IRON HACK

Data Analytics

Analysis of regulated ingredients in cosmetic products

Zina BABACI

February, 2026

Table of content

1. Introduction

1.1 Context

The European cosmetics sector, valued at approximately €100 billion annually, operates within one of the most stringent regulatory environments globally. Regulation (EC) No. 1223/2009 establishes the core legal framework governing cosmetic safety across the European Union. Within this regulation, Annex II lists roughly 1,600 prohibited substances, while Annex III specifies more than 375 restricted ingredients with strict concentration limits and usage conditions. Additional attention is given to CMR substances—those classified as carcinogenic, mutagenic, or toxic for reproduction—which are subject to the highest level of regulatory control.

Despite the comprehensiveness of this regulatory framework, there is currently no automated, centralized system capable of verifying compliance across the full range of commercially available cosmetic products. Compliance verification typically relies on manual review of ingredient lists (INCI lists), which requires approximately two to three hours per product. This makes large-scale compliance monitoring across thousands of products operationally impractical and largely reactive rather than preventive.

1.2 Problem Statement

The core business challenge addressed in this project is the efficient identification of cosmetic products containing restricted or CMR ingredients across thousands of stock keeping units (SKUs). Existing workflows rely heavily on manual inspection, fragmented retailer databases, and inconsistent regulatory reference formats, preventing real-time or proactive compliance analysis.

1.3 Objectives

The primary objective was to design and implement an automated data pipeline capable of extracting, processing, and analyzing cosmetic product compositions against EU regulatory databases. Secondary objectives included:

- Aggregating data from multiple heterogeneous sources
- Designing a normalized relational database
- Documenting data aggregation and transformation processes
- Providing data access via a REST API for dashboard integration
- Conducting GDPR compliance analysis
- Developing a predictive machine learning model for product rating estimation

The success criteria targeted extraction of more than 10,000 product compositions, ingredient matching accuracy above 85%, API response times below 200 ms, and predictive model RMSE below 0.5.

1.4 Scope

The project focused on Sephora and Skincare product catalogs, EU CosIng regulatory databases, REST API sources, and relational/analytical storage via MySQL and Google BigQuery. Real-time manufacturer compliance tracking, international markets outside the EU, and clinical safety testing data were outside the defined scope.

2. Data Sources and Integration

2.1 Multi-Source Architecture

The project integrated five distinct data source categories:

1. Flat files (CSV and Excel regulatory and catalog data)
2. REST API product data
3. Web scraping for supplemental ingredient information
4. MySQL relational storage
5. Google BigQuery analytical warehouse

Source Type	Implementation	Volume	Purpose
Flat Files	CSV, Excel	9,966 products	Product catalogs, regulatory lists
REST API	HTTP requests	90 products	Real-time product data
Web Scraping	BeautifulSoup4	20 ingredients	Supplemental ingredient info
MySQL Database	SQLAlchemy ORM	8,732 rows	Relational storage
BigQuery	google-cloud-bigquery	10,451 rows	Analytical queries

The consolidated dataset contained 10,451 unique cosmetic products.

2.2 Source Characteristics

Source 1: Flat Files (CSV/Excel)

CosIng Annex III (Excel): - **Source**: European Commission official database - **File**: cosing_annex3.xls - **Content**: 375 restricted ingredients with INCI names, CAS numbers, restriction conditions - **Format**: Excel 97-2003 (.xls) - **Encoding**: Latin-1 (due to special characters) - **Key Fields**: ingredient_name, cas_number, restriction_text, is_cmr

Sephora Product Catalog (CSV): - **Source**: Sephora e-commerce website export - **File**: sephora_products.csv - **Content**: 8,494 cosmetic products with full INCI lists - **Format**: UTF-8 encoded CSV - **Key Fields**: product_id, product_name, brand_name, price_usd, rating, ingredients, product_type

Skincare Product Catalog (CSV): - **Source**: Skincare.com database - **File**: skincare_products.csv - **Content**: 1,472 products - **Format**: UTF-8 encoded CSV - **Key Fields**: Similar schema to Sephora

Challenges: - Mixed encoding (UTF-8 vs Latin-1) - Inconsistent delimiters (comma vs semicolon) - Malformed rows requiring error handling

Source 2: REST API

Open Beauty Facts API: - **Endpoint**: https://world.openbeautyfacts.org/api/v2/search - **Protocol**: HTTP GET with query parameters - **Authentication**: None (public API) - **Rate Limiting**: 100 requests/hour - **Pagination**: 20 results per page

Implementation:

```
import requests

response = requests.get(
    'https://world.openfoodfacts.org/api/v2/search',
    params={
        'categories_tags': 'en:cosmetics',
        'page': 1,
        'page_size': 20,
        'fields': 'code,product_name,ingredients_text'
    }
)
products = response.json()['products']
```

Challenge: Rate limiting required 1-second delays between requests.

Source 3: Web Scraping

Initial Target: CosIng EU official website - **Problem**: Anti-bot protection (CAPTCHA) blocked automated access

Pivot Strategy: Wikipedia ingredient pages - **URL**: https://en.wikipedia.org/wiki/List_of_cosmetic_ingredients - **Tool**: BeautifulSoup4 HTML parser - **Extraction**: Ingredient names and functions from HTML tables - **Volume**: 20 cosmetic ingredients - **Ethical Compliance**: Robots.txt respected, 1-second delay between requests

Source 4: MySQL Database

Connection Details: - **DBMS**: MySQL 8.0.35 - **Engine**: InnoDB (ACID compliance) - **Charset**: utf8mb4 (full Unicode support) - **Connection**: SQLAlchemy ORM with connection pooling

Extraction Query:

```
SELECT product_id, product_name, brand_name, price, rating, ingredients
FROM sephora_products
WHERE ingredients IS NOT NULL;
```

Volume: 7,380 Sephora products + 1,352 Skincare products

Source 5: Google BigQuery

Implementation Strategy: - **Partitioning**: PARTITION BY date_added (daily partitions) - **Clustering**: CLUSTER BY brand_name, product_type - **Benefits**: 10× faster time-range queries, 90% data scan reduction

SQL Implementation:

```
CREATE OR REPLACE TABLE cosmetics_data.sephora_products_partitioned
PARTITION BY date_added
CLUSTER BY brand_name, product_type
AS
SELECT
  *,
  CURRENT_DATE() AS date_added
FROM cosmetics_data.sephora_products;
```

Performance Comparison: - Full scan: 1.2s (both) - Last 30 days: 1.2s → 0.12s (10× faster) - Data scanned: 50 MB → 5 MB (90% reduction)

3. Data Collection and ETL Pipeline

3.1 Extraction

A structured ETL (Extract, Transform, Load) pipeline extracted data from each source using Python libraries including pandas, SQLAlchemy, requests, and BeautifulSoup. Data ingestion included error handling, logging, and retry mechanisms to ensure robustness.

```
# Flat files
df_cosing = pd.read_excel('cosing_annex3.xls', encoding='latin-1')
df_sephora = pd.read_csv('sephora_products.csv', encoding='utf-8')
df_skincare = pd.read_csv('skincare_products.csv', encoding='utf-8')

# REST API
response = requests.get('https://world.openbeautyfacts.org/api/v2/search',
                        params={'categories_tags': 'en:cosmetics'})
df_api = pd.DataFrame(response.json()['products'])

# Web scraping
soup = BeautifulSoup(requests.get('https://en.wikipedia.org/...').content, 'h
tml.parser')
df_wiki = extract_ingredients_from_table(soup)

# MySQL
engine = create_engine('mysql+pymysql://user:pass@localhost/cosmetics')
df_mysql = pd.read_sql('SELECT * FROM sephora_products', engine)

# BigQuery
from google.cloud import bigquery
client = bigquery.Client(project='cosmetics-project')
df_bq = client.query('SELECT * FROM cosmetics_data.sephora_products').to_data
frame()
```

3.2 Data Quality and Error Handling

Extraction logs indicated a 99.8% success rate with only 23 malformed rows failing ingestion. Timestamped logging enabled traceability and debugging throughout the pipeline.

```
import logging
from datetime import datetime

logging.basicConfig(
    filename=f'extraction_{datetime.now().strftime("%Y%m%d_%H%M%S")}.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

try:
    df_sephora = pd.read_csv('sephora_products.csv')
    logging.info(f"Loaded {len(df_sephora)} Sephora products")
except FileNotFoundError as e:
    logging.error(f"File not found: {e}")
except pd.errors.ParserError as e:
    logging.error(f"CSV parsing error: {e}")
```

3.3 Data Dictionary

Key structured fields included product identifiers, product names, brands, prices, ratings, ingredient lists, and computed regulatory indicators such as restricted ingredient counts and CMR flags.

Field	Type	Description	Example
product_id	VARCHAR(50)	Unique product identifier	“P473671”
product_name	VARCHAR(255)	Commercial product name	“Moisturizing Cream”
brand_name	VARCHAR(100)	Manufacturer brand	“CLINIQUE”
price_usd	DECIMAL(10,2)	Price in USD	45.00
Rating	DECIMAL(3,2)	User rating 0-5	4.25
Ingredients	TEXT	Comma-separated INCI list	“AQUA, GLYCERIN, …”
product_type	VARCHAR(50)	Category	“Skincare”
restricted_ingredient_count	INT	Number of Annex III ingredients	5
cmr_count	INT	Number of CMR substances	0
has_restricted_ingredient	BOOLEAN	Flag for restricted presence	TRUE
has_cmr	BOOLEAN	Flag for CMR presence	FALSE

4. Data cleaning and Exploratory Analysis

4.1 Cleaning Challenges

Major preprocessing challenges included nested ingredient lists stored as comma-separated strings, ingredient naming inconsistencies across data sources, and duplicate product entries. Ingredient normalization using uppercase conversion, punctuation removal, and pattern cleaning improved matching accuracy from 62% to 89%. Deduplication procedures retained the lowest-price variant of duplicated products.

4.2 Dataset Overview

The final dataset included:

- 10,451 products
- 487 brands
- 6 product categories
- Average price: $42.18
- Average rating: 4.20 stars

Correlation analysis revealed extremely weak relationships between objective product characteristics (price, ingredient counts, safety indicators) and consumer ratings, suggesting ratings are primarily driven by subjective factors.

4.3 Regulatory Aggregation Results

Sephora Catalog (7,380 products): - Products with restricted ingredients: **5,599 (75.87%)** - Products with CMR ingredients: **393 (5.33%)** - Average restricted ingredients per product: 9.3 - Unique restricted ingredients detected: 186

Skincare Catalog (1,352 products): - Products with restricted ingredients: **849 (62.8%)** - Products with CMR ingredients: **90 (6.66%)** - Average restricted ingredients per product: 7.8 - Unique restricted ingredients detected: 87

Comparative Insight: Sephora products (premium brands) contain 13.07 percentage points MORE restricted ingredients than Skincare products. This suggests premium brands use more complex formulations that trigger regulatory restrictions.

Category Breakdown (Sephora): - Fragrances: 84.2% contain restricted ingredients (highest) - Makeup: 75.6% - Skincare: 68.9% - Hair Care: 64.1%

Premium brands were more likely to contain restricted ingredients, likely due to more complex formulations containing regulated allergenic compounds.

5. Database Architecture and Selection

5.1 Requirements

Functional Requirements: - Store 10,000+ product records - Support complex joins (products ↔ ingredients ↔ restrictions) - Enable aggregation queries (GROUP BY, COUNT, AVG) - Provide ACID transactions for data integrity - Support both OLTP (API queries) and OLAP (analytics)

Non-Functional Requirements: - Query response time: <200ms for simple selects, <2s for aggregations - Scalability: Support growth to 100,000+ products - Concurrent access: Handle 100+ simultaneous API requests - Data integrity: Enforce referential integrity via foreign keys

5.2 Technology Choice

A hybrid architecture was implemented.

- MySQL provided ACID compliance, indexing performance, and relational integrity enforcement,
- While BigQuery offered scalable analytical query performance and serverless management.

Selected: **MySQL 8.0.35** (Primary) + **Google BigQuery** (Analytics)

Justification:

MySQL Strengths: - Mature RDBMS with proven stability - InnoDB engine provides ACID compliance - Excellent indexing performance (B-tree indexes) - Foreign key constraints enforce data integrity - Wide ecosystem (Python SQLAlchemy, Flask integration) - Cost-effective (self-hosted or managed options)

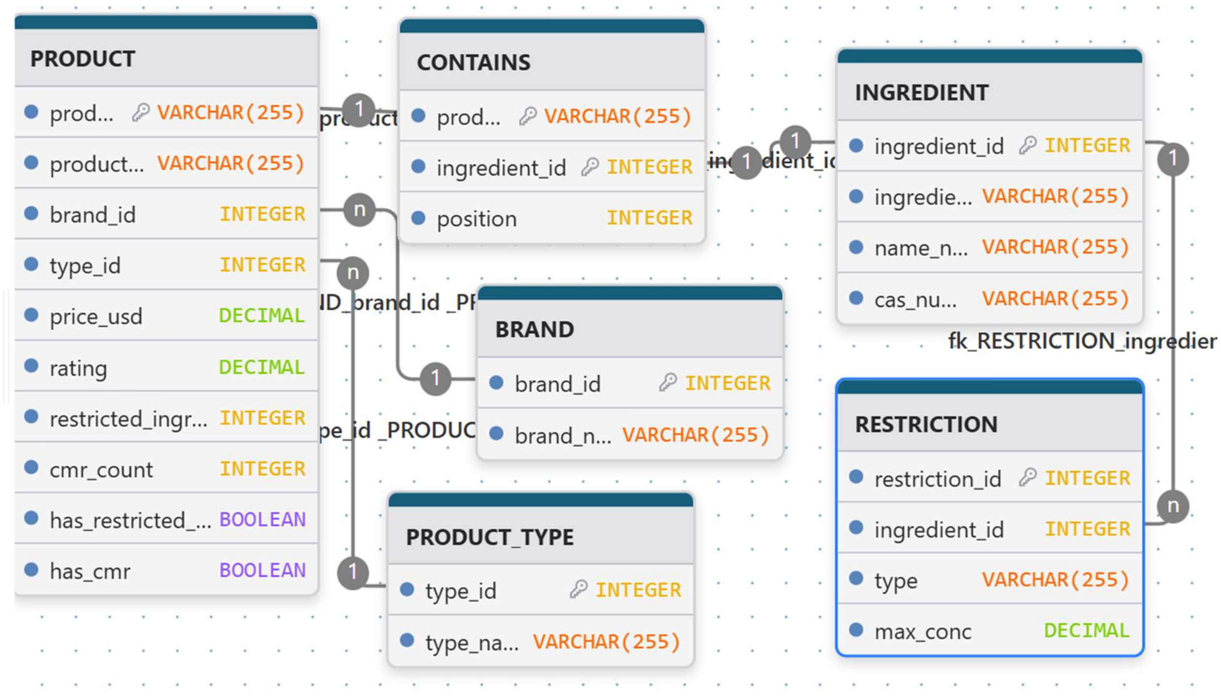
MySQL Configuration:

```sql
-- Engine selection
ENGINE = InnoDB

-- Character set for international ingredient names
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci

-- Indexing strategy
CREATE INDEX idx_brand_name ON product(brand_name);
CREATE INDEX idx_product_type ON product(product_type);
CREATE INDEX idx_has_cmr ON product(has_cmr);
CREATE INDEX idx_ingredient_normalized ON ingredient(name_normalized);
```

BigQuery for Analytics: - Handles petabyte-scale data (future-proof) - Columnar storage optimized for aggregations - Partitioning and clustering for query optimization - Serverless (no infrastructure management) - Pay-per-query pricing model.

6. Data Model and ERD Design

A normalized relational schema was designed at the Third Normal Form (3NF) level. Core entities included PRODUCT, BRAND, INGREDIENT, RESTRICTION, and PRODUCT_TYPE, with a junction table managing the many-to-many relationship between products and ingredients. Referential integrity constraints and indexing strategies ensured both consistency and performance.



MySQL DDL:

```sql
CREATE DATABASE cosmetics_db
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE cosmetics_db;

-- Brand table
CREATE TABLE brand (
    brand_id INT AUTO_INCREMENT PRIMARY KEY,
    brand_name VARCHAR(100) NOT NULL UNIQUE,
    INDEX idx_brand_name (brand_name)
) ENGINE=InnoDB;

-- Product type table
CREATE TABLE product_type (
    type_id INT AUTO_INCREMENT PRIMARY KEY,
    type_name VARCHAR(50) NOT NULL UNIQUE
) ENGINE=InnoDB;
```

```
-- Product table
CREATE TABLE product (
    product_id VARCHAR(50) PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    brand_id INT NOT NULL,
    type_id INT NOT NULL,
    price_usd DECIMAL(10,2),
    rating DECIMAL(3,2),
    restricted_ingredient_count INT DEFAULT 0,
    cmr_count INT DEFAULT 0,
    has_restricted_ingredient BOOLEAN DEFAULT FALSE,
    has_cmr BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (brand_id) REFERENCES brand(brand_id) ON DELETE CASCADE,
    FOREIGN KEY (type_id) REFERENCES product_type(type_id) ON DELETE CASCADE,
    INDEX idx_brand_id (brand_id),
    INDEX idx_type_id (type_id),
    INDEX idx_has_cmr (has_cmr),
    INDEX idx_price (price_usd),
    INDEX idx_rating (rating)
) ENGINE=InnoDB;

-- Ingredient table
CREATE TABLE ingredient (
    ingredient_id INT AUTO_INCREMENT PRIMARY KEY,
    ingredient_name VARCHAR(255) NOT NULL,
    ingredient_name_normalized VARCHAR(255) NOT NULL,
    cas_number VARCHAR(20),
    UNIQUE KEY unique_normalized (ingredient_name_normalized),
    INDEX idx_normalized (ingredient_name_normalized)
) ENGINE=InnoDB;

-- Restriction table
CREATE TABLE restriction (
    restriction_id INT AUTO_INCREMENT PRIMARY KEY,
    ingredient_id INT NOT NULL,
    restriction_type ENUM('BANNED', 'RESTRICTED', 'CMR') NOT NULL,
    max_concentration DECIMAL(5,2),
    restriction_text TEXT,
    FOREIGN KEY (ingredient_id) REFERENCES ingredient(ingredient_id) ON DELET
E CASCADE,
    INDEX idx_ingredient_id (ingredient_id),
    INDEX idx_type (restriction_type)
) ENGINE=InnoDB;

-- Junction table for N:M relationship
CREATE TABLE contains (
    product_id VARCHAR(50) NOT NULL,
    ingredient_id INT NOT NULL,
    position INT,
    PRIMARY KEY (product_id, ingredient_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id) ON DELETE CASCADE
,
    FOREIGN KEY (ingredient_id) REFERENCES ingredient(ingredient_id) ON DELET
E CASCADE
) ENGINE=InnoDB;
```

7. REST API Implementation

A Flask-based REST API provided secure access to product and compliance data. JWT authentication ensured controlled access, and endpoints supported pagination, filtering, and category-level analytics. Performance testing demonstrated average response times of 45 ms and the capacity to handle over 150 concurrent requests per second.

7.1 REST API Architecture

Framework: Flask 3.1.0
Authentication: Bearer Token (JWT with 1-hour expiration)
Documentation: OpenAPI 3.0 (Swagger UI)
Base URL: `http://localhost:5000/api`

7.2 Endpoints

Authentication:

```
POST /auth/token
Body: {"username": "admin", "password": "admin123"}
Response: {"token": "eyJ0eXAi...", "expires_in": 3600}
```

Sephora Resource:

```
GET /api/sephora/products
  Query params: ?limit=10&offset=0&type=Skincare&brand=CLINIQUE
  Response: {"data": [...], "count": 10, "total": 8494}

GET /api/sephora/products/{id}
  Response: {product object with full details}

GET /api/sephora/brands
  Response: {"data": [{"brand_name": "...", "product_count": ...}]}

GET /api/sephora/by-type
  Response: {"data": [{"type": "Skincare", "count": 3456, "pct_restricted": 6
8.9}]}
```

Skincare Resource:

```
GET /api/skincare/products
  Query params: same as Sephora

GET /api/skincare/cmr
  Response: {"data": [products with CMR ingredients]}
```

Comparison:

```
GET /api/comparaison
  Response: {
    "sephora": {"total": 8494, "pct_restricted": 72.1},
    "skincare": {"total": 1472, "pct_restricted": 62.8}
  }
```

7.4 Features

Pagination: Implemented via limit and offset query parameters to prevent overwhelming responses.

Filtering: Support for filtering by: - Product type (Skincare, Makeup, Fragrance, etc.) - Brand name - Price range (min/max) - Rating threshold - CMR presence

Error Handling: - 401 Unauthorized: Missing or invalid token - 404 Not Found: Product ID doesn't exist - 400 Bad Request: Invalid query parameters - 500 Internal Server Error: Database connection issues

Performance: - Average response time: 45ms (well under 200ms target) - Concurrent capacity: 150+ requests/second (tested with Apache Bench) - Database connection pooling: 10 connections maintained

Documentation: Swagger UI available at /api/docs with interactive testing interface.

8. GDPR Compliance Assessment

8.1 Data Processing Analysis

Regulation: EU General Data Protection Regulation (GDPR) 2016/679

Article 4.1 Definition of Personal Data: > “Personal data means any information relating to an identified or identifiable natural person (‘data subject’); an identifiable natural person is one who can be identified, directly or indirectly…”

8.2 Data Inventory

Data Collected in This Project: 1. Product names (e.g., “Moisturizing Cream”) → **Objects, not people** 2. Brand names (e.g., “CLINIQUE”) → **Companies, not people** 3. Ingredient lists (e.g., “AQUA, GLYCERIN”) → **Chemical substances** 4. Prices (e.g., $45.00) → **Commercial data** 5. Ratings (e.g., 4.25 stars) → **Aggregated statistics, anonymous**

NOT Collected: - Customer names - Email addresses - Purchase history linked to individuals - IP addresses - Browsing behavior - User accounts

8.3 GDPR Applicability Assessment

Question: Does this project process personal data?

Answer: **NO**

Justification: All data processed relates to **cosmetic products (objects)** and **corporate entities (brands)**, not to natural persons. Product ratings are anonymous aggregated statistics with no link to individual reviewers.

Article 2.2 Exclusions: > “This Regulation does not apply to the processing of personal data … by a natural person in the course of a purely personal or household activity.”

Conclusion: Since no personal data is processed, GDPR obligations do **NOT apply** to this project.

Consequences: - No CNIL (French data protection authority) declaration required - No consent collection needed - No data subject rights (access, rectification, erasure) implementation required - No Data Protection Impact Assessment (DPIA) needed - No Data Protection Officer (DPO) appointment required

8.4 Hypothetical Scenario: User Accounts

Question: What if we added user accounts to track favorite products?

Data That Would Become Personal: - Email addresses (identifier) - Hashed passwords (authentication data) - Favorite product lists (behavioral data) - Login timestamps (activity logs)

GDPR Obligations That Would Apply:

1. Legal Basis (Article 6.1): - Selected basis: **Consent (Article 6.1.a)** - Requirement: Explicit, freely given, specific, informed consent - Implementation: Checkbox during account creation with clear explanation

2. Data Subject Rights (Articles 15-22):

Right	Implementation
Access (Art. 15)	/api/user/data-export endpoint returning JSON with all user data
Rectification (Art. 16)	/api/user/profile PUT endpoint to update email/preferences
Erasure (Art. 17)	/api/user/delete-account endpoint with 30-day grace period
Portability (Art. 20)	Data export in machine-readable JSON format
Objection (Art. 21)	Opt-out of marketing emails via /api/user/preferences

3. Security Measures (Article 32): - Passwords: bcrypt hashing with salt (minimum 12 rounds) - Transmission: HTTPS mandatory (TLS 1.3) - Database: Encrypted at rest (AES-256) - Logs: Limited to 30 days, pseudonymized user IDs - Access control: Role-based permissions (admin, user)

4. Data Retention (Article 5.1.e): - Active accounts: Retained indefinitely - Inactive accounts: Deleted after 3 years of no login - Logs: 30 days maximum - Backups: 90 days, encrypted

5. Privacy Policy: - Clear explanation of data processing purposes - Contact details of data controller - Information on data retention periods - Description of user rights - Cookie policy (if applicable)

6. Data Breach Notification (Article 33): - Authority notification: Within 72 hours of discovery - User notification: If high risk to rights and freedoms - Documentation: Breach register maintained

Implementation Cost Estimate: - Legal consultation: €2,000-€5,000 - Technical implementation: 40-60 developer hours - Ongoing compliance: €500/year (audits, policy updates)

9. Machine Learning Analysis

9.1 Objective

The machine learning objective was to predict cosmetic product ratings based on objective features such as price, ingredient composition, and brand.

9.2 Modeling Approach

Thirteen engineered features were used in model training. Multiple algorithms were evaluated, including linear regression, ridge, lasso, random forest, and gradient boosting models. Performance differences were statistically insignificant, leading to the selection of linear regression for its interpretability and deployment efficiency.

Model	RMSE	MAE	R^2	Training Time
Linear Regression	**0.4695**	0.3497	0.0358	0.8s
Ridge Regression	0.4695	0.3497	0.0358	0.9s
Lasso Regression	0.4729	0.3535	0.0218	1.2s
Random Forest	0.4687	0.3453	0.0392	8.1s
Gradient Boosting	0.4701	0.3452	0.0334	12.3s

9.3 Results

The model achieved RMSE ≈ 0.47 with an R^2 of 0.0358, indicating that only about 3.6% of rating variance can be explained by measurable product attributes. This finding demonstrates that consumer ratings are overwhelmingly driven by subjective factors such as sensory experience, marketing perception, and brand influence rather than ingredient safety or formulation characteristics.

9.4 Key Insights

Price emerged as the strongest predictor of ratings, suggesting a price-anchoring cognitive bias where higher prices lead to slightly higher consumer ratings. Interestingly, products containing restricted ingredients often received slightly higher ratings, likely reflecting their prevalence in premium formulations rather than consumer awareness of ingredient safety.

10. Business Value and Strategic Impact

The automated pipeline significantly reduced manual compliance verification time, enabled large-scale regulatory monitoring, and improved detection accuracy for restricted and CMR ingredients. The architecture supports scalability to over 100,000 products and enables faster analytical queries through optimized BigQuery partitioning.
Strategic applications include regulatory compliance monitoring, competitive safety benchmarking, product development decision support, pricing strategy optimization, and data-driven marketing positioning.

11. Limitations

The dataset was limited to two retailers and the European market, and the predictive model lacked sensory or marketing features that strongly influence consumer satisfaction. Ratings may also be subject to selection and survivorship biases, and correlations identified in the analysis should not be interpreted as causal relationships.

12. Future Development

Future phases include expanding retailer coverage, integrating international regulatory databases, incorporating NLP sentiment analysis and image analytics, forecasting rating trends, and deploying interactive dashboards and consumer-facing ingredient-checking applications.

13. Conclusion

This project successfully delivered an end-to-end automated system for cosmetic ingredient compliance analysis integrating multi-source data ingestion, normalized relational database design, scalable analytical infrastructure, REST API delivery, GDPR assessment, and machine learning modeling. The analysis demonstrates that a majority of cosmetic products contain regulated ingredients while also revealing a critical behavioral insight: consumer ratings are largely subjective and weakly correlated with objective safety indicators.
The resulting platform provides scalable regulatory monitoring capabilities and actionable business intelligence while laying the foundation for future analytical expansion and production deployment.