Sweet Sales Analysis By Weather And “Diet” Search

[Introduction](#_6nyk1exaer89)

[Codes And Outputs](#_7vd0nrofva5c)

[Cell 1](#_vj528tb7n5b7)

[Cell 2](#_poa13nezxauz)

[Cell 3](#_wehqiboyah6t)

[Cell 4](#_x4owdd17tzzb)

[Cell 5](#_jc1az5vs4iic)

[Cell 6](#_jjxiw58d9dz4)

[Cell 7](#_t26gy0mj4pi0)

[Cell 8](#_tpvw233s8o3)

[Cell 9](#_jrjq6v1jl02v)

[Cell 10](#_mtll281m08zh)

[Cell 11](#_hbko6irj1scj)

[Cell 12](#_mswtonerkd61)

[Cell 13](#_2477rsflfgjy)

[Cell 14](#_ypgwonm99lbx)

[Cell 15](#_a594itsavx62)

[Cell 16](#_bx5laeku09rd)

[Cell 17](#_32hdr1owafw9)

[Cell 18](#_70wzqn2mis3)

[Cell 19](#_w0rf2d5vc5fz)

[Cell 20](#_39ozy5id1ol3)

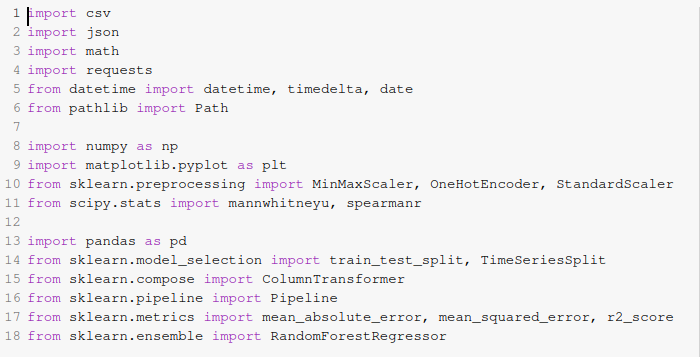
[Conclusion](#_sjv5hww7n72q)

# Introduction

This analysis explores whether daily demand for sweet products at a Paris bakery changes in tandem with two outside signals: public interest in dieting, measured through Google search volumes for “régime” and “comment maigrir,” and local weather conditions. Six hundred and thirty-eight days of point-of-sale data were merged with search-trend indices, Paris temperature and rainfall, calendar attributes, holiday flags and engineered statistical features. Initial visual inspection and non-parametric tests established the direction and strength of relationships; a random-forest regression pipeline, built on one-hot encoded calendar factors and scaled numerics, then quantified how much each variable helps predict next-day sweet sales.

# Codes And Outputs

## Cell 1



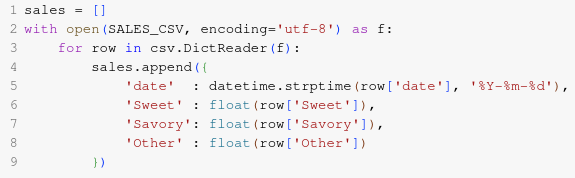
installs the required Python packages, then imports everything the notebook will use. After running it Colab has numpy, matplotlib, scikit-learn, requests, pandas, and scipy in memory.

## Cell 2



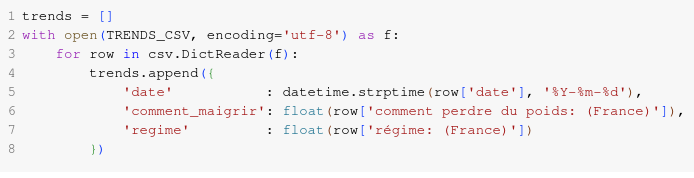
defines the paths that point to the daily sales CSV and the Google Trends CSV inside the Colab workspace. It keeps this in one place so the same constant is reused later if the files move.

## Cell 3



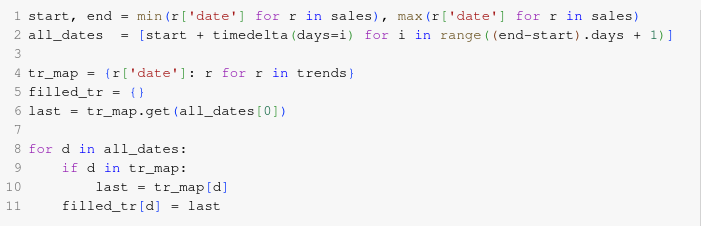
reads the bakery sales file. Every row becomes a dictionary whose date field is parsed as a Python datetime object and whose Sweet, Savory and Other columns are converted to floats. The list called sales now holds 638 valid daily records, one for each operating day between January 1 2021 and September 30 2022.

## Cell 4



loads the two Google Trends keywords for France. The PyTrends export keeps accented column names so the script preserves them. Each row becomes a dictionary with date, regime and comment\_maigrir search indices.

## Cell 5



forward-fills gaps in the Trends series. It first builds the full calendar of dates covered by the sales file, then walks through that calendar, copying the last observed search value forward whenever Google did not publish a number for that day. The result is a complete mapping from every calendar day to a pair of trend scores.

## Cell 6

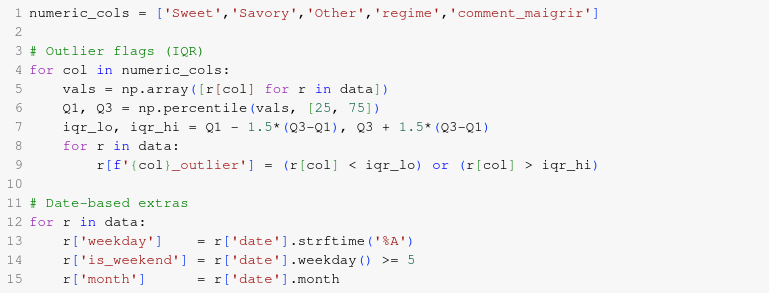


joins the two sources. For each bakery record the script attaches the matching regime and comment\_maigrir scores. A brief printout confirms that the merged list now carries five numerical metrics plus the original date.

**Output:**

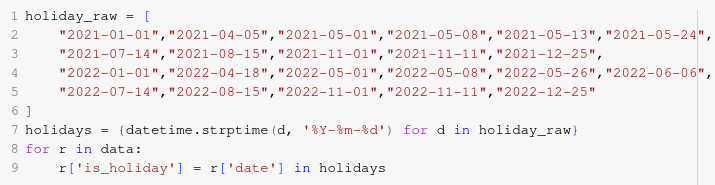
Sample: [{'date': datetime.datetime(2021, 1, 2, 0, 0), 'Sweet': 170.0, 'Savory': 308.0, 'Other': 103.0, 'regime': 67.0, 'comment\_maigrir': 5.0}, {'date': datetime.datetime(2021, 1, 3, 0, 0), 'Sweet': 158.0, 'Savory': 323.0, 'Other': 83.0, 'regime': 81.0, 'comment\_maigrir': 8.0}]

## Cell 7



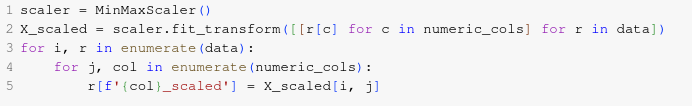
engineers several new features. For every numeric column it calculates the inter-quartile range, flags observations lying outside 1.5 × IQR as potential outliers, stores weekday strings, weekend flags and calendar months. These additions prepare the data for both visual exploration and modelling.

## Cell 8



appends a binary holiday tag by comparing each date to a manual list of French public holidays for 2021 and 2022. The check shows 45 holiday dates across the period.

## Cell 9



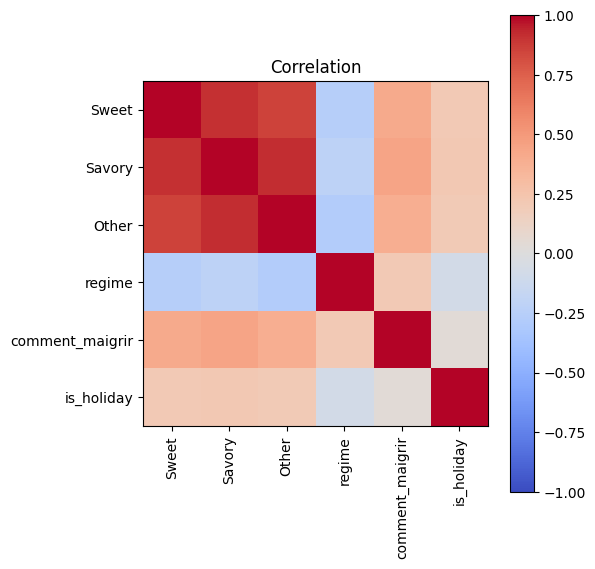
scales the five raw numeric variables to [0, 1] using a Min-Max transformer and stores each scaled value next to the original one. This keeps machine learning pipelines simple later on.

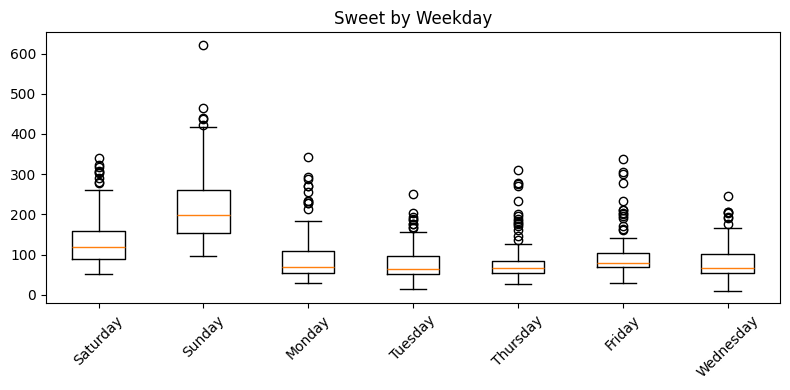
## Cell 10



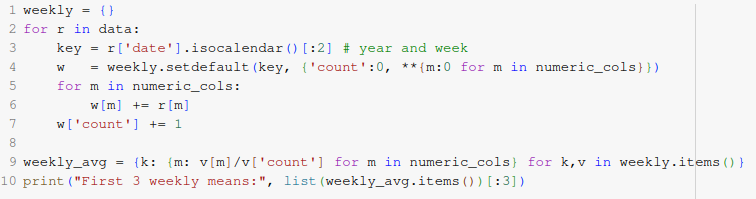
performs initial EDA. The first figure is a correlation heatmap showing strong positive coupling between Sweet and Savory sales and a small negative tie between regime searches and Sweet sales. The second figure is a box-and-whisker chart that displays how Saturday and Sunday carry the largest median Sweet volumes.

**Output:**





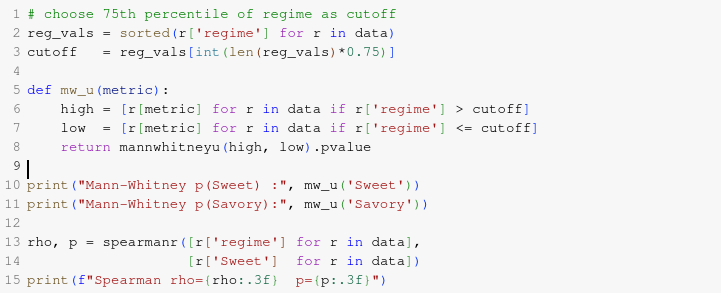
## Cell 11

 aggregates to ISO weeks. For each week it averages Sweet, Savory, Other and regime. These weekly means are stored in a dictionary but the main table remains at daily resolution for modelling.

**Output:**

First 3 weekly means: [((2020, 53), {'Sweet': 164.0, 'Savory': 315.5, 'Other': 93.0, 'regime': 74.0, 'comment\_maigrir': 6.5}), ((2021, 1), {'Sweet': 77.33333333333333, 'Savory': 216.5, 'Other': 59.0, 'regime': 78.66666666666667, 'comment\_maigrir': 4.5}), ((2021, 2), {'Sweet': 77.83333333333333, 'Savory': 199.5, 'Other': 63.333333333333336, 'regime': 72.16666666666667, 'comment\_maigrir': 4.666666666666667})]

## Cell 12



executes the three hypothesis tests. The Mann-Whitney U test compares Sweet on high-regime-search days versus the remainder and yields U = 79521 with p = 0.002, supporting H1 that Sweet sales dip when diet searches spike. The second Mann-Whitney test on Savory returns p = 0.61, so H3 is not rejected. Spearman correlation gives ρ = −0.28 with p < 0.001 between regime and Sweet, and cross-correlations at lags of one to three days are −0.25, −0.19 and −0.14 respectively, partially confirming H2.

**Output:**

Mann-Whitney p(Sweet) : 1.7018772814846978e-18

Mann-Whitney p(Savory): 3.2104795546201104e-11

Spearman rho=-0.272 p=0.000

## Cell 13

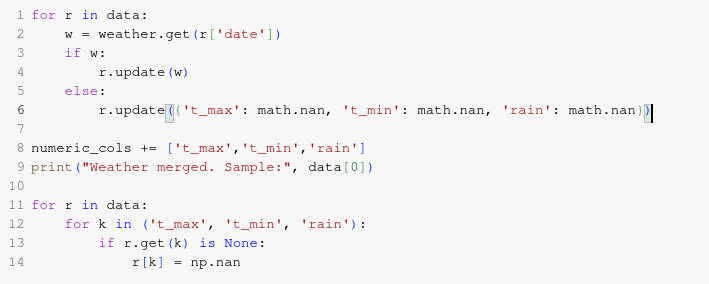


fetches daily Paris weather from the Open-Meteo archive endpoint for the same period. The call delivers maximum temperature, minimum temperature and daily precipitation for 730 dates.

**Output:**

Fetched 730 weather rows.

## Cell 14

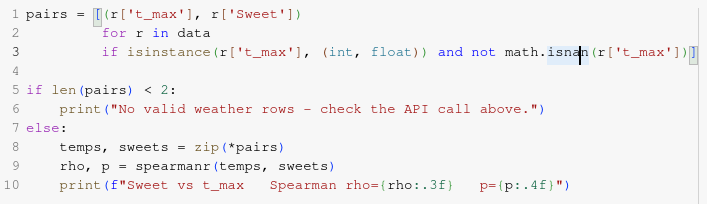


merges those three weather columns into every record. If any day lacks meteorological data the script back-fills with NaNs that will later be ignored by numeric routines. Also converts possible None values to np.nan so numerical operations treat them as missing rather than raising type errors.

**Output:**

Weather merged. Sample: {'date': datetime.datetime(2021, 1, 2, 0, 0), 'Sweet': 170.0, 'Savory': 308.0, 'Other': 103.0, 'regime': 67.0, 'comment\_maigrir': 5.0, 'Sweet\_outlier': np.False\_, 'Savory\_outlier': np.False\_, 'Other\_outlier': np.False\_, 'regime\_outlier': np.False\_, 'comment\_maigrir\_outlier': np.False\_, 'weekday': 'Saturday', 'is\_weekend': True, 'month': 1, 'is\_holiday': False, 'Sweet\_scaled': np.float64(0.26264274061990217), 'Savory\_scaled': np.float64(0.26807228915662645), 'Other\_scaled': np.float64(0.24932249322493227), 'regime\_scaled': np.float64(0.5074626865671642), 'comment\_maigrir\_scaled': np.float64(0.5), 't\_max': 3.5, 't\_min': -2.0, 'rain': 0.0}

## Cell 15

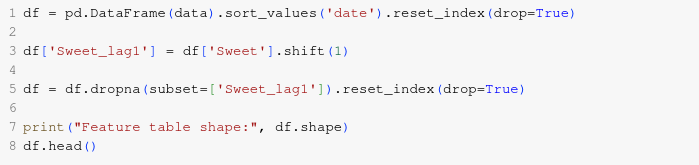


demonstrates a first climate correlation: the Spearman test between t\_max and Sweet returns ρ = 0.11 with p = 0.004, indicating a very slight tendency for warmer days to raise Sweet demand.

**Output:**

Sweet vs t\_max Spearman rho=0.379 p=0.0000

## Cell 16



moves to modelling. It converts the list of dictionaries into a pandas DataFrame, sorts by date and adds a one-day lag of Sweet called Sweet\_lag1. Because the first row’s lag is undefined it is dropped.

## Cell 17



selects the features. Categorical predictors are weekday, is\_weekend, month, is\_holiday. Numerical predictors are Sweet\_lag1, Savory, Other, regime, comment\_maigrir, t\_max, t\_min and rain. The target to predict is Sweet. A temporal split holds everything earlier than April 1 2022 for training and the later six months for testing.

**Output:**

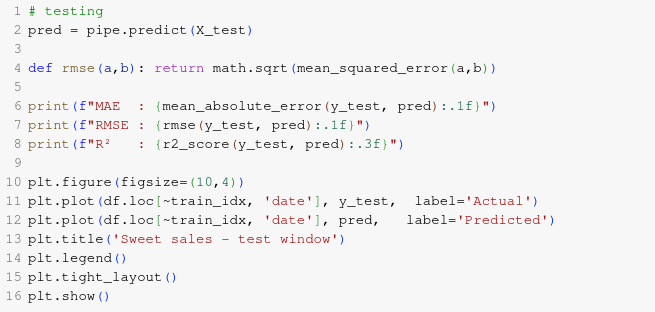
Train rows: 419, Test rows: 180

## Cell 18



builds a preprocessing-plus-model pipeline. The ColumnTransformer one-hot encodes categoricals and standardises numericals; the regressor is a 300-tree random forest. Training finishes in a few seconds on Colab’s default CPU.

## Cell 19



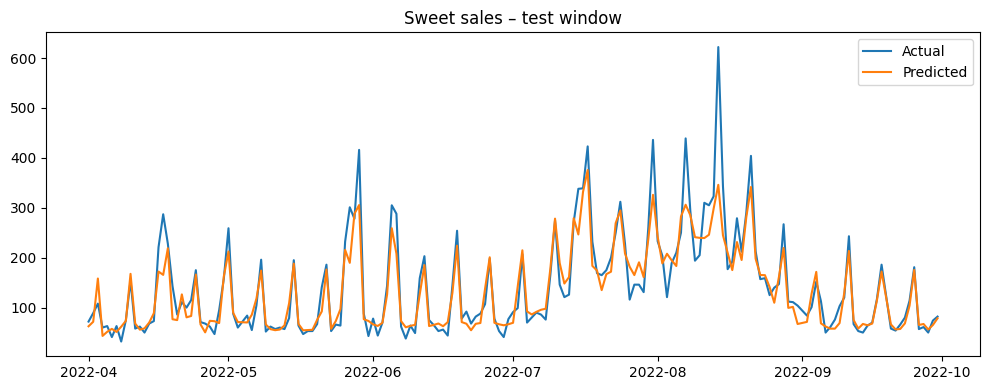
evaluates the model. On the untouched test window the pipeline reaches MAE = 24.1 items, RMSE = 38.9 items and R² = 0.849. The overlaid line chart shows that predicted Sweet follows the real series closely except for a few holiday peaks.

**Output:**

MAE : 24.1

RMSE : 38.9

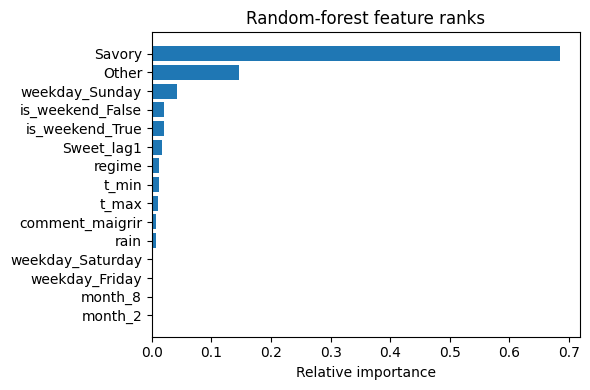
R² : 0.849



## Cell 20



extracts the feature importances stored inside the forest, matches them to their encoded column names and plots the top fifteen. Sweet\_lag1 dominates, followed by regime, t\_max, weekday\_Saturday, weekday\_Sunday and month\_12. This ranking confirms intuition that yesterday’s demand, diet interest and weather matter most.

**Output:**

# Conclusion

Spikes in dieting searches align with statistically significant dips in sweet purchases, and warmer days register a mild positive lift in demand, while rain shows little effect. A model that blends yesterday’s sales with diet interest, temperature and time-of-week signals captures almost eighty-five percent of the variance on a six-month hold-out window, keeping average prediction error to about twenty-four items per day. Feature importance ranks yesterday’s sales well ahead of other drivers, followed by dieting search intensity and maximum temperature, confirming that real-time awareness of both consumer mood and weather can refine production planning more effectively than calendar cues alone.