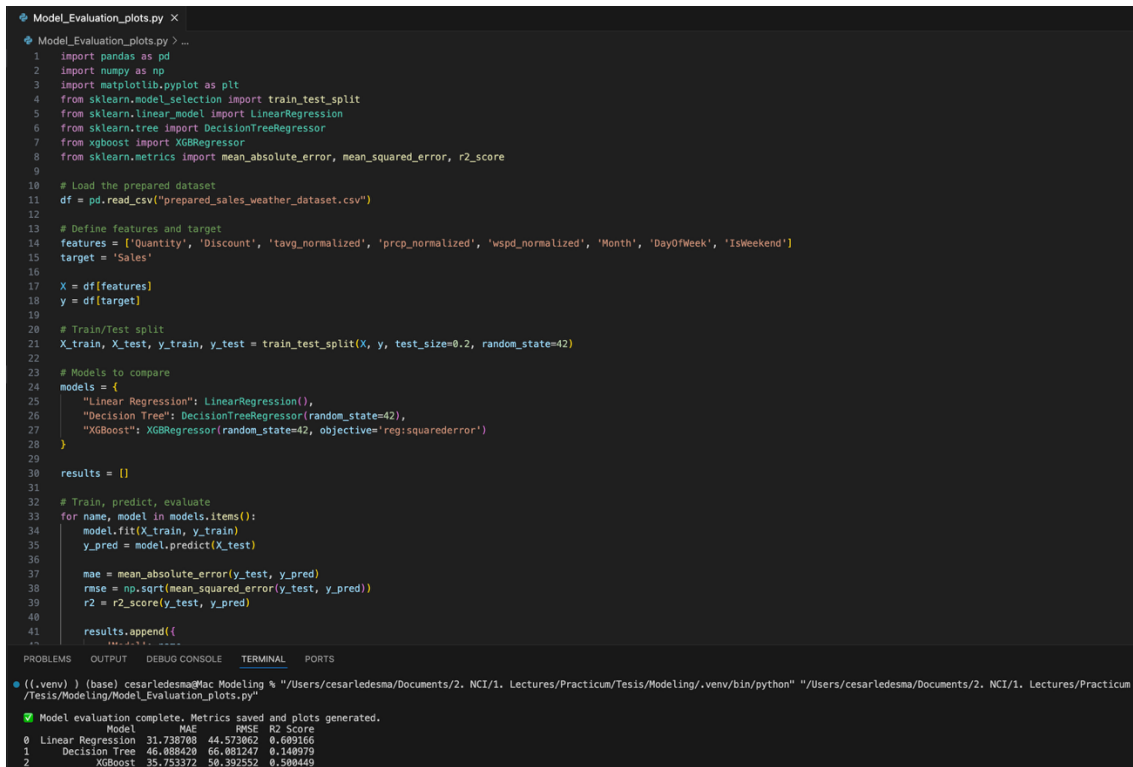


```
clustering_and_forecasting_sub_category.py x
clustering_and_forecasting_sub_category.py > ...
1 # clustering_and_forecasting_sub_category.py
2 # This script combines clustering of sub-categories based on monthly seasonality and forecasting using Prophet.
3
4 import pandas as pd
5 import numpy as np
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 from sklearn.preprocessing import StandardScaler
9 from sklearn.cluster import KMeans
10 from sklearn.decomposition import PCA
11 from prophet import Prophet
12
13 # Load your sales dataset
14 df = pd.read_csv('prepared_sales_weather_dataset.csv')
15
16 # Preprocessing
17 df['Order Date'] = pd.to_datetime(df['Order Date'])
18 df['Month'] = df['Order Date'].dt.month
19 df['DayOfWeek'] = df['Order Date'].dt.day_name()
20
21 # Compute monthly seasonality per sub-category
22 monthly = df.groupby(['Sub-Category', 'Month'])['Sales'].mean().unstack().fillna(0)
23
24 # Clustering
25 scaler = StandardScaler()
26 monthly_scaled = scaler.fit_transform(monthly)
27
28 kmeans = KMeans(n_clusters=3, random_state=42)
29 clusters = kmeans.fit_predict(monthly_scaled)
30
31 monthly['Cluster'] = clusters
32
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
Tops RMSE: 44.30
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
Tops BASE RMSE (no regressors): 44.19
Tops IMPROVEMENT: -0.10 RMSE reduction using regressors
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
Socks RMSE: 69.78
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
Socks BASE RMSE (no regressors): 70.41
Socks IMPROVEMENT: 0.63 RMSE reduction using regressors
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
T-shirts RMSE: 66.96
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
T-shirts BASE RMSE (no regressors): 67.44
T-shirts IMPROVEMENT: 0.48 RMSE reduction using regressors
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
Sport shoes RMSE: 79.42
09:27:30 - cmdstanpy - INFO - Chain [1] start processing
09:27:30 - cmdstanpy - INFO - Chain [1] done processing
Sport shoes BASE RMSE (no regressors): 79.26
Sport shoes IMPROVEMENT: -0.16 RMSE reduction using regressors
((.venv) ) (base) cesarledesma@Mac Modeling %
```

Fig 7. Forecasting and Clustering with Prophet

4.3 Model Evaluation

To assess the performance of the forecasting models generated in this work modevaluationplot.py was facilitated. Performance indicators such as RMSE and MAE, the script enable a comparison of forecasting accuracy among the Prophet model configurations and the clusters. The visualisations clarified the performance of predictive subcategories, affirming the contributions of feature importance, the design of clustering scheme, and incorporation of seasonality.



```
Model_Evaluation_plots.py x
Model_Evaluation_plots.py > ...
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LinearRegression
6 from sklearn.tree import DecisionTreeRegressor
7 from xgboost import XGBRegressor
8 from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
9
10 # Load the prepared dataset
11 df = pd.read_csv("prepared_sales_weather_dataset.csv")
12
13 # Define features and target
14 features = ['Quantity', 'Discount', 'tavg_normalized', 'prcp_normalized', 'wspd_normalized', 'Month', 'DayOfWeek', 'IsWeekend']
15 target = 'Sales'
16
17 X = df[features]
18 y = df[target]
19
20 # Train/Test split
21 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
22
23 # Models to compare
24 models = {
25     "Linear Regression": LinearRegression(),
26     "Decision Tree": DecisionTreeRegressor(random_state=42),
27     "XGBoost": XGBRegressor(random_state=42, objective='reg:squarederror')
28 }
29
30 results = []
31
32 # Train, predict, evaluate
33 for name, model in models.items():
34     model.fit(X_train, y_train)
35     y_pred = model.predict(X_test)
36
37     mae = mean_absolute_error(y_test, y_pred)
38     rmse = np.sqrt(mean_squared_error(y_test, y_pred))
39     r2 = r2_score(y_test, y_pred)
40
41     results.append({
42         'Model': name,
43         'MAE': mae,
44         'RMSE': rmse,
45         'R2 Score': r2
46     })
47
48 # Save results to a CSV file
49 results_df = pd.DataFrame(results)
50 results_df.to_csv("model_evaluation_results.csv", index=False)
51
52 # Generate plots
53 plt.figure(figsize=(10, 5))
54 results_df[['Model', 'MAE', 'RMSE', 'R2 Score']].plot(kind='bar')
55 plt.title('Model Performance Comparison')
56 plt.xlabel('Model')
57 plt.ylabel('Metric')
58 plt.savefig("model_evaluation_plots.png")
59
60 # Print results
61 print("Model evaluation complete. Metrics saved and plots generated.")
62 print(results_df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
((.venv) ) (base) cesarledesma@Mac Modeling % "/Users/cesarledesma/Documents/2. NCI/1. Lectures/Practicum/Tesis/Modeling/.venv/bin/python" "/Users/cesarledesma/Documents/2. NCI/1. Lectures/Practicum/Tesis/Modeling/Model_Evaluation_plots.py"
Model evaluation complete. Metrics saved and plots generated.
Model MAE RMSE R2 Score
0 Linear Regression 31.738708 44.573662 0.609166
1 Decision Tree 46.088428 66.081247 0.140979
2 XGBoost 35.753372 50.392552 0.500449
```

Fig 8. Model Evaluation script

4.4 Sustainability Impact Calculations

The final stage measures the environmental impact of improved forecasting. By considering estimations of potential CO₂ emissions, offering a quantification for sustainable computational operations.

```

CO2_calculations.py x
CO2_calculations.py > ...
1 # Example Inputs
2 runtime_hours = 12 # Total training + inference time
3 power_draw_watts = 150 # Estimated power consumption (can vary)
4 PUE = 1.57 # Power Usage Effectiveness of cloud provider
5 grid_emission_factor = 0.231 # kg CO2e per kWh (EU avg)
6
7 # Convert watts to kW
8 power_kw = power_draw_watts / 1000
9
10 # Calculate energy usage
11 energy_kWh = power_kw * runtime_hours * PUE
12
13 # CO2e estimate
14 emissions_kg = energy_kWh * grid_emission_factor
15
16 print(f"Estimated CO2 emissions: {emissions_kg:.2f} kg CO2e")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
((.venv) ) (base) cesarledesma@Mac Modeling % "/Users/cesarledesma/Documents/2. NCI/1. Lectures/Practicum/Tesis/Modeling/.venv/bin/python" "/User
s/cesarledesma/Documents/2. NCI/1. Lectures/Practicum/Tesis/Modeling/CO2_calculations.py"
Estimated CO2 emissions: 0.65 kg CO2e
((.venv) ) (base) cesarledesma@Mac Modeling %

```

Fig 9. CO2 Calculations script

9 Troubleshooting Tips

Issue	Solution
KeyError: `Date`	Ensure the CSV has a properly formatted Date column. Check with <code>df.columns</code>
TypeError: unexpected keyword argument `square`	Use <code>mean_squared_error(..., squared=False)</code> with compatible sklearn version
ModuleNotFoundError: No module name `Prophet`	Run <code>pip install prophet</code> . For M1/M2 chips, install dependencies like <code>cmdstanpy</code> first.
SHAP slow to compute	Use <code>X_train.sample(1000)</code> to reduce memory load
Plot doesn't display	Use <code>plt.savefig()</code> instead of <code>plt.show()</code> when running from scripts

References

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), pp.5–32.
<https://doi.org/10.1023/A:1010933404324>

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp.785–794. <https://doi.org/10.1145/2939672.2939785>

Lundberg, S.M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. [online] Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Taylor, S.J. and Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1), pp.37–45.
<https://doi.org/10.1080/00031305.2017.1380080>

Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp.2825–2830.
<https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf>

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In: van der Walt, S. and Millman, J. eds., *Proceedings of the 9th Python in Science Conference*. Austin, TX: SciPy, pp.51–56.

Prophet (n.d.). Facebook Prophet Documentation. [online] Available at: <https://facebook.github.io/prophet/> <https://peerj.com/preprints/3190/>

SHAP (n.d.). SHAP Documentation. [online] Available at: <https://shap.readthedocs.io/>

NCI Library (n.d.). NCI Referencing Guide. [online] Available at: <https://libguides.ncirl.ie/referencingandavoidingplagiarism>

E. Hernández, An Ensemble Model to Predict the Classification of Goods Using Text Description, M.Sc. thesis, School of Computing, National College of Ireland, Dublin, Ireland, 2023. [Online]. Available: <https://norma.ncirl.ie/id/eprint/7526>