Data Analytics

Thomas Giret

Churn Bank Customer's Prediction

🎯 Table of Content

1. How i worked

📕 **Planning :** Notes

📊 **Import data, data analysis and prediction :** Anaconda / jupyter notebook / python

🗂️ **Database and queries :** My sql Workbench

---

2. Data Collect & Context

Data Acquisition Methodology

The dataset was sourced from Kaggle and manually downloaded as a CSV file. It was imported into Python using the pandas library for inspection, cleaning, and transformation. API extraction was not used, as CSV files are preferred in business settings for their simplicity, reproducibility, security, and compatibility with existing tools. This approach ensures transparency, traceability, and aligns with standard data analysis practices.

Context

Working as a data scientist for a bank, job is to find the best churn prediction.
In other words, I am in charge of a model creation that finds the largest number of churn customers.

---

3. Metadata

Dataset Overview

The **Churn Modelling** dataset is commonly used in data science and machine learning to analyze customer behavior and predict **customer churn**, i.e., whether a customer is likely to leave a bank.

Each row represents **one customer**, and each column describes a **customer attribute** or the **target variable (churn)**.

A. Target Variable

🚪 Exited

- **Type:** Binary (0 / 1)
- **Description:** Indicates whether the customer has left the bank.
- **Values:**
  - `0` → Customer stayed
  - `1` → Customer left (churned)

This is the **dependent variable** used for churn prediction.

B. Customer Identification Variables

🔢 RowNumber

- **Type:** Integer
- **Description:** Index of the row in the dataset.
- **Usage:** Not useful for analysis or modeling.

🆔 CustomerId

- **Type:** Integer
- **Description:** Unique identifier for each customer.
- **Usage:** Identification only; should be removed before modeling.

👤 Surname

- **Type:** Categorical (String)
- **Description:** Customer's last name.
- **Usage:** Not predictive; usually removed.

C. Demographic Variables

🌍 Geography

- **Type:** Categorical
- **Description:** Country where the customer is located.
- **Typical values:** France, Spain, Germany
- **Usage:** Important predictor of churn behavior.

⚧ Gender

- **Type:** Categorical
- **Description:** Customer's gender.
- **Typical values:** Male, Female

🎂 Age

- **Type:** Numerical (Integer)
- **Description:** Customer's age in years.
- **Usage:** Strong predictor; churn often varies by age group.

D. Banking Relationship Variables

⏳ Tenure

- **Type:** Numerical (Integer)
- **Description:** Number of years the customer has been with the bank.

E. Financial Variables

💼 EstimatedSalary

- **Type:** Numerical (Float)
- **Description:** Estimated annual salary of the customer.

💳 CreditScore

- **Type:** Numerical (Integer)
- **Description:** Credit score assigned to the customer.
- **Usage:** Used to assess financial reliability

💰 Balance

- **Type:** Numerical (Float)
- **Description:** Account balance of the customer.

📦 NumOfProducts

- **Type:** Numerical (Integer)
- **Description:** Number of bank products used by the customer.

💳 HasCrCard

- **Type:** Binary (0 / 1)
- **Description:** Indicates whether the customer owns a credit card.
- **Values:**
  - `1` → Yes
  - `0` → No

⚡ IsActiveMember

- **Type:** Binary (0 / 1)
- **Description:** Indicates whether the customer is an active member.
- **Usage:** One of the strongest churn indicators..

---

4. Data Cleaning

✔️ Cleaning

- None **THAT**, none **duplicate**.
- Removing unnecessary columns:
  RowNumber, **Surname**

---

5. ERD

Creation Database process :

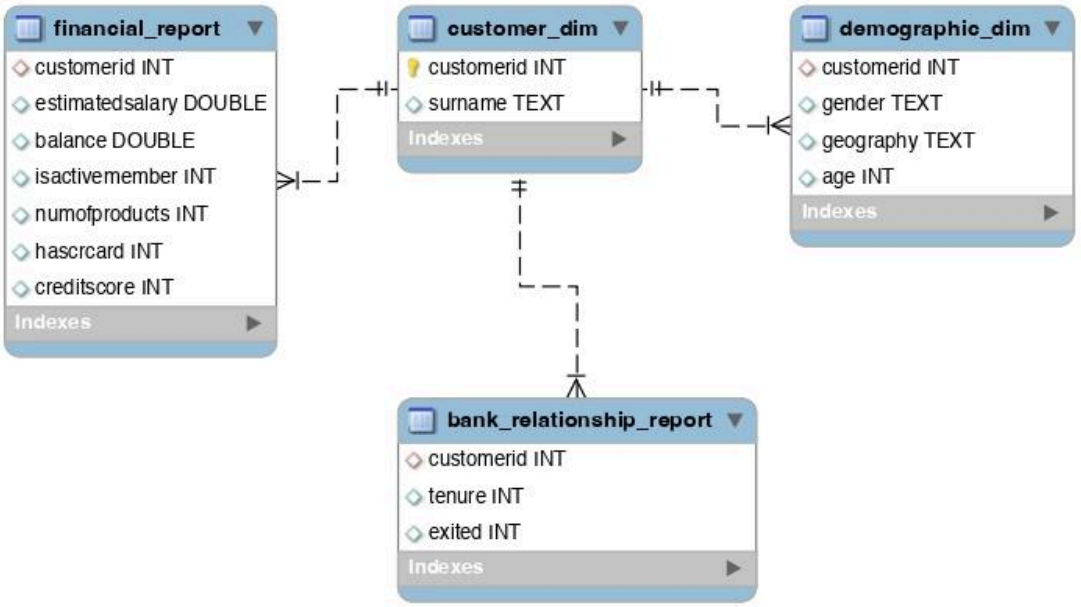
- Import the customer database in a jupyter notebook
- Create 4 tables (as precise by the following informations)
- Export the tables in csv
- Import tables in My SQL Workbench (including type column choosing)
- Define Primary keys and foreigners keys

- Create a model and a diagram

ERD conception based on 4 tables :

- *Customer_dim* (main informations about customer : surname, customer_id - primary key)
- *Financial_report* (informations about customer's activity)
- *Bank_relationship_report* (informations about the relation between customers and bank)
- *demographie_dim* (informations about customers)

Diagram :



---

6. Queries

I have processed the results from the five SQL queries.
All queries have been realised in Mysql Workbench (including keys management).

Query Results Analysis

Query 1: Churn Rate by Geography and Gender

Objective: Calculate the customer churn rate (average 'exited' status) grouped by geography and gender to identify demographic segments with the highest propensity to churn.

geography	gender	churn_rate	total_customers
France	Female	0.20	2727
Spain	Female	0.21	1142
Germany	Female	0.24	1195
France	Male	0.15	2753
Spain	Male	0.13	1297
Germany	Male	0.17	1383

Query 2: Age and Product Count by Churn Status

Objective: Compare the average age and the average number of products held by churned customers versus active customers.

exited	avg_age	avg_num_products	total_customers
0	37	1.54	7963
1	45	1.48	2037

Query 3: Financial Metrics by Active Member Status

Objective: Determine the average estimated salary and average bank balance for customers who are active members versus those who are not.

isactivemember	avg_estimated_salary	avg_balance	total_customers
0	99238	120560	4849
1	100743	117716	5151

Query 4: Balance and Tenure for Customers with Multiple Products

Objective: Identify customers who hold more than one product and show their average balance and tenure, ordered by balance to see if higher balances correlate with product count and tenure.

numofproducts	avg_balance	avg_tenure	total_customers
2	119524	5	5000
3	112261	5	266
4	93088	4.6	42

Query 5: Age Range of Churned Customers

Objective: Find the maximum and minimum age of customers who have churned ('exited' = 1) to understand the age range of customers most likely to leave.

max_age_churned	min_age_churned
84	18

---

7. API Development and Exposing Data via API (Flask)

A. Introduction: Bank Customer Analytics API

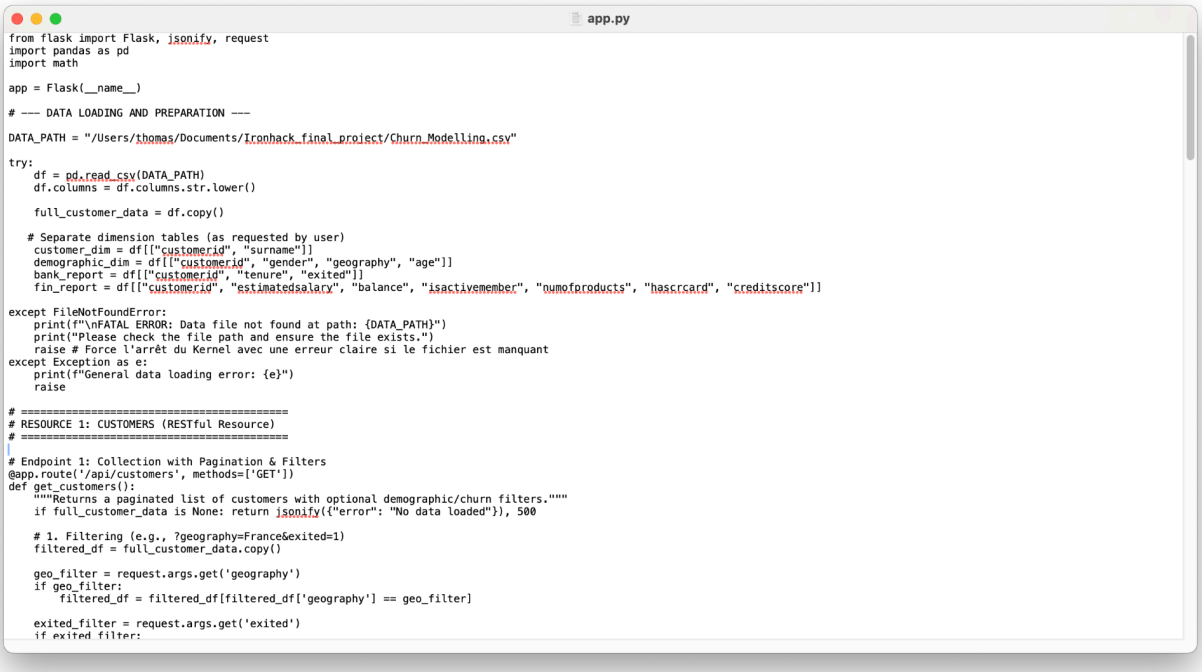
This project presents a RESTful API developed with **Flask** to query and analyze a bank customer dataset (sourced from a churn modeling database). The API provides access to data through paginated and filterable routes, and executes complex analytical queries (**Pandas** aggregations) directly on the server side.

- **Technology:** Python 3.x, Flask, Pandas.
- **Access Point (Base URL):** `http://127.0.0.1:5001/api`

B. Source Code (`app.py`)

This single file implements secure data loading, dimension table separation, and the logic for the four core endpoints.

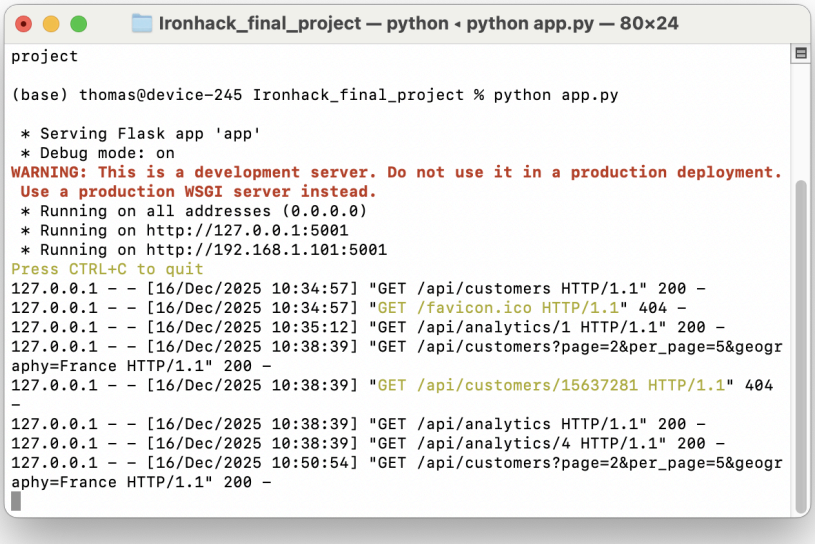
The source code has been executed in the text edit.



All the code has been saved in the jupyter notebook.

C. Server Functionality

The Flask server was launched locally via the terminal, ensuring a stable environment for handling requests.



D. Detailed Endpoint Documentation and Results

Resource 1: Customers (`/customers`)

1. Collection with Pagination and Filters (`GET /api/customers`)

Feature	Details
URL	`/api/customers`
Method	GET
Parameters	`page` (int, default 1), `per_page` (int, default 10), `geography` (str, e.g., `France`), `exited` (0 or 1).
Example	`/api/customers?geography=France&page=2&per_page=5`

Result (Filtering and Pagination Example) *Screenshot of the DataFrame output from your successful TEST A (French customers, page 2).*

```
print( \n  +  - *50 +  \n )

--- TEST A: Clients (Page 2, 5 par page, filtré France) ---
Statut: OK (200). Total clients filtrés: 5014
Premiers résultats de la page 2:
   age    balance  creditscore  customerid  estimatedsalary  exited gender  \
0   27  134603.88          684    15592389         71725.73       0   Male
1   31  102016.72          528    15767821         80181.12       0   Male

  geography  hascrcard  isactivemember  numofproducts  rownumber surname  \
0    France          1               1              1         10      H?
1    France          0               0              2         11  Bearce

   tenure
0       2
1       6
```

2. Single Customer Detail with Nesting (`GET /api/customers/<int:customer_id>`)

Feature	Details
URL	`/api/customers/{customer_id}`
Method	GET
Objective	Manually combines data from dimension tables (`customer_dim`, `demographic_dim`, etc.) to structure the response into nested JSON objects.
Customer id	15592389

```
### 2. UNIQUE CUSTOMER (Nested Details)
Status: OK (200). Details for H?:
  > First Name: N/A
  > Age: 27
  > Balance: 134603.88
  > Churn: False
```

Resource

2: Analytics (`/analytics`)

3. Report List (`GET /api/analytics`)

- **URL:** `/api/analytics`
- **Result:** Returns an index of all 5 available analysis reports (ID, name, and description).

4. Report Execution (`GET /api/analytics/<int:report_id>`)

Feature	Details
URL	`/api/analytics/{report_id}`
Method	GET
Objective	Executes the corresponding Pandas query (e.g., aggregation, mean calculation, filtering) and returns the result.

Result (Example of Query 4: Multi-Product Analysis)

```
========================================
     TESTING THE 4 MAIN ENDPOINTS
========================================

### 1. CUSTOMERS (Filtering and Pagination)
Status: OK (200). Total filtered customers (France): 5014
Current page: 2
First results on the page:
   customerid surname geography  age
0    15592389      H?    France   27
1    15767821  Bearce    France   31
2    15632264     Kay    France   34
3    15691483    Chin    France   25
4    15568982     Hao    France   24

--------------------------------------------------

### 2. SINGLE CUSTOMER (Nested Details)
!!! SINGLE CUSTOMER FAILED !!! Code: 404
{'error': 'Customer not found'}

--------------------------------------------------

### 3. ANALYTICS LIST (List of Reports)
Status: OK (200). Number of available reports: 5
Report #4 Name: Multi-Product Analysis

### 4. ANALYTICS EXECUTION (Query 4 - Multi-Product)
Status: OK (200). Report ID 4 loaded.
Description: Balance and Tenure for customers with >1 product, ordered by balance
Query Result (Avg Balance by number of products):
    avg_balance  avg_tenure  numofproducts  total_customers
0  93733.135000    5.300000              4               60
1  75458.328195    5.003759              3              266
2  51879.145813    5.051852              2             4590

========================================
```

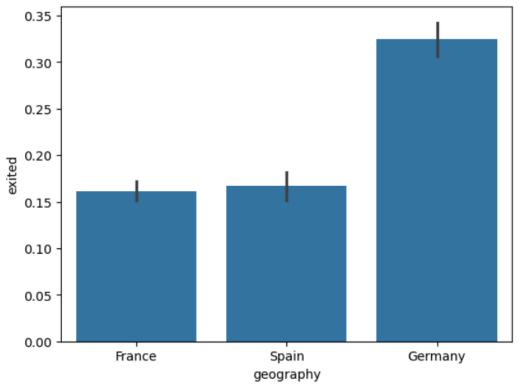
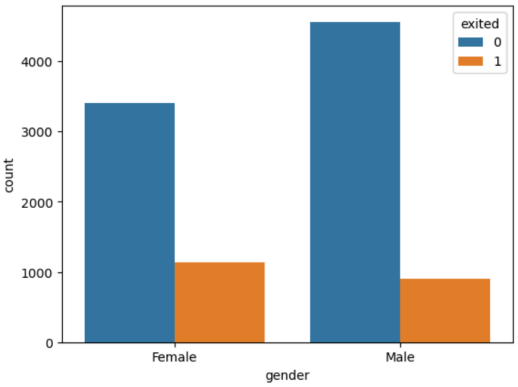
8. Statistical Analysis

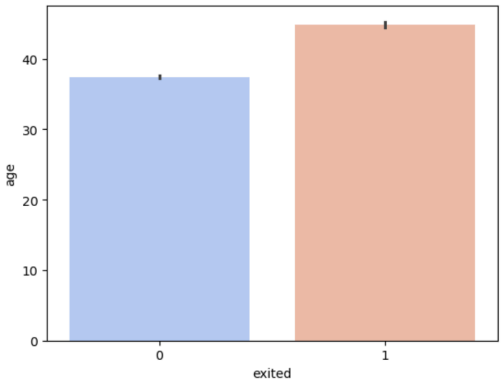
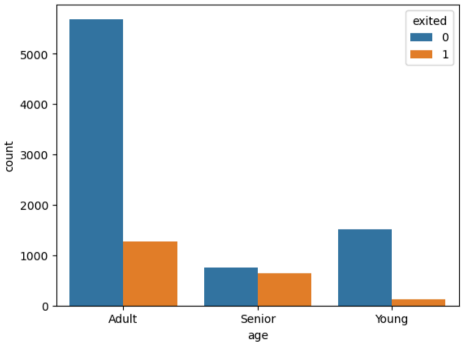
All the statistical analysis has been realised in a jupyter notebook.

A. Customer Identification Variables

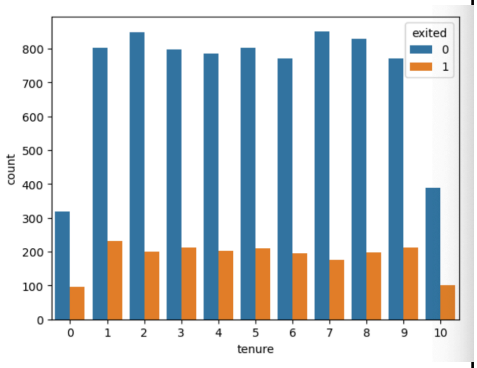
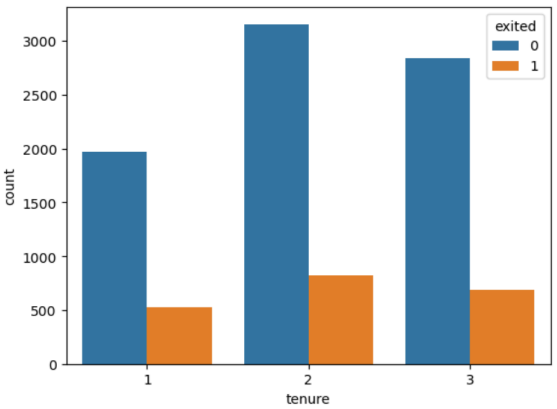
Variable	Visualization Used	Statistical Test Used	Analysis & Action Taken
🔢 **RowNumber**	N/A	N/A	**Removed**: Considered noise/index, not predictive.
🆔 **CustomerId**	N/A	N/A	**Keep :** Unique identifier with no predictive value.
👤 **Surname**	N/A	N/A	**Removed**: High-cardinality categorical variable, not useful for prediction.

B. Demographic variables

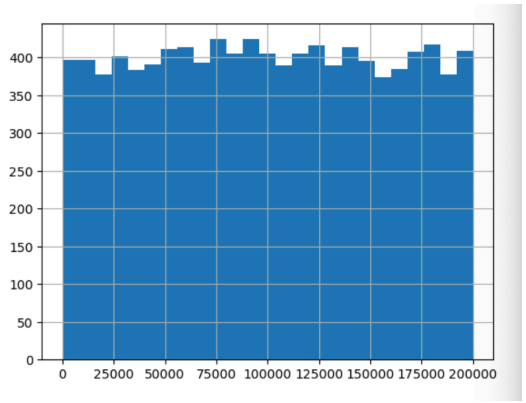
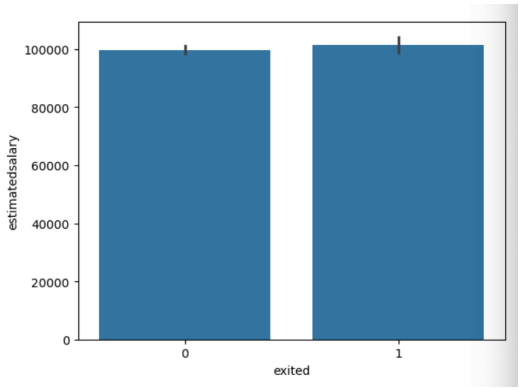
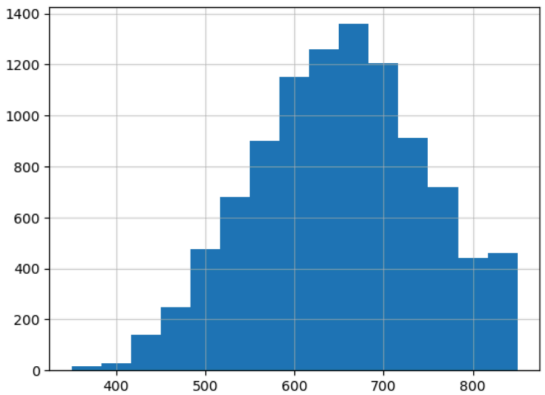
Variable	Visualization Used	Statistical Test Used	Analysis & Action Taken
🌍 **Geography**	sns.countplot sns.barplot 	**Chi-2 Test** (Global Significance) **Proportion Z-test**(France/Spain Comparison)	**Regrouped**: Z-test showed no significant difference in churn between **France** and **Spain**. **Action**: Merged "France" and "Spain", leaving "Germany" separate.
⚧ **Gender**	sns.countplot 	**Chi-2 Test** **Proportion Z-test**	**Kept**: Tests confirmed females churn significantly more than males. **Action**: Retained and Encoded.

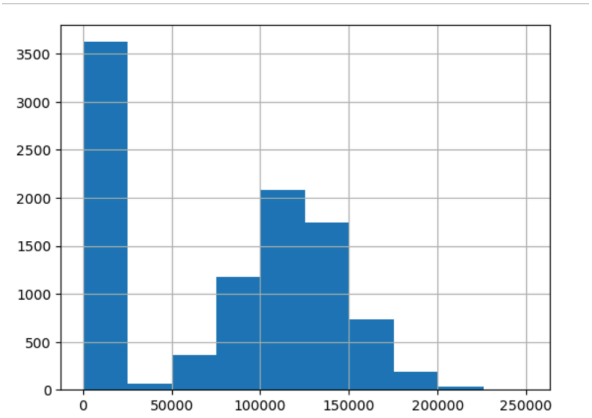
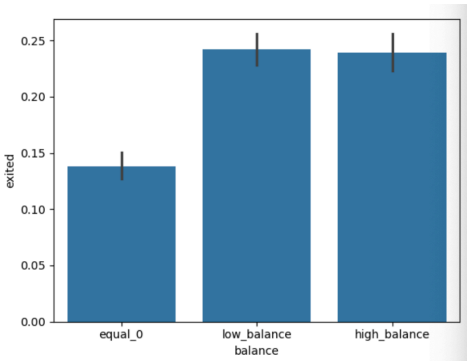
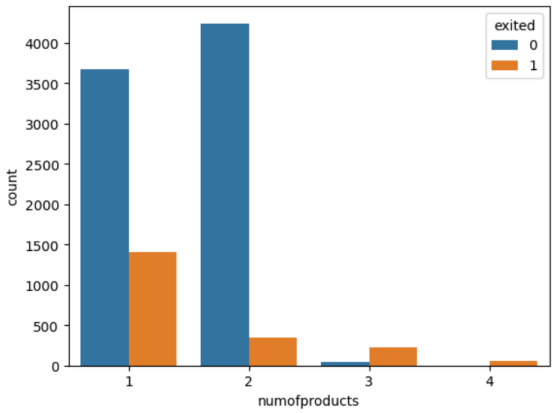
🎂 **Age**	plt.hist & sm.qqplot sns.barplot 	**Shapiro-Wilk Test** (Normality) **T-test** (Difference in Mean) **Proportion Z-test** (Age group comparison)	**Categorized**: T-test confirmed a significant difference. **Action**: Grouped into **bins**: **Young** (<30), **Adult** (30-50), and **Senior** (>50). 

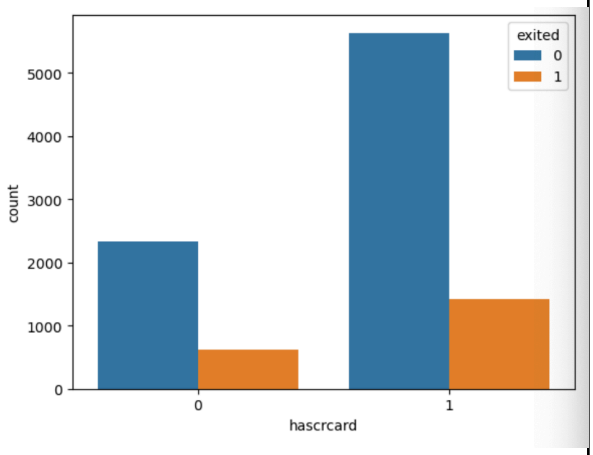
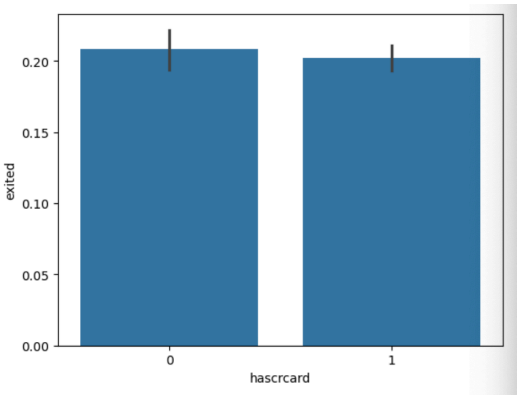
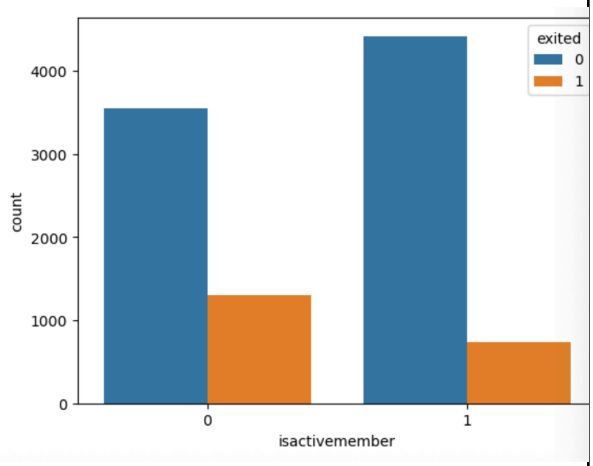
C. Banking Relationship Variables

Variable	Visualization Used	**Statistical Test Used**	Analysis & Action Taken
⏳ **Tenure**	`sns.countplot` 	**Chi-2 Test** **Proportion Z-test**	**Dropped**: Even after a categorisation of tenure, Chi-2 and Z-tests showed no significant relationship with churn. **Action**: Removed from the dataset. 

D. Financial Variables

Variable	Visualization Used	**Statistical Test Used**	Analysis & Action Taken
💼 EstimatedSalary	plt.hist sns.barplot 	**T-Test**	**Dropped**: T-test and categorization showed no difference in churn proportion. **Action**: Removed from the dataset 
💳 CreditScore	plt.hist sns.violinplot 	**Shapiro-Wilk Test** **T-test** (Difference in Mean) **Proportion Z-test** (Score group comparison)	**Transformed**: Z-test showed no difference between "Medium" and "High" scores. **Action**: Grouped into **Binary**: "Low Score" vs "Medium-High Score".

💰 **Balance**	plt.hist & sm.qqplot sns.barplot 	**Shapiro-Wilk Test** **T-test** (Difference in Mean) **Proportion Z-test**(Balance group comparison)	**Transformed**: T-test showed churners have higher balances. Z-test justified merging positive balances. **Action**: Converted to **Binary**: 0 (Zero Balance) vs 1 (Positive Balance). 
📦 **NumOfProducts**	sns.countplot 	**Chi-2 Test** **Proportion Z-test**	**Kept**: Chi-2 showed a strong relationship (high Cramer's V). **Action**: Retained.

💳 **HasCrCard**	sns.countplot sns.barplot 	**Chi-2 Test** **Proportion Z-test**	**Dropped**: Tests confirmed the absence of a significant relationship with churn. **Action**: Removed from the dataset. 
⚡ **IsActiveMember**	sns.countplot 	**Chi-2 Test** **Proportion Z-test**	**Kept**: Z-test confirmed inactive members are ~2x more likely to churn. **Action**: Retained.

❌ Variables removed

- **Tenure**
- **HasCrCard**
- **EstimatedSalary**

---

9. Modelisation

🔧 Preparation

- **Encoding : *LabelEncoder* for XGBoost Modeling**

Introduction to Encoding: Before training a machine learning model, categorical (non-numerical) features must be converted into a numerical format. **Label Encoding** is one method that assigns a unique integer to each category, typically starting from 0. This is suitable for **ordinal variables** (where categories have a meaningful order, like Age groups) and **binary variables** (like Gender) in tree-based models like XGBoost, which are insensitive to the numerical magnitude of the labels.

Variables	Input (Original Category / Group)	Output (Encoded Numerical Value)	Type of Encoding	Justification for XGBoost
⚧ **Gender**	`Female` `Male`	`0` or `1` `1` or `0`	Label Encoding	**XGBoost Justification:** Using **0 and 1 is standard and effective** for binary features in tree-based models like XGBoost. The model interprets these as simple category flags (a split criteria) where the numerical order (1 > 0) is ignored.

🎂 **Age**	`Young`(<30)
	`Adult`(30-50)
	`Senior`(>50)
		`0`
		`1`
		`2`
			Label Encoding	**Ordinal Feature:** This is appropriate because the categories have an inherent order (age progression). Label Encoding maintains this sequence (0 < 1 < 2), which the model can use.

💰 **Balance**	`Zero Balance`(= 0) `Positive Balance`(> 0)	`0` `1`	Binary Encoding (Specific Mapping)	**XGBoost Justification:** Similar to Gender, this binary feature is treated as a simple flag. The encoding directly reflects your EDA finding that zero-balance accounts behave differently from positive-balance accounts.
🌍 **Geography** (After grouping)	`Germany` `France-Spain`	`Geography_Germany: 1, 0` `Geography_France-Spain: 0, 1`	One-Hot Encoding (`get_dummies`)	**Nominal Feature:** While you reduced the number of categories, they remain nominal (no inherent order). One-Hot Encoding is the best practice to prevent the model from assuming that `Germany` (if it had been coded, say, `0`) is less than `France-Spain` (coded `1`).

- **Scaling : *MinMaxScaler***

Put values between 0 and 1 :

- Follow the initial distribution
- Allow us to be sure that there is not a dominant variable because of high values

- **Split 70/30**

With the same churn rate in each sample.

🧠 Model used

- **XGBoost**
  - **Why ?** Good model for tables prediction (including bank database)
  - **boost model :**
    - Learn by iteration about previous prediction (learn form him mistake)
    - Allow quickly to get prediction with quality
  - **Result :**
    - Accuracy : **~0.76** = don't trust it because with unbalanced dataset, accuracy can by hight and can only predict customers remaining
    - F1-score : **~0.56** = show that génèrates a lot of false positive
    - Precision : **~0.48** = I does not matter because it does not cost the bank to find false positive
    - Recall : **~0.71** = show that found a good part of true positive churner
      - **Why is Recall the most important ?**
        Our goal is to find the largest number of churn customers.
        Recall take the proportion of True positives that we found divided by all the positives.

$$\text{Recall} = \frac{\text{True Positives } (TP)}{\text{True Positives } (TP) + \text{False Negatives } (FN)}$$

In that case Recall is the most important because it costs a lot to the bank to miss churn customers.

  - **Model Management :**
    - Dataset unbalanced.
    - **Smote :**
      - Trying to give more importance to churn customers (purpose is to find them)
    - **Threshold :**
      - Using different threshold to give more importance to churn customers

I used few threshold different above than 0,5 and the one who give the best recall is 0,6

Why not other models?

- **Logistic Regression:** Too linear → misses interactions like Age × Balance × Products.
- **SVM (Support Vector Machine):** Too slow, not easily interpretable, and less suited for large volumes.
- **Random Forest:** Solid, but less performant and less optimizable than XGBoost."

---

10. Model limits

Even with some good results, XGBoost has some limits :

- **2052 fails** (20,52 %)
- **560 False Negative** → worst scenario
- **1492 False Postive** → tolerable but expensive

👉 Model miss to much churn
👉 A improve simple improve is necessary

---

11. Own Scoring Creation

🎛️ **Objective :** To build a business score (a custom feature) that strengthens the model and better distinguishes between churn and non-churn profiles.

⚙️ Method For the variables used in each scoring test :

1. Using sample
2. Choosing the dataset
3. Using **Principle of Impact Encoding :**

   The objective is to replace each variable with the **mean probability of the target variable** for that variable.

   📝 **Formula and Method** for a given category in a variable (C):

$$\text{Impact Score} = \text{Mean of the target for variable } C$$

If your target is binary (0 or 1, such as Churn/Non-Churn), the Impact Score is simply the **proportion of class 1 (the churn) within that variable**.

4. Sum the scored variables for each customer.
5. Normalization 0→1.
6. Classification into 4 risk levels :

Score Range	Risk Level
0.0−0.25	Very Low
0.25−0.5	Moderate
0.5−0.7	⚠️ High
>0.7	⚠️⚠️ Very High

👉 The higher the score → the more likely the customer is to churn.

5. Calculate the distribution of churn (1) and non-churn (0) customers within each risk level, then compute the percentage of each group in each category.

➡️ The goal is to observe that churned customers (1) have a high average score (+0.5) and non-churned customers (0) have a low average score (−0.5)

---

12. Scoring Tests

📍 Test 1

- **Dataset:** Same dataset as for the model (including several categorized variables).
- **Value:** Churn proportion for the variable (20% of those with a credit score of 0 churn, thus row value = 0.2).

- **Result:** Score is too flat → insufficient spread/differentiation.
  → Probably because of a problem in my categorisation/encoding

📍 ⭐ Test 2 – Final Score Retained

- **Dataset:** Unprepared dataset but using only variables with a strong relationship to churn.
- **Value:** Churn Proportion.
- **Result:**
  - 97% of non-churned customers <0.25
  - 0% of non-churned customers >0.5
  - 13% of churned customers <0.25
  - 69% of churned customers >0.5
- **Conclusion:** Everything above 0.5 is churned.

📍 Test 3

- **Dataset:** Unprepared dataset including **all** variables.
- **Value:** Churn Proportion.
- **Result:**
  - 99% of non-churned customers <0.25
  - 1% of non-churned customers >0.25
  - 18% of churned customers <0.25
  - 82% of churned customers >0.25
- **Question:** Is this more satisfactory, given that we have more churned customers <0.25, but we can almost conclude that 100% of those with a score >0.25 are churned?

---

13. Combining Scoring and Model

Objective: Reduce False Negatives without drastically increasing False Positives.

⭐ Combining Test 1 – The Best Strategy

- Rule:
  - If score ≥ 0.3→ predict churn
  - Otherwise → use model's prediction

📊 Results

- Total Errors: **1654** (vs 2050 in the original model)
- Missed Churn (False Negatives): **195** (vs 560) = 1% of all churn cases (195/2057)
- → **365 churn cases recovered**

- → **Significantly more performant than the model alone.**

Other Combinations

- Test 2: More precise, but loses too many churn cases (higher FN).
- Test 3: Too many False Positives (FP).
- Test 4: Inconclusive.

👉 **The best compromise is clearly Combining Test 1.**

📌 Conclusion

The establishment of a comprehensive pipeline has allowed for the identification of key churn drivers, the modeling of predictions, and a significant improvement in results through an internal scoring mechanism.
The final approach avoids searching complex parameters in the model and uses simple methods to maximize churn detection.
But in fact, the retained hybrid approach is a pragmatic and high-performing solution for the project environment. However, it sacrifices some of the simplicity and methodological elegance of a purely algorithmic pipeline in favor of an immediate performance gain on the critical metric.

📌 Challenges and next steps

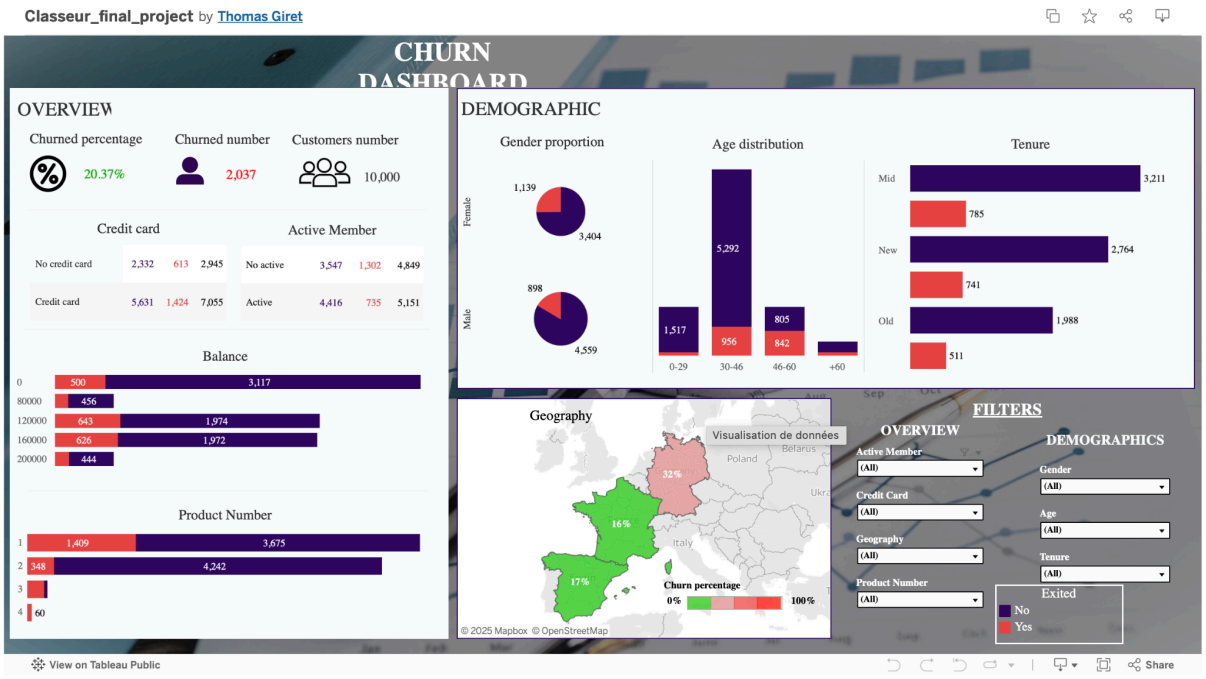
- **Multivariate analysis**
- **Trying to find the best customers prediction with only one model or own scoring**

📕 Appendix

Library used

Library	Alias
pandas	pd
numpy	np
matplotlib.pyplot	plt
seaborn	sns
statsmodels.api	sm
scipy.stats	stats
statsmodels.stats.proportion	
math	
os	

Project Documentation: Churn Prediction Dashboard



[Dashboard link]

1. Overview

This dashboard provides a comprehensive visual analysis of customer retention for a base of **10,000 customers**. It is designed to identify key patterns and risk factors associated with customer churn (referred to as "Exited" in the data).

Key Performance Indicators (KPIs)

- **Total Customers:** 10,000
- **Churned Number:** 2,037
- **Churn Percentage:** 20.37%

2. Dashboard Sections & Visualizations

A. Overview & Account Details

- **Status Comparison:** Breaks down the population by **Credit Card** ownership and **Active Member** status.
    - *Observation:* It allows you to see if being an active member significantly reduces the likelihood of exiting.
- **Financial Metrics (Balance):** Categorizes customers into balance brackets (0 to 200,000+).

- **Product Usage:** A bar chart showing how many products a customer holds.
    - *Utility:* This helps identify if customers with only one product are more likely to churn than those with multiple products.

B. Demographic Profiles

- **Gender Proportion:** Pie charts for Male and Female segments, highlighting the volume of churn within each gender.
- **Age Distribution:** Grouped bar charts (0-29, 30-45, 46-60, +60).
    - *Insight:* This identifies specific age groups (e.g., 46-60) that might show higher exit rates.
- **Tenure:** Segments customers into "New," "Mid," and "Old" to analyze if loyalty over time impacts retention.

C. Geographic Analysis

- **Interactive Map:** Displays churn percentages by country (France, Germany, Spain).
    - *Critical Finding:* **Germany** shows a significantly higher churn rate (**32%**) compared to France (16%) and Spain (17%), indicating a need for localized business strategies.

3. Interactive Functionalities

- **Global Cross-Filtering:** By clicking on any visual element (e.g., a specific age group or a country on the map), the entire dashboard updates to show data specifically for that segment.
- **Multi-Attribute Filters:** A dedicated "FILTERS" panel allows users to drill down by:
    - **Demographics:** Gender, Age, Tenure.
    - **Account Info:** Active Member status, Credit Card ownership, Geography, and Product Number.
- **Color-Coded Logic:** Consistent use of color (Purple for "No" Exit, Red for "Yes" Exit) provides immediate visual recognition of churn density across all charts.

4. Strategic Decision-Making & Self-Service Capability

- **Marketing Empowerment:** The dashboard is specifically structured to streamline **decision-making** for marketing teams. By segregating behavioral data from demographic profiles, users can instantly identify which customer segments require urgent retention campaigns.
- **Self-Service Interface:** The inclusion of the **filter panel on the right** transforms this into a high-value **"self-service" tool** for business users. It allows non-technical stakeholders to explore the data independently and answer specific business questions without requiring further technical assistance.