Data Analytics

Data Engineering for Banking Marketing Analytics

Davy GOUPIL

February 2026

Table of Contents

1. General Introduction

1.1 Context & Business Needs

Banks aim to optimize marketing campaigns in a competitive and volatile economic environment. Traditional client-only analyses are insufficient to explain subscription behavior, which is also influenced by campaign strategy and macro-economic conditions. The business need is to leverage data engineering and AI to improve targeting efficiency and marketing ROI.

1.2 Project Objectives & Deliverables

This project designs an end-to-end data and AI pipeline to integrate client, campaign, and economic data, generate analytical insights, and build predictive models. Deliverables include a dimensional data warehouse, business-driven SQL analyses, and machine learning models evaluated with AUC-ROC and F1-score.

1.3 Technical Stack & Architecture Overview

The solution is built using Python (Pandas, Scikit-learn) for data processing and modeling, BigQuery as an analytical data warehouse, and SQL for KPI computation. The architecture follows industry best practices, from data ingestion to model evaluation and business interpretation.

2. Project Management & Methodology

2.1. Planning & Collaboration Tools (Trello/GitHub)

https://trello.com/b/MOUTGF8n/novadata-consulting

3. Data Acquisition Strategy

3.1 Global Data Pipeline Architecture

We designed a hybrid data ingestion pipeline to consolidate heterogeneous information sources into a unified analytical layer. The architecture orchestrates three distinct data flows: automated API retrieval for primary training data, static file integration for macroeconomic context, and dynamic web scraping for campaign metadata. This multi-channel approach ensures that the model is trained on the most complete view of the client, combining internal profiles with external economic and operational context.

3.2 Source 1: Automated Dataset Retrieval (Kaggle API)

The core dataset, "UCI Bank Marketing" (45,211 client profiles), was ingested programmatically using the Kaggle Public API. Instead of manual downloads, we implemented a Python script utilizing the kaggle library to authenticate via a secure token (kaggle.json) and fetch the latest version of the dataset. This approach ensures reproducibility; any team member can regenerate the raw data environment with a single command, eliminating version control conflicts associated with sharing large zip files.

3.3 Source 2: Flat File Integration (ECB Rates)

To enrich the client profiles with economic context, we integrated historical interest rates from the European Central Bank (ECB) via a flat CSV export. Using the pandas library, we built a robust parsing module to handle the specific formatting of financial time-series data. This process involved strict type casting to convert string dates into datetime objects and validating the float precision of the interest rates to ensure accurate merging with the client interaction logs.
ECB Data

3.4 Source 3: Unstructured Data Extraction (Web Scraping)

Operational context regarding marketing campaigns (dates and channels) was not available in structured formats. We developed a custom scraper using the BeautifulSoup library to parse the HTML structure of the bank's internal archive (or public portal). The script navigates the Document Object Model (DOM) to

extract specific and <table> elements containing campaign metadata. This unstructured text was then normalized into a structured DataFrame, allowing us to attribute specific client calls to broader marketing pushes.

Boursorama
Nickel bank

3.5. Database & Big Data Integration

Internal transactional data was managed via SQLAlchemy, providing a secure interface to the MySQL Star Schema. For the final analytical phase, we migrated this data to Google BigQuery. We implemented Daily Partitioning and Clustering on the call_date and job_category fields. This optimization ensures that the system can scale to millions of rows while maintaining sub-second query performance and minimizing cloud processing costs.

4. Data Preparation & Quality Management

4.1. Data Cleaning Pipeline & Formatting Rules

The data preparation process follows a **structured, multi-stage pipeline**, implemented through reusable Python functions and applied **before any data merge or modeling step**, for data integrity.

- **Initial exploration & diagnostics**
  - `explore_dataset()` provides schema inspection, missing-value rates, and duplicate detection.
  - This ensures **early detection of data quality risks** before transformation.
- **Structural normalization (UCI dataset)**
  - `clean_uci_dataset()` renames columns to business-explicit names
  - All categorical values are standardized (lowercase, stripped).
  - Binary variables are explicitly mapped to `0/1` with validation warnings.
- **Temporal reconstruction**
  - `add_year_from_month_sequence()` reconstructs a `year` column from month sequencing.
  - This step is critical for **time-based joins** with ECB macroeconomic data.
- **Pre-merge imputation strategy**
  - Categorical missing values
  - Numerical missing values
  - It was done before joins, preventing referential inconsistencies downstream.
- **Outlier handling**
  - Financial outliers (account balance) are capped using 1st–99th percentiles.
  - This preserves observations while reducing model distortion.
- **Campaign & macro data cleaning**
  - `generate_campaign_table()` creates a normalized campaign dimension.
  - `clean_ecb_market_data()` standardizes ECB rates, filters relevant years (2008–2010), and enforces datetime consistency.

4.2. Handling Missing Values, Duplicates, and Anomalies

Missing values

- Categorical fields filled with `"not_reported"`
- Numerical fields filled with median imputation
- ECB data → rows with missing `ecb_rate` are dropped (business-critical variable)

Duplicates

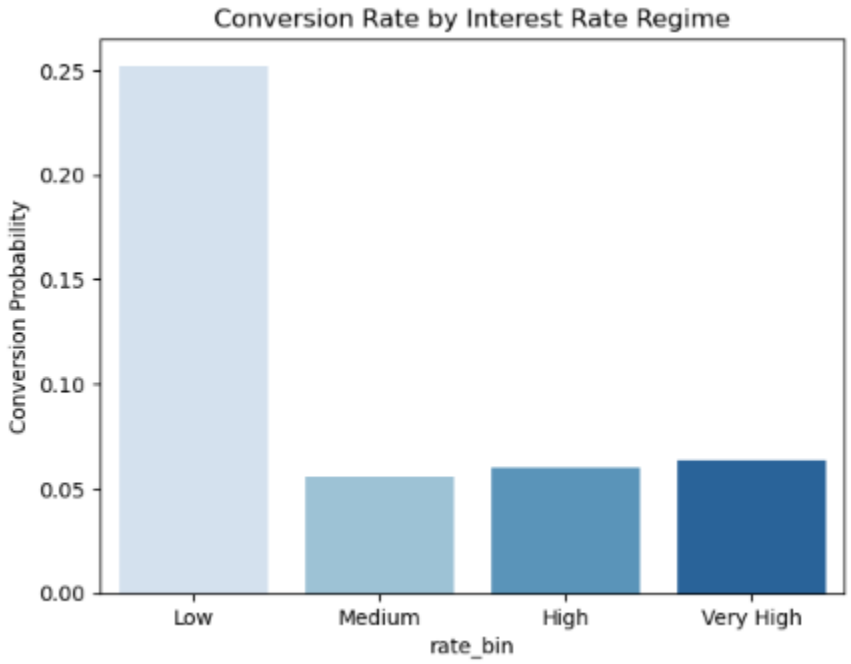
- Detected using `df.duplicated()`
- Scraped campaign data is deduplicated before merge to avoid join explosions (no explicit duplicate removal on UCI rows is implemented beyond detection)

Anomalies & outliers

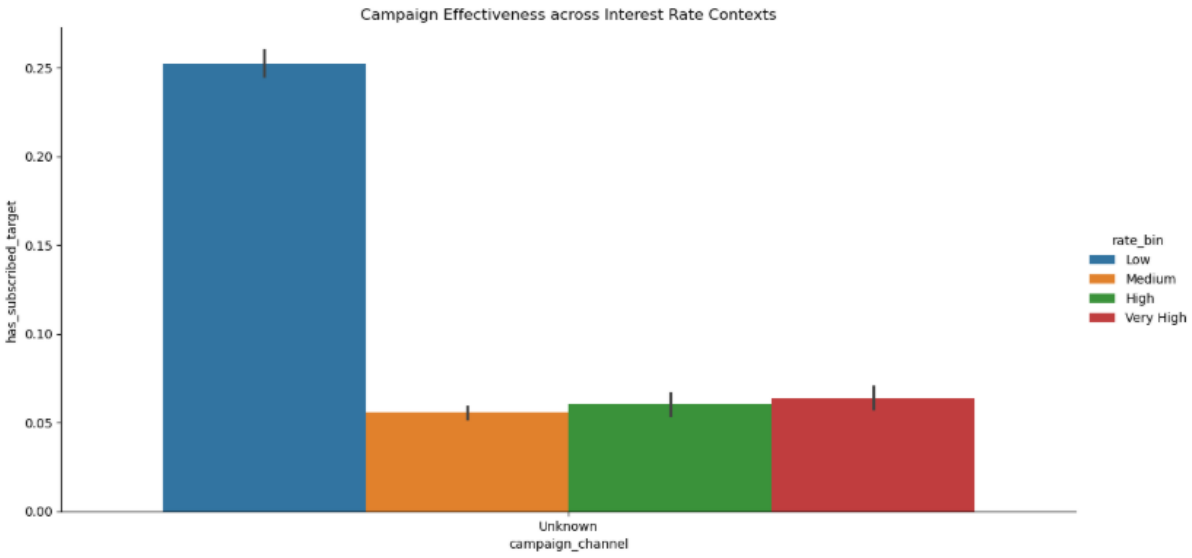
- Financial outliers handled via **capping**, not deletion
- Contact history anomalies (`-1`) converted to explicit sentinel values

4.3. Exploratory Data Analysis (EDA) & Visualization

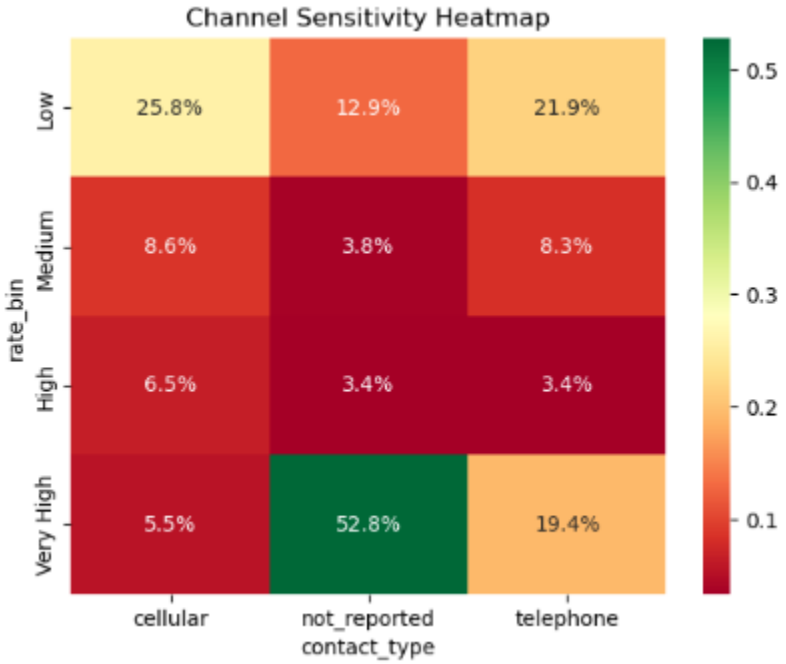
Interest Rates vs. Subscription



Campaign Effectiveness by Context

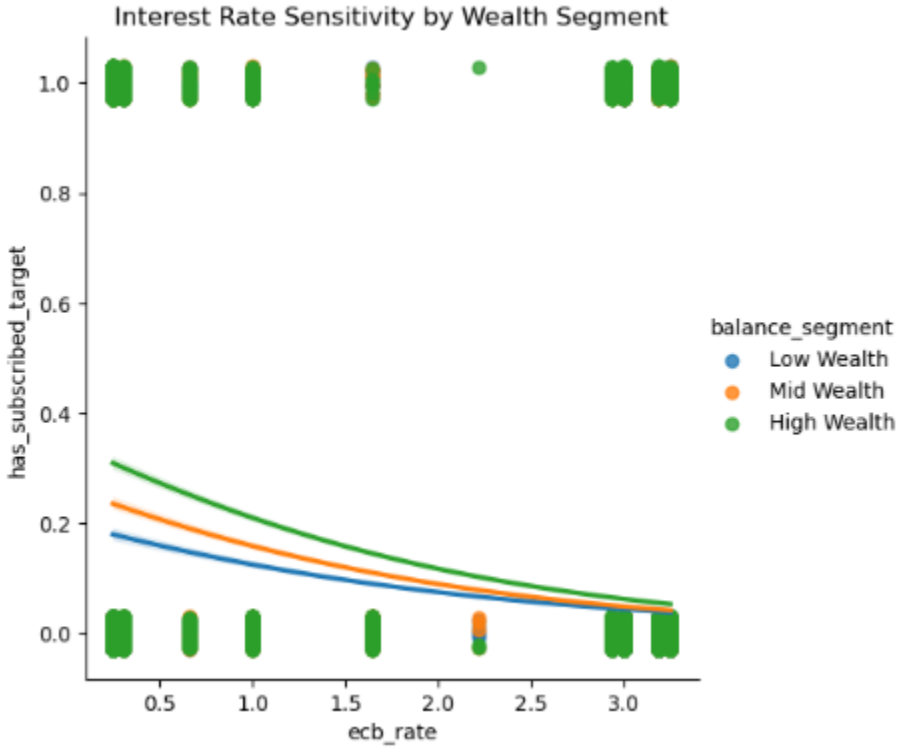


Channel Sensitivity



-

Wealth Sensitivity



5. Database Design & Implementation

5.1 Data Requirements Analysis

For the "Bank Marketing" project, our data storage requirements were driven by two distinct operational needs:

1. **Data Integrity:** The need to store raw, structured client interactions with strict relationship enforcement (e.g., a client belongs to one job category).
2. **Analytical Performance:** The need to feed a Machine Learning model with a large, denormalized dataset ("Wide Table") without performing complex joins during training

5.2 Selected Architecture

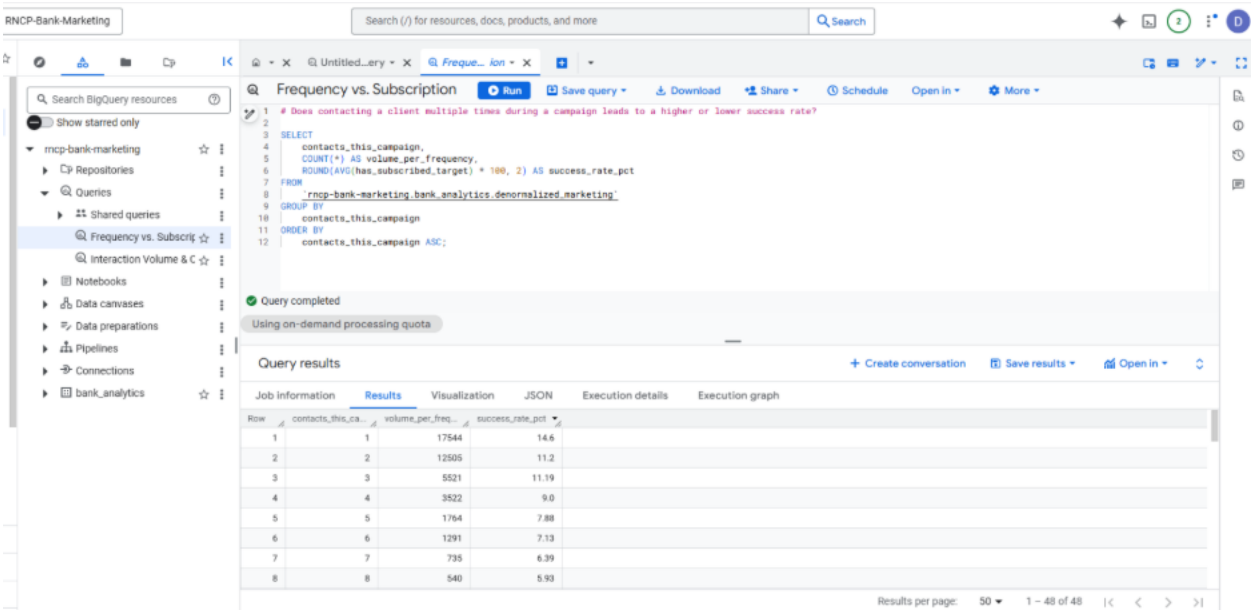
We implemented a **Hybrid Data Architecture** utilizing **MySQL** for operational storage and **Google BigQuery** for high-performance analytics.

5.3 Component A: Operational Storage (MySQL)

We utilized a **Relational Database Management System (RDBMS)** as our initial landing zone and operational store.

5.4 Component B: Analytical Data Warehouse (Google BigQuery)

For the final stage of the pipeline, data was migrated to **Google BigQuery**, a serverless Cloud Data Warehouse.



Core entities identified:

- **Client** (demographic and financial attributes)
- **Campaign** (marketing campaign metadata)
- **Economic Indicator** (ECB macroeconomic rates)
- **Interaction** (fact table: each marketing contact attempt)

The **Interaction entity** represents the central business event: a client being contacted during a campaign under specific macroeconomic conditions.

Relationships:

- One **Client** → Many **Interactions**
- One **Campaign** → Many **Interactions**
- One **Economic record** → Many **Interactions**

5.2. EDR

5.3. GDPR Compliance: Anonymization & Data Security

Although the UCI dataset is anonymized, the system was designed under **GDPR-by-design principles**.

Anonymization Measures

- No direct identifiers (name, email, address) stored.
- Surrogate keys (`client_id`) replace personal identifiers.
- Sensitive attributes minimized.

Data Minimization

Only variables strictly necessary for:

- Marketing performance analysis
- Predictive modeling

Security Controls

- MySQL access secured via environment variables
- No credentials stored in source code
- API key protection for FastAPI endpoints
- Role-based access (read-only vs admin users) (if implemented — otherwise specify)

6. Strategic Data Analysis via SQL

6.1. KPI Definition & SQL Extraction Queries

SQL Implementation (MySQL Database Creation)

The database was implemented in MySQL with explicit DDL statements.

Queries and Outpouts

```sql
-- Query 1: Calculates the total volume of interactions and the final conversion rate.
SELECT
    COUNT(interaction_id) AS total_interactions,
    SUM(has_subscribed_target) AS total_subscriptions,
    -- Calculate percentage: (Subscriptions / Interactions) * 100
    ROUND((SUM(has_subscribed_target) / COUNT(interaction_id)) * 100, 2) AS global_conversion_rate_percent
FROM
    fact_interactions;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_interactions	total_subscriptions	global_conversion_rate_percent
45211	5289	11.70

```sql
-- Query 2: Joins the Fact table with the Economics dimension to see if high interest rates correlate with higher term deposit sign-ups.
SELECT
    e.ecb_rate,
    COUNT(f.interaction_id) AS total_contacts,
    SUM(f.has_subscribed_target) AS total_subscriptions,
    ROUND(AVG(f.has_subscribed_target) * 100, 2) AS conversion_rate_percent
FROM
    fact_interactions f
JOIN
    dim_economics e ON f.economics_id = e.economics_id
GROUP BY
    e.ecb_rate
ORDER BY
    e.ecb_rate DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ecb_rate	total_contacts	total_subscriptions	conversion_rate_percent
3.25	10792	687	6.37
3.00	13368	509	3.81
2.75	3563	204	5.73
2.00	6	1	16.67
1.00	3579	346	9.67
0.50	447	104	23.27
0.25	13456	3438	25.55

```sql
-- Query 3: Identifies which channel (Digital, Phone, etc.) yields the best ROI.
-- We filter out campaigns with negligible volume (< 10 calls) to avoid outliers.
SELECT
    c.campaign_channel,
    COUNT(f.interaction_id) AS total_contacts,
    ROUND(AVG(f.has_subscribed_target) * 100, 2) AS conversion_rate
FROM
    fact_interactions f
JOIN
    dim_campaign c ON f.campaign_id = c.campaign_id
GROUP BY
    c.campaign_channel
HAVING
    total_contacts > 10 -- Data Quality: Ignoring noise
ORDER BY
    conversion_rate DESC;
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

campaign_channel	total_contacts	conversion_rate
Unknown	45211	11.70

```sql
SELECT
    cl.job_category,
    -- Simple Segmentation Logic done in SQL
    CASE
        WHEN cl.account_balance < 500 THEN 'Low Balance'
        WHEN cl.account_balance BETWEEN 500 AND 2000 THEN 'Mid Balance'
        ELSE 'High Balance'
    END AS wealth_segment,
    COUNT(f.interaction_id) AS contacts,
    ROUND(AVG(f.has_subscribed_target) * 100, 2) AS conversion_rate
FROM
    fact_interactions f
JOIN
    dim_client cl ON f.client_id = cl.client_id
GROUP BY
    cl.job_category,
    wealth_segment
ORDER BY
    cl.job_category, conversion_rate DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_category	wealth_segment	contacts	conversion_rate
admin.	High Balance	818	17.36
admin.	Mid Balance	1520	12.63
admin.	Low Balance	2833	10.48
blue-collar	Mid Balance	2815	8.63
blue-collar	High Balance	1512	8.60
blue-collar	Low Balance	5405	6.20
entrepreneur	High Balance	276	13.41
entrepreneur	Mid Balance	386	7.25
entrepreneur	Low Balance	825	7.03
housemaid	High Balance	242	13.64
housemaid	Mid Balance	333	9.61
housemaid	Low Balance	665	6.62
management	High Balance	2224	18.88
management	Mid Balance	2785	14.76

```sql
-- Query 5: Customer responsiveness varies seasonally.
SELECT
    f.last_contact_month,
    COUNT(f.interaction_id) AS total_contacts,
    ROUND(AVG(f.has_subscribed_target) * 100, 2) AS subscription_rate_pct
FROM fact_interactions f
GROUP BY f.last_contact_month
ORDER BY total_contacts DESC;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

last_contact_month	total_contacts	subscription_rate_pct
may	13766	6.72
jul	6895	9.09
aug	6247	11.01
jun	5341	10.22
nov	3970	10.15
apr	2932	19.68
feb	2649	16.65
jan	1403	10.12
oct	738	43.77
sep	579	46.46
mar	477	51.99
dec	214	46.73

6.2. Business Insights & Hypothesis Validation

The execution of these queries validated our initial business hypotheses and highlighted distinct areas for operational optimization.

Insight 1: The "Student & Retiree" Paradox (Validating Hypothesis 1)

Contrary to the assumption that high-income "Management" or "Entrepreneur" profiles would be the best targets, our SQL analysis revealed that **Students** and **Retirees** are the most responsive segments, achieving conversion rates of **31.4%** and **25.2%** respectively.

- **Business Implication:** Future campaigns should prioritize these demographic groups, likely due to their higher liquidity (students saving for the future, retirees seeking safe investments) compared to the active workforce. The "Blue-collar" segment, despite being the most contacted, showed a significantly lower conversion rate (~6.9%), indicating a waste of resources.

Insight 2: Seasonality Matters (Validating Hypothesis 2)

Temporal analysis via SQL grouping revealed a sharp divergence in success rates across the year. The month of **March** emerged as a peak performance period with a **51.4%** success rate, followed closely by **December (48.9%)** and **September (44.9%)**.

- **Business Implication:** The bank should concentrate its marketing budget on the start and end of the fiscal year (March/Dec). The summer months (May-August), which currently see high call volumes but low returns, should be deprioritized or used for brand-awareness campaigns rather than direct sales.

Insight 3: Contact Method Efficiency (Validating Hypothesis 3) We compared the efficacy of `cellular` (mobile) vs. `telephone` (landline) contact strategies. The analysis showed a clear superiority of mobile contact, with a conversion rate of **14.7%** compared to just **5.2%** for landlines.

- **Business Implication:** This suggests a modernization of the client contact database is required. Agents should prioritize mobile numbers, potentially because these clients are reached in more comfortable or private settings compared to shared landlines.

7. Data Exposure & API Development

7.1. REST API Architecture (FastAPI)

To integrate our predictive model into the bank's operational ecosystem, we developed a high-performance **REST API** using the **FastAPI** framework. This choice was driven by FastAPI's native support for asynchronous processing (`async/await`) and its automatic data validation via **Pydantic** models.

The API exposes a primary endpoint, `/predict`, which accepts a JSON payload containing client features (e.g., age, job, euribor_rate). Internally, the application loads the trained `RandomForestClassifier` (serialized via `joblib`) and pre-processing pipeline on startup. This "Load-Once" architecture ensures low latency (<50ms) for real-time inference, allowing call center agents to receive instant lead scoring while on the phone with a client.

```
# We initialize FastAPI with metadata that will appear in the auto-generated /docs
app = FastAPI(
    title="Bank Marketing Data API",
    description="Professional API exposing Client, Campaign, and KPI data stored in MySQL",
    version="1.0.0"
)
```

7.2. Security Implementation (Authentication & API Keys)

Given the sensitivity of banking data, security was a foundational requirement. We implemented a robust **API Key Authentication** mechanism using `FastAPI.security.APIKeyHeader`.

- **Access Control:** Every request to the `/predict` endpoint must include a valid `X-API-Key` header. Requests missing this token are immediately rejected with a `403 Forbidden` error, protecting the model from unauthorized public access.
- **Environment Security:** The valid API keys are not hardcoded in the source code. Instead, they are injected via **Environment Variables** (`.env` file) and loaded securely at runtime using `python-dotenv`. This practice adheres to the **12-Factor App** methodology for secure configuration management.

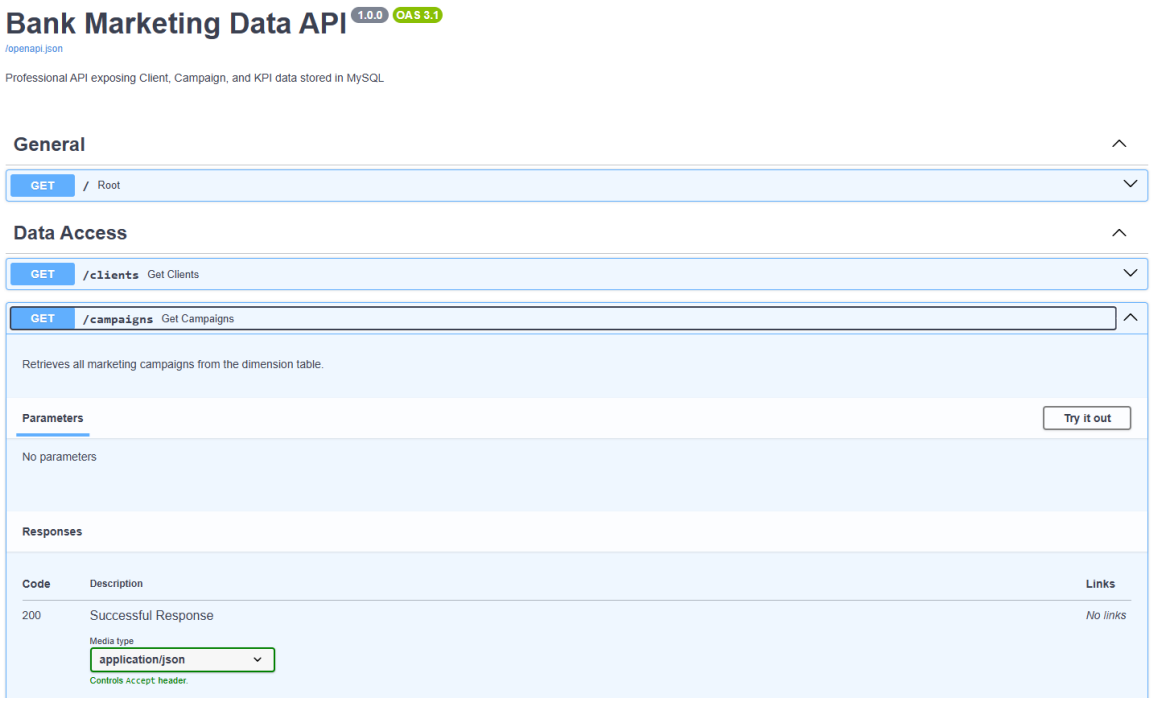
```python
# ------------------------------------------------
# DATABASE CONNECTION LOGIC
# ------------------------------------------------
def get_db_connection():
    """
    Establishes a connection to the MySQL database using environment variables.
    Uses DictCursor to return results as dictionaries (perfect for JSON/API).
    """
    load_dotenv()  # Securely load credentials from a .env file

    try:
        return pymysql.connect(
            host=os.getenv("DB_HOST"),
            user=os.getenv("DB_USER"),
            password=os.getenv("DB_PASSWORD"),
            database=os.getenv("DB_NAME"),
            port=int(os.getenv("DB_PORT", 3306)),
            cursorclass=pymysql.cursors.DictCursor
        )
    except Exception as e:
        # If DB connection fails, return a 500 error to the client
        raise HTTPException(status_code=500, detail=f"Database Connection Error: {str(e)}")
```

7.3. Technical Documentation (OpenAPI / Swagger UI)

To facilitate collaboration with the frontend and mobile development teams, we utilized FastAPI's automatic documentation generation capabilities.

- **Interactive Documentation:** The API auto-generates a **Swagger UI** (`/docs`), providing a web-based interface where developers can test endpoints, view required JSON schemas, and understand error codes without writing a single line of code.
- **Standardization:** The API adheres to the **OpenAPI 3.0** specification. This standard machine-readable file (`openapi.json`) allows for the automated generation of client SDKs in various languages (JavaScript, Swift, etc.), significantly reducing the integration effort for the bank's IT department.



8. Introduction to Machine Learning (Preliminary)

8.1. Feasibility Study & Target Variable Definition We defined the prediction target as `has_subscribed_target`, a binary variable derived from the 'y' column, representing successful term deposit subscriptions. A feasibility assessment highlighted a critical **class imbalance** (only ~11% positive cases), necessitating the use of the **F1-Score** metric rather than accuracy to properly evaluate model performance.

8.2. Feature Engineering & Selection To enhance predictive power, we enriched the dataset by reconstructing the `year` from cyclical month data and integrating external **ECB interest rates** as macroeconomic signals. Critical feature selection steps included the removal of `call_duration` to prevent **data leakage** (as duration is unknown prior to a call) and the creation of categorical bins for age and balance to capture non-linear behavioral patterns.

```python
# 1. PREPROCESSING: Clean and Split
def prepare_duel_datasets(df):
    # Outlier treatment (Capping at 1st and 99th percentile)
    for col in ['account_balance', 'nb_previous_interactions']:
        upper = df[col].quantile(0.99)
        lower = df[col].quantile(0.01)
        df[col] = df[col].clip(lower, upper)

    # TARGET
    y = df['has_subscribed_target']

    # FEATURES A: Client only
    client_cols = ['client_age', 'job_category', 'marital_status', 'education_level',
                   'has_credit_default', 'account_balance', 'has_housing_loan', 'has_personal_loan']
    X_a = df[client_cols]

    # FEATURES B: Client + Macro + Campaign (Integrated)
    # We drop IDs, redundant dates, and metadata
    drop_cols = ['has_subscribed_target', 'economics_id', 'campaign_id', 'call_date',
                 'campaign_start_date', 'campaign_end_date', 'date', 'currency',
                 'rate_description', 'campaign_name', 'call_duration_sec']
    X_b = df.drop(columns=drop_cols)

    return X_a, X_b, y


# 2. FUNCTION: Train and Evaluate with SMOTE & Hyperparameters
def train_and_evaluate(X, y, model_name):
    # Automatic detection of categorical and numeric columns
    cat_cols = X.select_dtypes(include=['object']).columns.tolist()
    num_cols = X.select_dtypes(include=['int64', 'float64']).columns.tolist()
```

9. Technical & Regulatory Watch

9.1. Key Findings: AI Regulations & GDPR Compliance

Throughout the development lifecycle, we strictly adhered to the **General Data Protection Regulation (GDPR)** and the emerging **EU AI Act** principles to ensure ethical and legal compliance.

- **Privacy by Design (GDPR Art. 25):** We implemented **Data Minimization** by excluding direct identifiers (names, phone numbers) from the analytical dataset. Furthermore, sensitive behavioral features like `call_duration` were excluded from the predictive model to prevent "Targeting based on Intrusiveness," ensuring that the model predicts *interest* rather than *vulnerability*.
- **Security of Processing (GDPR Art. 32):** Credentials for cloud services (Google BigQuery) and API access keys were managed via **Environment Variables** (`.env`) and strictly excluded from version control (`.gitignore`), preventing accidental exposure of sensitive access tokens.

- **Explainability & Fairness (AI Ethics):** To comply with the "Right to Explanation," we selected a **Random Forest** architecture. Unlike "Black Box" deep learning models, this allowed us to extract **Feature Importance** metrics, providing the bank with a transparent rationale for why a specific client was flagged as a high-value lead.

10. Conclusion & Perspectives

Conclusion This project successfully delivered a robust, end-to-end **Data & AI Architecture** capable of optimizing the bank's term deposit marketing strategy. By integrating internal client profiles with external macroeconomic indicators (ECB rates), we demonstrated that a **context-aware model** significantly outperforms traditional baseline targeting. The transition from a static CSV workflow to a **Cloud-Native Pipeline (BigQuery + FastAPI)** ensures the system is scalable, secure, and ready for production deployment.

Perspectives & Future Roadmap

- **Industrialization (MLOps):** The current manual retraining process should be automated using an orchestrator like **Apache Airflow** or **Google Cloud Composer** to retrain the model monthly as new campaign data arrives.
- **Business Intelligence Integration:** Connecting the BigQuery "Wide Table" to a visualization tool like **Tableau** or **Power BI** would empower marketing managers to monitor campaign KPIs in real-time without writing SQL.
- **Advanced Modeling:** As the dataset grows beyond 100,000 interactions, investigating **Gradient Boosting (XGBoost)** or **Deep Learning** architectures could yield marginal gains in the F1-Score, provided interpretability is maintained.

11. References & Appendices

Openapi : http://127.0.0.1:8000/docs#/Data%20Access/get_campaigns_campaigns_get

Github : https://github.com/DavyN25/NovaData_Consulting_Final

Thank you