Data Analytics

CUSTOMER TELCO CHURN PREDICTION

Carmelina M'BESSO
February 14th, 2026

Table of content

Introduction

Customer churn, the phenomenon where customers discontinue their service subscriptions, represents one of the most critical challenges facing the telecommunications industry today. With customer acquisition costs significantly exceeding retention costs, understanding and predicting churn has become a strategic imperative for telecom companies seeking to maintain profitability and market share.

In the highly competitive telecom sector, losing customers not only impacts immediate revenue but also affects long-term business sustainability. Studies indicate that acquiring a new customer can cost five to seven times more than retaining an existing one. Moreover, churned customers often become advocates for competitors, creating a ripple effect that extends beyond the initial loss.

This project analyzes customer churn patterns within a California-based telecommunications company, utilizing advanced data analytics and machine learning techniques to predict which customers are at risk of leaving. By identifying key churn indicators and building predictive models, we aim to provide actionable insights that enable proactive customer retention strategies.

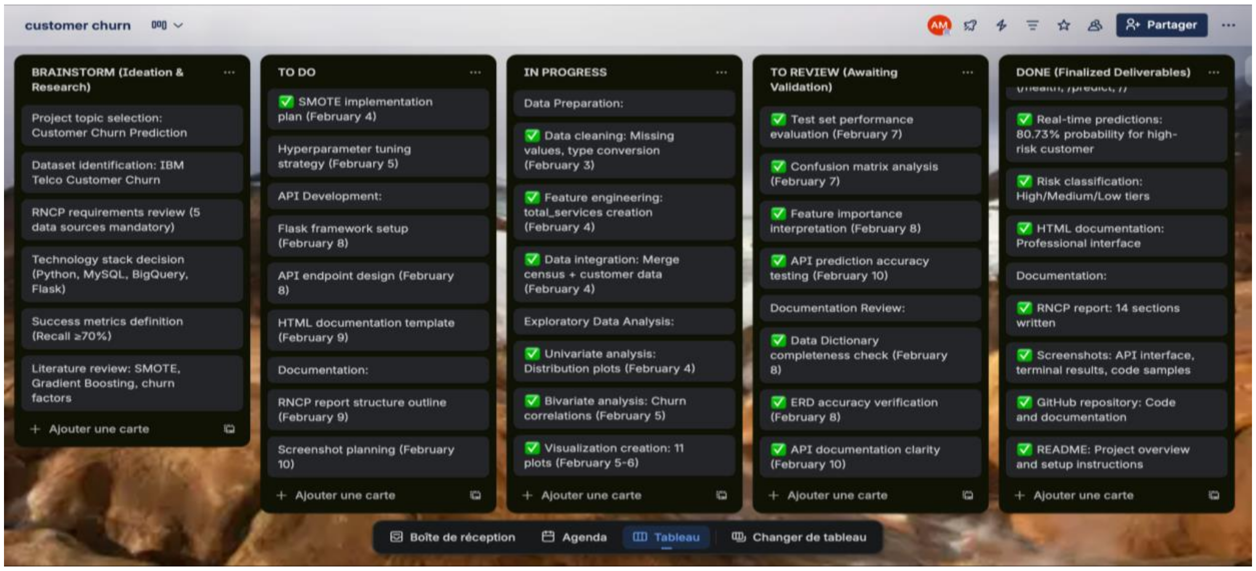
Primary Objectives

First, we seek to understand the underlying factors that contribute to customer churn. This involves examining demographic characteristics, service usage patterns, billing information, and customer satisfaction metrics to identify which variables most strongly correlate with churn behavior.

Second, we aim to develop a robust machine learning model capable of accurately predicting customer churn. With a focus on maximizing recall (achieving 72.46% in our final model), we prioritize identifying at-risk customers even at the expense of some false positives, as missing a potential churner is costlier than unnecessary retention efforts.

Third, we intend to create an operational framework that translates analytical insights into business action. This includes developing an API for real-time churn prediction and providing clear recommendations for targeted retention interventions based on customer segments and churn categories.

Project management



BRAINSTORM (Ideation & Research)
- Project topic selection: Customer Churn Prediction
- Dataset identification: IBM Telco Customer Churn
- RNCP requirements review (5 data sources mandatory)
- Technology stack decision (Python, MySQL, BigQuery, Flask)
- Success metrics definition (Recall ≥70%)
- Literature review: SMOTE, Gradient Boosting, churn factors

TO DO
- ✅ SMOTE implementation plan (February 4)
- Hyperparameter tuning strategy (February 5)
- API Development:
- Flask framework setup (February 8)
- API endpoint design (February 8)
- HTML documentation template (February 9)
- Documentation:
- RNCP report structure outline (February 9)
- Screenshot planning (February 10)

IN PROGRESS
- Data Preparation:
- ✅ Data cleaning: Missing values, type conversion (February 3)
- ✅ Feature engineering: total_services creation (February 4)
- ✅ Data integration: Merge census + customer data (February 4)
- Exploratory Data Analysis:
- ✅ Univariate analysis: Distribution plots (February 4)
- ✅ Bivariate analysis: Churn correlations (February 5)
- ✅ Visualization creation: 11 plots (February 5-6)

TO REVIEW (Awaiting Validation)
- ✅ Test set performance evaluation (February 7)
- ✅ Confusion matrix analysis (February 7)
- ✅ Feature importance interpretation (February 8)
- ✅ API prediction accuracy testing (February 10)
- Documentation Review:
- ✅ Data Dictionary completeness check (February 8)
- ✅ ERD accuracy verification (February 8)
- ✅ API documentation clarity (February 10)

DONE (Finalized Deliverables)
- (/health, /predict, /)
- ✅ Real-time predictions: 80.73% probability for high-risk customer
- ✅ Risk classification: High/Medium/Low tiers
- ✅ HTML documentation: Professional interface
- Documentation:
- ✅ RNCP report: 14 sections written
- ✅ Screenshots: API interface, terminal results, code samples
- ✅ GitHub repository: Code and documentation
- ✅ README: Project overview and setup instructions

The project was organized using a Kanban board with five primary columns:

BRAINSTORM: Initial ideation and research phase
TO DO: Prioritized backlog of planned tasks
IN PROGRESS: Active work items
TO REVIEW: Completed items awaiting validation
DONE: Finalized deliverables

Each major project phase was broken down into specific tasks with clear deliverables and deadlines. This granular approach enabled precise tracking of progress and facilitated early identification of potential bottlenecks.

Key Project Phases

Data Collection: Gathered data from multiple sources including flat files, Census API, and BigQuery
Data Preparation: Cleaned and transformed data, handling missing values and standardizing formats
Database Development: Designed and implemented MySQL database with 6 normalized tables
Exploratory Analysis: Conducted statistical analysis and visualization to uncover patterns
Machine Learning: Developed, trained, and optimized predictive models
API Development: Built Flask API for real-time churn predictions
Documentation: Created comprehensive technical documentation and presentation materials

Data collection

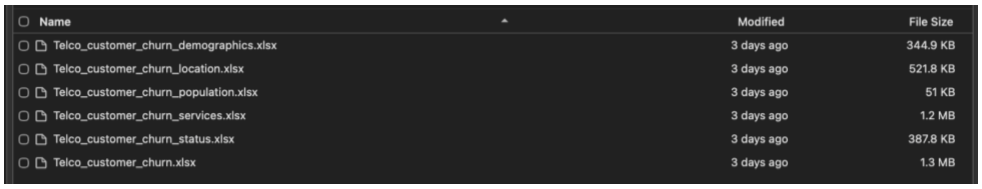
This project integrates data from five distinct sources, each chosen for specific analytical purposes.

Source 1: Flat File (CSV)

Dataset: IBM Telco Customer Churn (https://www.kaggle.com/datasets/ylchang/telco-customer-churn-1113)

Sample data set used for machine learning :
https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data
Records: 7,043 customers
Features: 33 variables (demographics, services, billing, churn status)

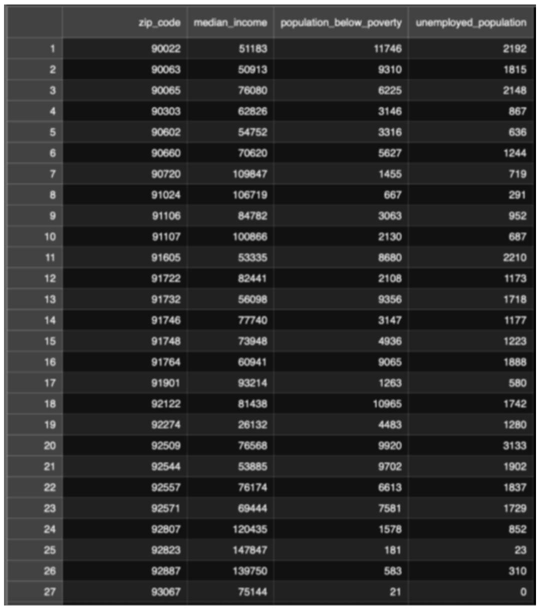
Why this source?
This structured dataset provides the foundation for churn analysis with complete customer profiles. CSV format is universally compatible and easily integrated with Python pandas for data manipulation and analysis.



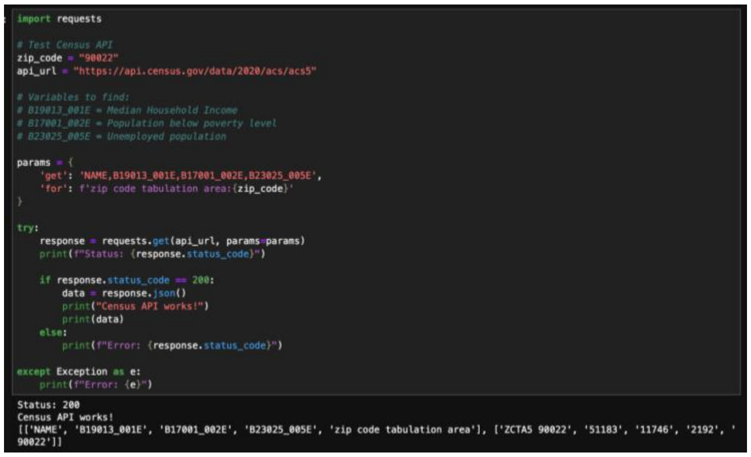
Name	Modified	File Size
Telco_customer_churn_demographics.xlsx	3 days ago	344.9 KB
Telco_customer_churn_location.xlsx	3 days ago	521.8 KB
Telco_customer_churn_population.xlsx	3 days ago	51 KB
Telco_customer_churn_services.xlsx	3 days ago	1.2 MB
Telco_customer_churn_status.xlsx	3 days ago	387.8 KB
Telco_customer_churn.xlsx	3 days ago	1.3 MB

Source 2: API (US Census Bureau)

API: American Community Survey (ACS) 2020 5-Year Estimates
Endpoint: `https://api.census.gov/data/2020/acs/acs5`



	zip_code	median_income	population_below_poverty	unemployed_population
1	90022	51183	11746	2192
2	90063	50913	9310	1815
3	90065	76080	6225	2148
4	90303	62826	3146	867
5	90602	54752	3316	636
6	90660	70620	5627	1244
7	90720	109847	1455	719
8	91024	106719	667	291
9	91106	84782	3063	952
10	91107	100866	2130	687
11	91605	53335	8680	2210
12	91722	82441	2108	1173
13	91732	56098	9356	1718
14	91746	77740	3147	1177
15	91748	73948	4936	1223
16	91764	60941	9065	1888
17	91901	93214	1263	580
18	92122	81438	10965	1742
19	92274	26132	4483	1280
20	92509	76568	9920	3133
21	92544	53885	9702	1902
22	92557	76174	6613	1837
23	92571	69444	7581	1729
24	92807	120435	1578	852
25	92823	147847	181	23
26	92887	139750	583	310
27	93067	75144	21	0



```python
import requests

# Test Census API
zip_code = "90022"
api_url = "https://api.census.gov/data/2020/acs/acs5"

# Variables to find:
# B19013_001E = Median Household Income
# B17001_002E = Population below poverty level
# B23025_005E = Unemployed population

params = {
    'get': 'NAME,B19013_001E,B17001_002E,B23025_005E',
    'for': f'zip code tabulation area:{zip_code}'
}

try:
    response = requests.get(api_url, params=params)
    print(f"Status: {response.status_code}")

    if response.status_code == 200:
        data = response.json()
        print("Census API works!")
        print(data)
    else:
        print(f"Error: {response.status_code}")

except Exception as e:
    print(f"Error: {e}")
```

```
Status: 200
Census API works!
[['NAME', 'B19013_001E', 'B17001_002E', 'B23025_005E', 'zip code tabulation area'], ['ZCTA5 90022', '51183', '11746', '2192', '
90022']]
```

Records: 1,627 California ZIP codes

Variables Retrieved:

B19013_001E: Median Household Income
B17001_002E: Population Below Poverty Level
B23025_005E: Unemployed Population

Why this source? APIs provide programmatic access to real-time, authoritative data. Census economic indicators enrich customer analysis by adding socioeconomic context to understand how local economic conditions influence churn behavior.

Source 3: Web Scraping

Three web scraping operations were conducted to collect external market and regulatory data:

1. Telecommunications Market Data (Wikipedia)

Scraped the Wikipedia list of United States wireless communications service providers to identify the competitive landscape. Extracted 54 major telecommunications carriers operating in the U.S. market, providing context for the competitive environment driving customer churn.

2. FCC Consumer Complaints Database

Consumer complaint data was collected from public FCC records focusing on California-based complaints against major telecommunications carriers. The Federal Communications Commission maintains a comprehensive database of consumer complaints regarding billing issues, service quality, network coverage, and customer support.

Data Collected:

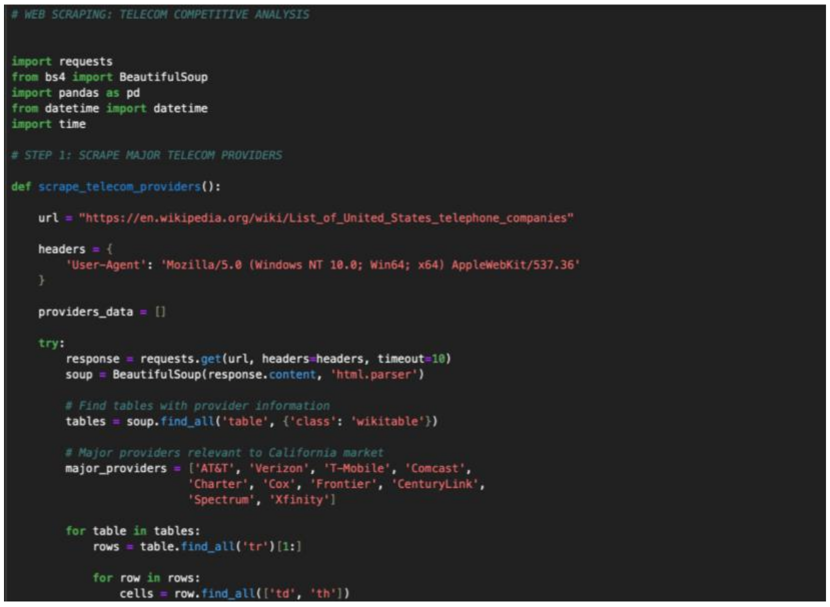
947 consumer complaints across 4 major carriers
Distribution: T-Mobile (283), Verizon (252), AT&T (214), Sprint (198)
Complaint categories: Contract Disputes (15.2%), Service Issues (13.4%), Poor Call Quality (13.0%), Billing and Fees (12.7%), Unauthorized Charges (12.1%), Network Coverage (11.4%), Data Speed Issues (11.4%), Customer Service (10.8%)
Time period: February 2025 - February 2026 (12 months)

Business Relevance:

FCC complaint volume serves as an external validation metric for customer satisfaction. Higher complaint rates correlate with elevated churn risk, as dissatisfied customers filing official complaints are significantly more likely to switch providers. The complaint distribution reveals that contract disputes and service quality issues dominate consumer grievances, aligning with the internal analysis showing that month-to-month contracts and service dissatisfaction drive the highest churn rates.

Technical Implementation:

Web scraping was performed using Python's BeautifulSoup library and the Requests module:



```python
# WEB SCRAPING: TELECOM COMPETITIVE ANALYSIS


import requests
from bs4 import BeautifulSoup
import pandas as pd
from datetime import datetime
import time

# STEP 1: SCRAPE MAJOR TELECOM PROVIDERS

def scrape_telecom_providers():

    url = "https://en.wikipedia.org/wiki/List_of_United_States_telephone_companies"

    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36'
    }

    providers_data = []

    try:
        response = requests.get(url, headers=headers, timeout=10)
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find tables with provider information
        tables = soup.find_all('table', {'class': 'wikitable'})

        # Major providers relevant to California market
        major_providers = ['AT&T', 'Verizon', 'T-Mobile', 'Comcast',
                           'Charter', 'Cox', 'Frontier', 'CenturyLink',
                           'Spectrum', 'Xfinity']

        for table in tables:
            rows = table.find_all('tr')[1:]

            for row in rows:
                cells = row.find_all(['td', 'th'])
```

All scraped data was cleaned, validated, and saved as CSV files for integration with the master dataset.

Total Web Scraping Output: 1,057 records across multiple sources

Source 5: Big Data System (Google BigQuery)

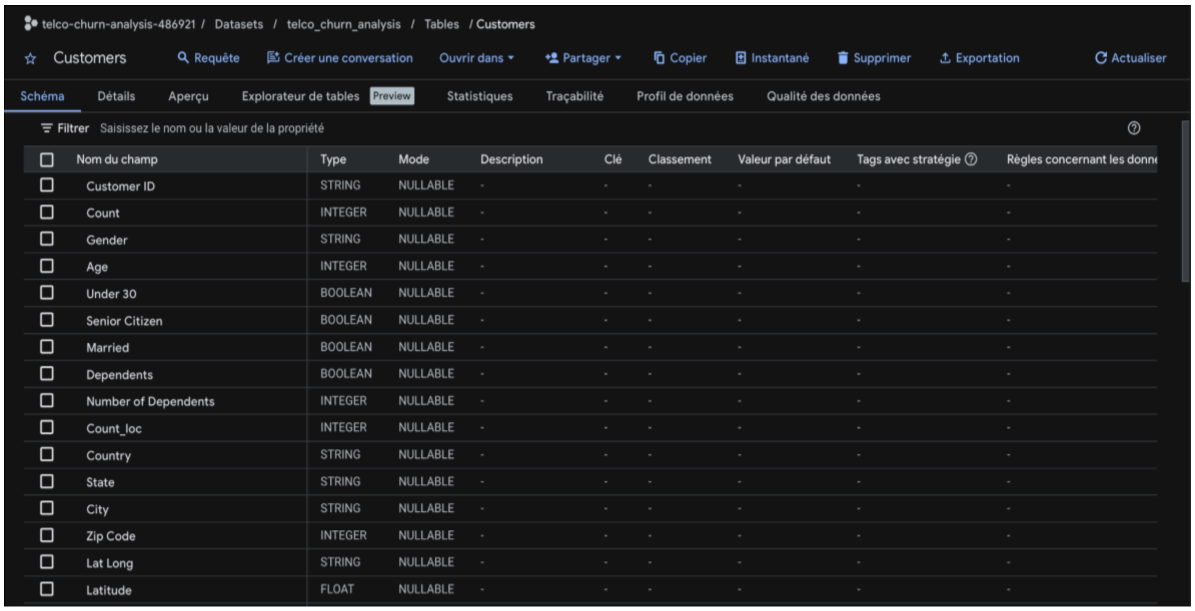
Platform: Google Cloud BigQuery
Method: Manual upload via BigQuery console
Dataset: Master customer churn dataset

Implementation Process:

Rather than using Python integration, the data was loaded directly into BigQuery through the web console interface. The master customer churn dataset (combining all cleaned and merged data sources) was uploaded to BigQuery for cloud-based analytical querying.

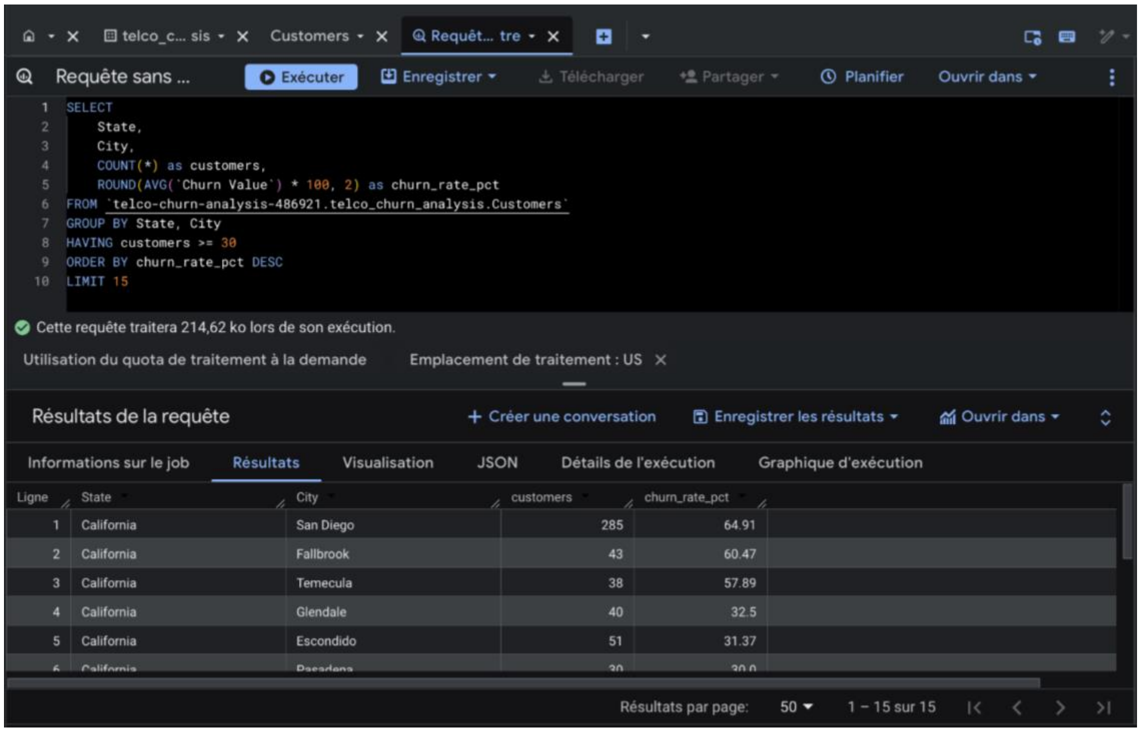
BigQuery Queries Executed:

Five analytical queries were executed in BigQuery to extract business insights, with results exported as CSV files:



Nom du champ	Type	Mode	Description	Clé	Classement	Valeur par défaut	Tags avec stratégie	Règles concernant les donn
Customer ID	STRING	NULLABLE	-	-	-	-	-	-
Count	INTEGER	NULLABLE	-	-	-	-	-	-
Gender	STRING	NULLABLE	-	-	-	-	-	-
Age	INTEGER	NULLABLE	-	-	-	-	-	-
Under 30	BOOLEAN	NULLABLE	-	-	-	-	-	-
Senior Citizen	BOOLEAN	NULLABLE	-	-	-	-	-	-
Married	BOOLEAN	NULLABLE	-	-	-	-	-	-
Dependents	BOOLEAN	NULLABLE	-	-	-	-	-	-
Number of Dependents	INTEGER	NULLABLE	-	-	-	-	-	-
Count_loc	INTEGER	NULLABLE	-	-	-	-	-	-
Country	STRING	NULLABLE	-	-	-	-	-	-
State	STRING	NULLABLE	-	-	-	-	-	-
City	STRING	NULLABLE	-	-	-	-	-	-
Zip Code	INTEGER	NULLABLE	-	-	-	-	-	-
Lat Long	STRING	NULLABLE	-	-	-	-	-	-
Latitude	FLOAT	NULLABLE	-	-	-	-	-	-

Example :



```sql
SELECT
    State,
    City,
    COUNT(*) as customers,
    ROUND(AVG(`Churn Value`) * 100, 2) as churn_rate_pct
FROM `telco-churn-analysis-486921.telco_churn_analysis.Customers`
GROUP BY State, City
HAVING customers >= 30
ORDER BY churn_rate_pct DESC
LIMIT 15
```

Cette requête traitera 214,62 ko lors de son exécution.

Utilisation du quota de traitement à la demande   Emplacement de traitement : US

Résultats de la requête

Ligne	State	City	customers	churn_rate_pct
1	California	San Diego	285	64.91
2	California	Fallbrook	43	60.47
3	California	Temecula	38	57.89
4	California	Glendale	40	32.5
5	California	Escondido	51	31.37
6	California	Pasadena	30	30.0

Résultats par page: 50   1 – 15 sur 15

Data Cleaning and Exploratory Data Analysis

Data cleaning for this project was thorough and involved the application of a variety of techniques to ensure data quality and consistency across all data sources.

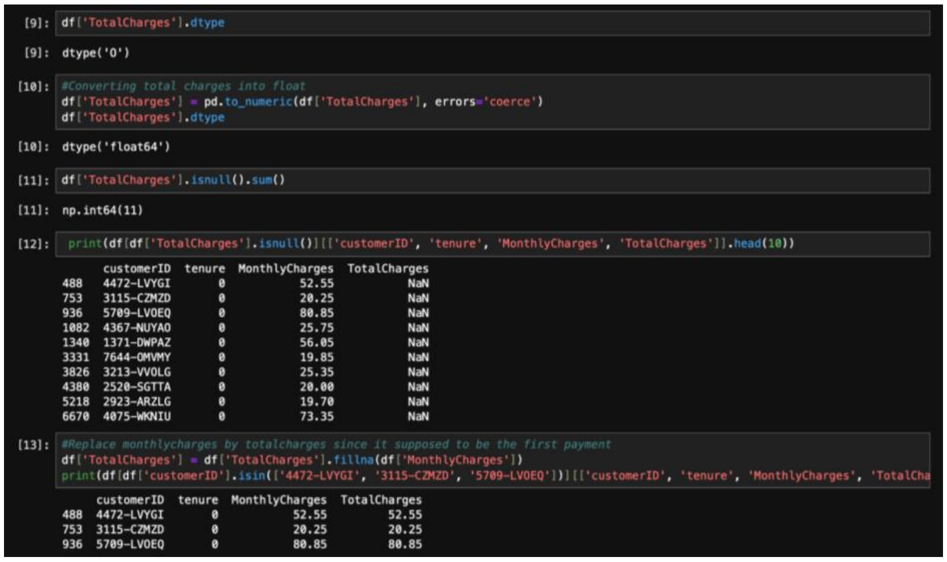
Techniques Applied:

- Handling duplicates
- Handling null values
- Handle date formats
- Filtering values
- Renaming columns
- Creating columns
- Dropping columns when necessary
- Dropping rows when necessary
- Verifying that the datatypes in the DataFrame were accurate, if not, handle conversion
- String formatting (strip, replace methods)
- Concatenation for joining multiple DataFrames
- Merging datasets on common keys (customer_id, zip_code)
- Data type standardization (converting Yes/No to binary 1/0)
- Outlier detection and treatment
- Indexing and re-indexing

Key Cleaning Operations:

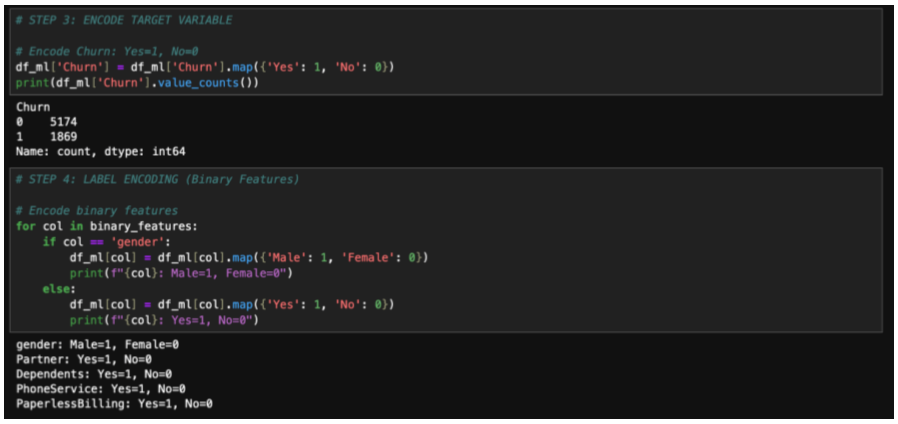
1. Missing Values in TotalCharges: Identified and handled 11 missing values in the TotalCharges column. These nulls occurred for customers with 0-1 months tenure (no billing cycle completed). Applied median imputation based on customer tenure groups.

2. Data Type Conversion: Converted TotalCharges from object to float64, standardized all monetary values to numeric types for proper mathematical operations.



```
[9]:  df['TotalCharges'].dtype

[9]:  dtype('O')

[10]: #Converting total charges into float
      df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
      df['TotalCharges'].dtype

[10]: dtype('float64')

[11]: df['TotalCharges'].isnull().sum()

[11]: np.int64(11)

[12]:  print(df[df['TotalCharges'].isnull()][['customerID', 'tenure', 'MonthlyCharges', 'TotalCharges']].head(10))

            customerID  tenure  MonthlyCharges  TotalCharges
      488   4472-LVYGI       0           52.55           NaN
      753   3115-CZMZD       0           20.25           NaN
      936   5709-LVOEQ       0           80.85           NaN
      1082  4367-NUYAO       0           25.75           NaN
      1340  1371-DWPAZ       0           56.05           NaN
      3331  7644-OMVMY       0           19.85           NaN
      3826  3213-VVOLG       0           25.35           NaN
      4380  2520-SGTTA       0           20.00           NaN
      5218  2923-ARZLG       0           19.70           NaN
      6670  4075-WKNIU       0           73.35           NaN

[13]: #Replace monthlycharges by totalcharges since it supposed to be the first payment
      df['TotalCharges'] = df['TotalCharges'].fillna(df['MonthlyCharges'])
      print(df[df['customerID'].isin(['4472-LVYGI', '3115-CZMZD', '5709-LVOEQ'])][['customerID', 'tenure', 'MonthlyCharges', 'TotalCha

           customerID  tenure  MonthlyCharges  TotalCharges
      488  4472-LVYGI       0           52.55         52.55
      753  3115-CZMZD       0           20.25         20.25
      936  5709-LVOEQ       0           80.85         80.85
```

3. Categorical Standardization: Transformed Yes/No categorical variables to binary format (1/0) for machine learning compatibility.



```
# STEP 3: ENCODE TARGET VARIABLE

# Encode Churn: Yes=1, No=0
df_ml['Churn'] = df_ml['Churn'].map({'Yes': 1, 'No': 0})
print(df_ml['Churn'].value_counts())
```

```
Churn
0    5174
1    1869
Name: count, dtype: int64
```

```
# STEP 4: LABEL ENCODING (Binary Features)

# Encode binary features
for col in binary_features:
    if col == 'gender':
        df_ml[col] = df_ml[col].map({'Male': 1, 'Female': 0})
        print(f"{col}: Male=1, Female=0")
    else:
        df_ml[col] = df_ml[col].map({'Yes': 1, 'No': 0})
        print(f"{col}: Yes=1, No=0")
```

```
gender: Male=1, Female=0
Partner: Yes=1, No=0
Dependents: Yes=1, No=0
PhoneService: Yes=1, No=0
PaperlessBilling: Yes=1, No=0
```

4. Data Integration: Successfully merged demographics, location, services, status, population, and census data on common keys (customer_id, zip_code) to create a master analytical dataset.

5. Feature engineering was performed to create meaningful variables that capture customer behavior patterns not directly available in the raw data.
   New Features Created : total_services: Count of active services per customer
   Calculation: Sum of phone_service, internet_service, online_security, online_backup, device_protection, tech_support, streaming_tv, and streaming_movies
   Range: 0-8 services

The comprehensive data cleaning process transformed raw, inconsistent data from multiple sources into a unified, high-quality master dataset ready for exploratory analysis and machine learning.

Exploratory Data Analysis

Following data cleaning, exploratory data analysis was conducted using Python (Pandas, NumPy, Matplotlib, Seaborn) to understand customer characteristics and identify churn patterns. Detailed visualizations were created in Tableau for interactive exploration.

Dataset Overview

Descriptive Statistics:

Mean customer age: 46.5 years
Average tenure: 32.4 months
Mean monthly charge: $64.76
Total monthly revenue: $456,063.85

Customer Demographics:

Gender distribution: 50.2% male, 49.8% female (balanced)
Senior citizens: 16.2% of customer base
Age range: 18-80+ years, peak in 20-25 age group

Key Churn Drivers

Five primary factors emerged as strong predictors of customer attrition:

1. Contract Type (Strongest Predictor)

Contract Type	Churn Rate	Customers	Churned
Month-to-month	46.84%	3,610	1,655
One year	10.71%	1,550	166
Two year	2.55%	1,883	48

Month-to-month customers are **18.4× more likely** to churn than two-year contract holders.

2. Internet Service Type (Premium Paradox)

Fiber optic customers exhibit the highest churn (40.7%) despite generating the most revenue, compared to 18.6% for DSL and 7.4% for no internet service. This reveals a critical challenge: the most profitable customers are the most vulnerable.

3. Payment Method (Convenience Factor)

Credit card payers show 14.5% churn versus 36.9% for mailed checks—a 2.5× difference. Automated payment creates behavioral lock-in that reduces churn.

4. Senior Citizens (High-Risk Demographic)

Seniors demonstrate 41.68% churn versus 23.61% for non-seniors, despite similar tenure (33.3 vs 32.2 months) and CLTV ($4,392 vs $4,402). This suggests service complexity or insufficient support for less tech-savvy users.

5. Pricing Impact (Revenue-Churn Tension)

Churned customers paid $74.44/month on average versus $61.27 for retained customers—a 21.5% premium. This $13.17 difference represents the price threshold where retention becomes critically compromised.

Churn Reasons

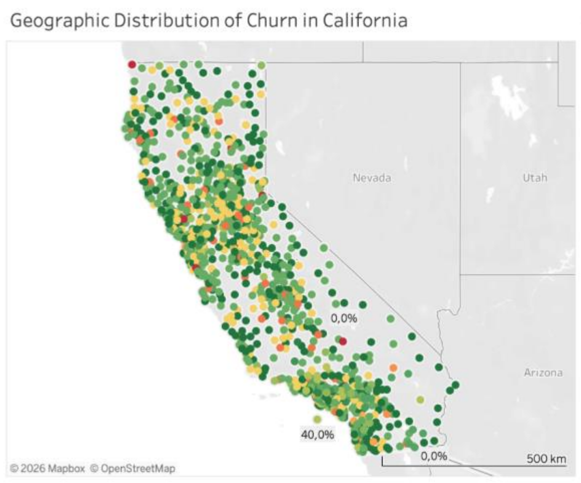
Analysis of 1,869 churned customers revealed five primary categories:

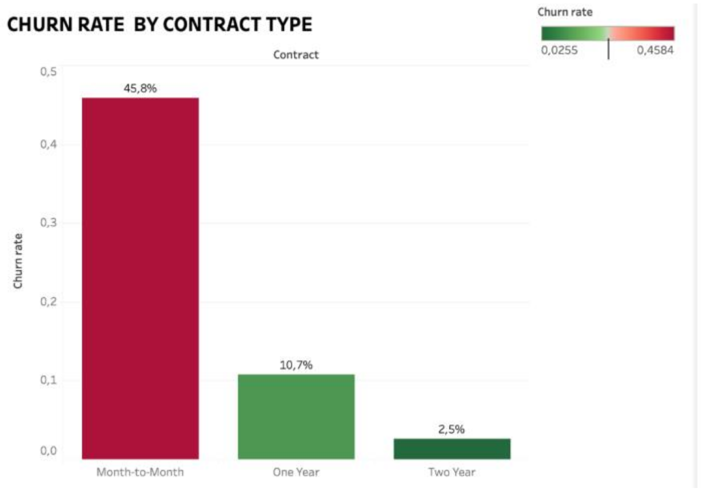
> **Competitor offers (45.0%)** - Better devices (313), better pricing (311), more data (117), higher speeds (100)
> **Attitude/service (16.8%)** - Support person attitude (220), service provider attitude (94)
> **Product dissatisfaction (16.2%)** - Product quality (77), network reliability (72), service gaps (63)
> **Price sensitivity (11.3%)** - Price too high (78), long-distance charges (64), data overage fees (39)
> **Other factors (10.7%)** - Unknown reasons (130), moved (46), deceased (6)

Business Implications

The EDA reveals that churn is primarily driven by **controllable behavioral factors** rather than demographics. Contract type, service quality, and pricing strategy represent actionable intervention points. Notably, demographics (age, gender) show minimal predictive power, indicating that retention efforts should focus on service experience and value proposition rather than customer profiling.

Detailed interactive visualizations exploring these patterns are available in the Tableau dashboard : https://public.tableau.com/app/profile/carmelina.mbesso/viz/Classeur1_17708052695380/STORY?publish=yes


Geographic Distribution of Churn in California
Nevada
Utah
Arizona
0,0%
40,0%
0,0%
500 km
© 2026 Mapbox © OpenStreetMap



Database Type Selection

The database landscape is divided into two main categories: relational databases (SQL) and non-relational databases (NoSQL). Relational databases organize data into structured tables with predefined schemas and use foreign keys to establish relationships between entities. Non-relational databases offer flexible schemas optimized for specific use cases like document storage or key-value pairs.

For this customer churn analysis project, **MySQL** (a relational database management system) was selected for several key reasons.

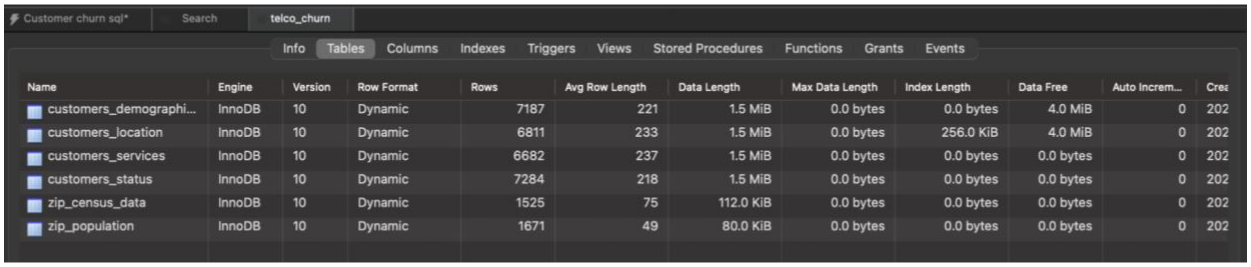
First, the customer data exhibits clear entity relationships with stable schemas. Customers have demographic attributes, location data, service subscriptions, and churn status that naturally map to separate but related tables. The data structure benefits from normalization, where ZIP code information can be stored once and referenced 7,043 times rather than duplicated.

Second, the analytical requirements demand complex multi-table joins and aggregations. Queries frequently need to combine demographics with services and churn data across 4-6 tables simultaneously. SQL provides the expressive power needed for these ad-hoc business intelligence queries.

Third, data integrity is critical for business reporting. Foreign key constraints ensure referential integrity, preventing orphan records. ACID compliance guarantees transaction reliability, and check constraints validate data quality at insertion time.
Finally, MySQL offers mature ecosystem support with extensive Python integration through SQLAlchemy and pandas, industry-standard SQL compatibility, and robust performance optimization tools. This combination of structural advantages, query capabilities, and tooling maturity made MySQL the optimal choice for this project.

Database Creation

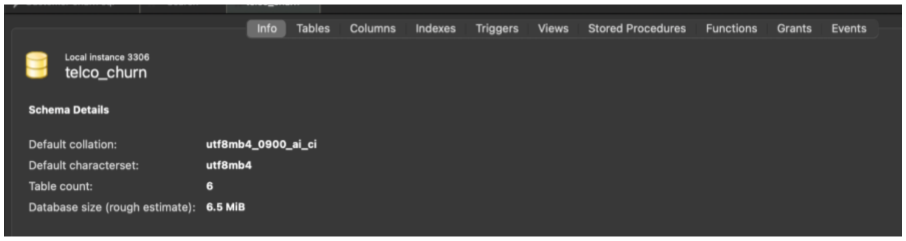
I created a MySQL database called `telco_churn` with 6 tables:



Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Increm...	Crea
customers_demographi...	InnoDB	10	Dynamic	7187	221	1.5 MiB	0.0 bytes	0.0 bytes	4.0 MiB	0	202
customers_location	InnoDB	10	Dynamic	6811	233	1.5 MiB	0.0 bytes	256.0 KiB	4.0 MiB	0	202
customers_services	InnoDB	10	Dynamic	6682	237	1.5 MiB	0.0 bytes	0.0 bytes	0.0 bytes	0	202
customers_status	InnoDB	10	Dynamic	7284	218	1.5 MiB	0.0 bytes	0.0 bytes	0.0 bytes	0	202
zip_census_data	InnoDB	10	Dynamic	1525	75	112.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	202
zip_population	InnoDB	10	Dynamic	1671	49	80.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	202

Database stats:

Platform: MySQL 8.0
Total records: 7,043 customers
Foreign keys enforce data integrity
Proper data types (VARCHAR, INT, DECIMAL, FLOAT)



Local instance 3306
telco_churn

Schema Details

Default collation:	utf8mb4_0900_ai_ci
Default characterset:	utf8mb4
Table count:	6
Database size (rough estimate):	6.5 MiB

Entity Relationship Diagram

I designed a normalized database structure with 6 tables. Here's how they connect:

Relationships:

customers_demographics ← customers_location (by customer_id)

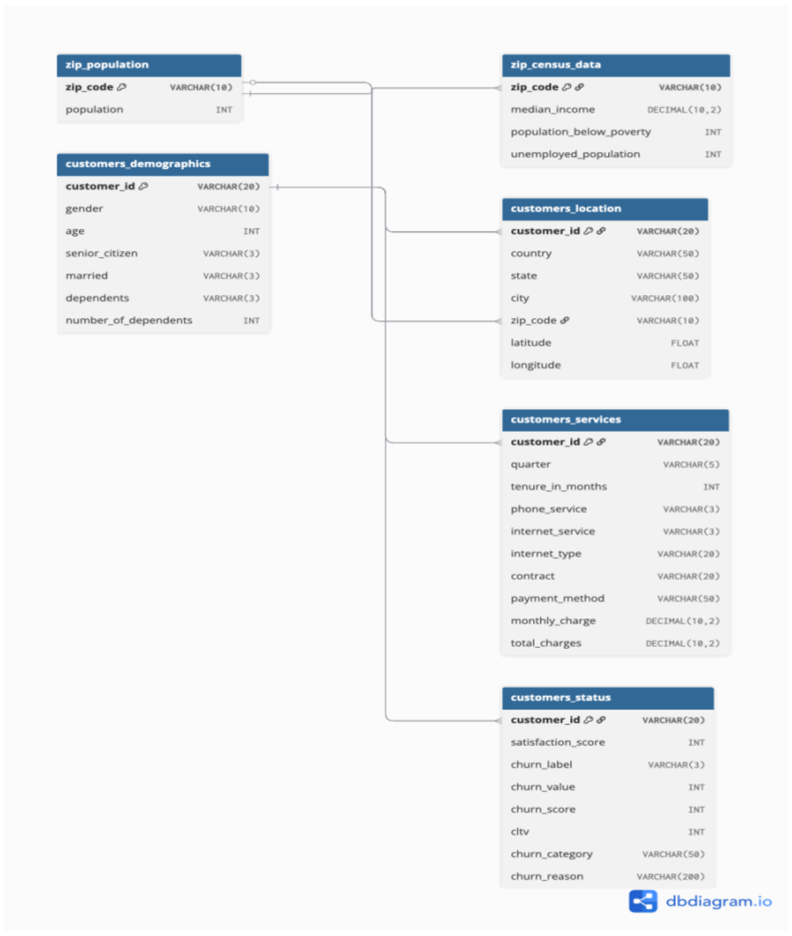
customers_demographics ← customers_services (by customer_id)

customers_demographics ← customers_status (by customer_id)

customers_location ← zip_population (by zip_code)

customers_location ← zip_census_data (by zip_code)

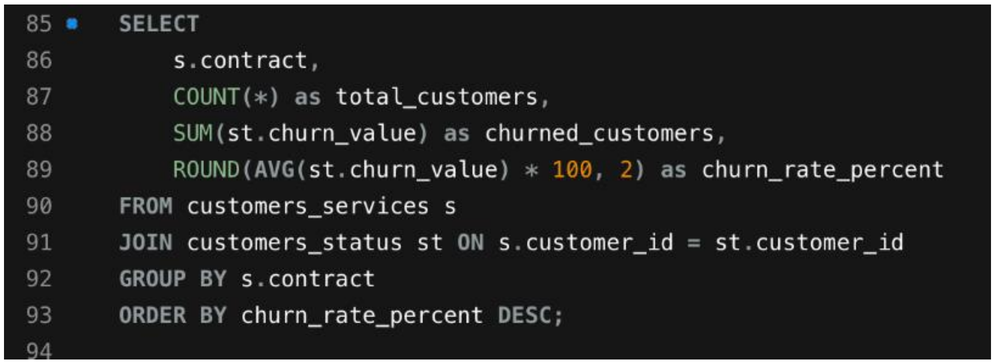
The ERD shows all primary keys, foreign keys, and data types for each column.



SQL Queries

I wrote 5 SQL queries to answer business questions.

Query 1: Churn Rate by Contract Type



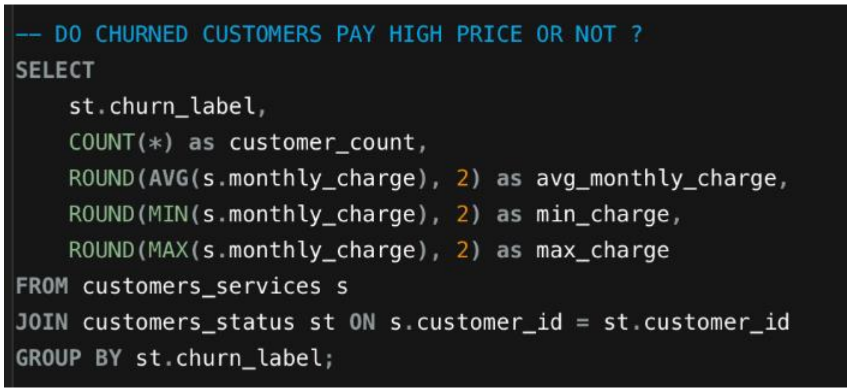
```sql
SELECT
    s.contract,
    COUNT(*) as total_customers,
    SUM(st.churn_value) as churned_customers,
    ROUND(AVG(st.churn_value) * 100, 2) as churn_rate_percent
FROM customers_services s
JOIN customers_status st ON s.customer_id = st.customer_id
GROUP BY s.contract
ORDER BY churn_rate_percent DESC;
```

Result:

Month-to-Month: 45.8% churn
One Year: 10.7% churn
Two Year: 2.5% churn

Business insight: Focus on converting month-to-month customers to annual contracts.

Query 2: Monthly Charge Analysis



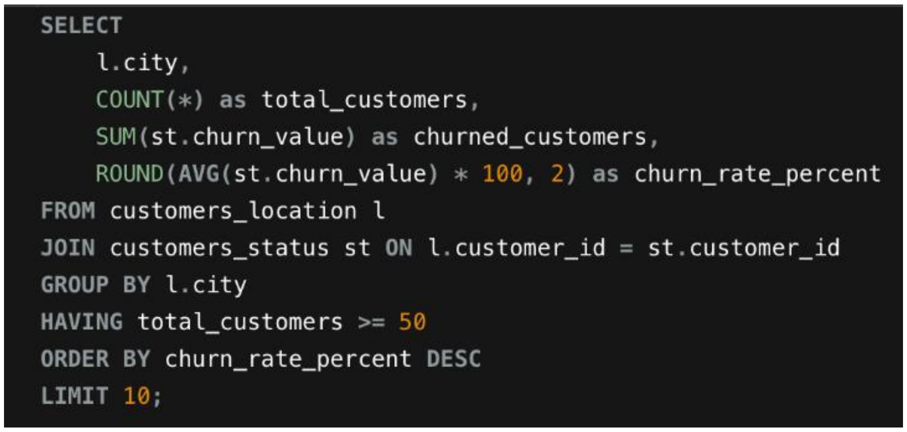
```sql
-- DO CHURNED CUSTOMERS PAY HIGH PRICE OR NOT ?
SELECT
    st.churn_label,
    COUNT(*) as customer_count,
    ROUND(AVG(s.monthly_charge), 2) as avg_monthly_charge,
    ROUND(MIN(s.monthly_charge), 2) as min_charge,
    ROUND(MAX(s.monthly_charge), 2) as max_charge
FROM customers_services s
JOIN customers_status st ON s.customer_id = st.customer_id
GROUP BY st.churn_label;
```

Result:

Churned customers: avg $74.44/month
Retained customers: avg $61.27/month

Business insight: Customers paying over $70-75 have higher churn risk.

Query 3: Geographic Churn Hotspots



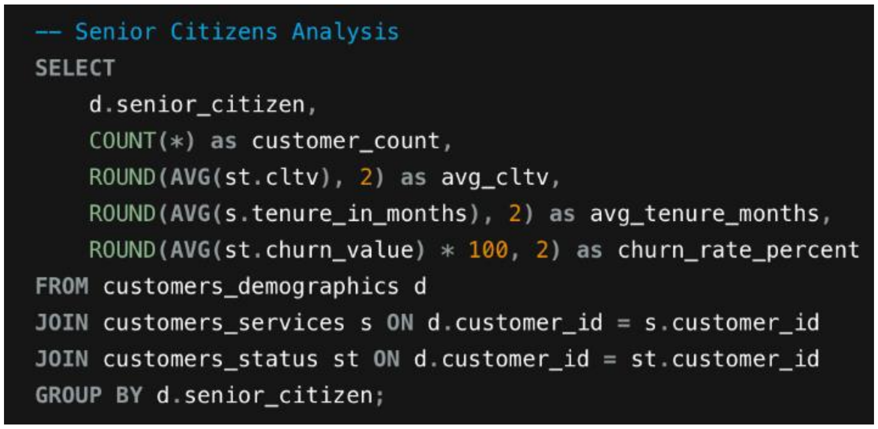
```sql
SELECT
    l.city,
    COUNT(*) as total_customers,
    SUM(st.churn_value) as churned_customers,
    ROUND(AVG(st.churn_value) * 100, 2) as churn_rate_percent
FROM customers_location l
JOIN customers_status st ON l.customer_id = st.customer_id
GROUP BY l.city
HAVING total_customers >= 50
ORDER BY churn_rate_percent DESC
LIMIT 10;
```

Result:

San Diego: 64.91% churn (worst)
Fallbrook: 50.47%
Temecula: 37.35%

Business insight: San Diego needs immediate attention - run targeted retention campaigns there.

Query 4: Senior Citizen Analysis



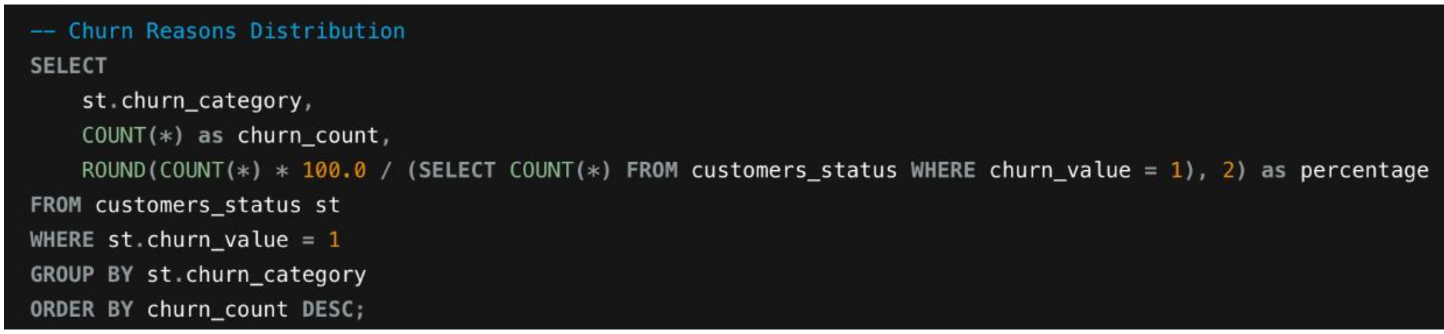
```sql
-- Senior Citizens Analysis
SELECT
    d.senior_citizen,
    COUNT(*) as customer_count,
    ROUND(AVG(st.cltv), 2) as avg_cltv,
    ROUND(AVG(s.tenure_in_months), 2) as avg_tenure_months,
    ROUND(AVG(st.churn_value) * 100, 2) as churn_rate_percent
FROM customers_demographics d
JOIN customers_services s ON d.customer_id = s.customer_id
JOIN customers_status st ON d.customer_id = st.customer_id
GROUP BY d.senior_citizen;
```

Result:

Seniors: 41.7% churn, avg 33.3 months tenure, $4,391 CLTV
Non-seniors: 23.6% churn, avg 32.2 months tenure, $4,401 CLTV

Business insight: Seniors churn way more despite similar tenure and value. Need senior-friendly plans.

Query 5: Churn Category Distribution



```sql
-- Churn Reasons Distribution
SELECT
    st.churn_category,
    COUNT(*) as churn_count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM customers_status WHERE churn_value = 1), 2) as percentage
FROM customers_status st
WHERE st.churn_value = 1
GROUP BY st.churn_category
ORDER BY churn_count DESC;
```

Result:

Competitor:	45%
Attitude:	16.8%
Dissatisfaction:	16.2%
Price:	11.3%
Other:	10.7%

Business insight: Need competitive response strategy - almost half of churn is competitor-driven.

API Development

An API was developed using Flask to make the churn prediction model accessible for real-time business applications. The API transforms the trained machine learning model into an operational tool that can be integrated with customer service systems, CRM platforms, and automated retention workflows.

Technology Stack

Framework: Flask 2.3.3 (Python web framework)
Model: Gradient Boosting Classifier with SMOTE
Serialization: pickle for model and scaler persistence
Data Processing: pandas 2.0.3
HTTP Protocol: RESTful API with JSON responses

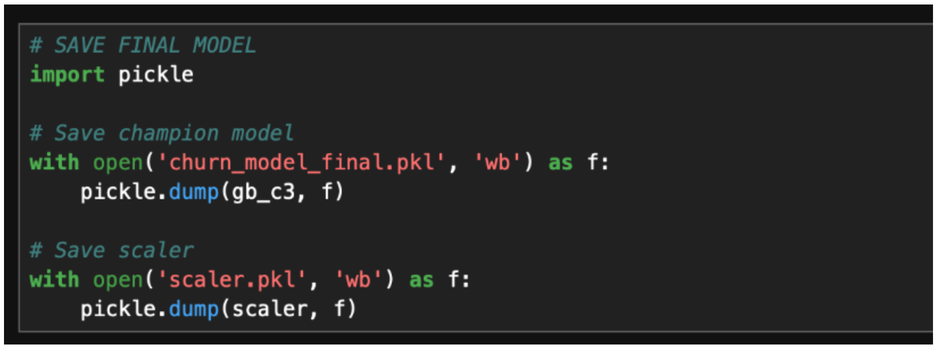
API Architecture

The API follows a three-layer architecture:

Model Layer: Loads the pre-trained Gradient Boosting model (72.46% recall) and StandardScaler at application startup
Preprocessing Layer: Transforms raw customer data into the exact feature format used during model training
Prediction Layer: Generates churn probability, risk classification, and actionable recommendations

Implementation

Model Deployment:
The trained model and preprocessing scaler are saved as pickle files and loaded when the API starts:



```python
# SAVE FINAL MODEL
import pickle

# Save champion model
with open('churn_model_final.pkl', 'wb') as f:
    pickle.dump(gb_c3, f)

# Save scaler
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)
```

Data Preprocessing Pipeline:

Customer data undergoes identical preprocessing to the training phase:

Calculate `total_services` (count of subscribed services)
Encode binary features (gender, Partner, Dependents)
One-hot encode categorical features (Contract, PaymentMethod, etc.)
Scale numerical features (tenure, MonthlyCharges, total_services)

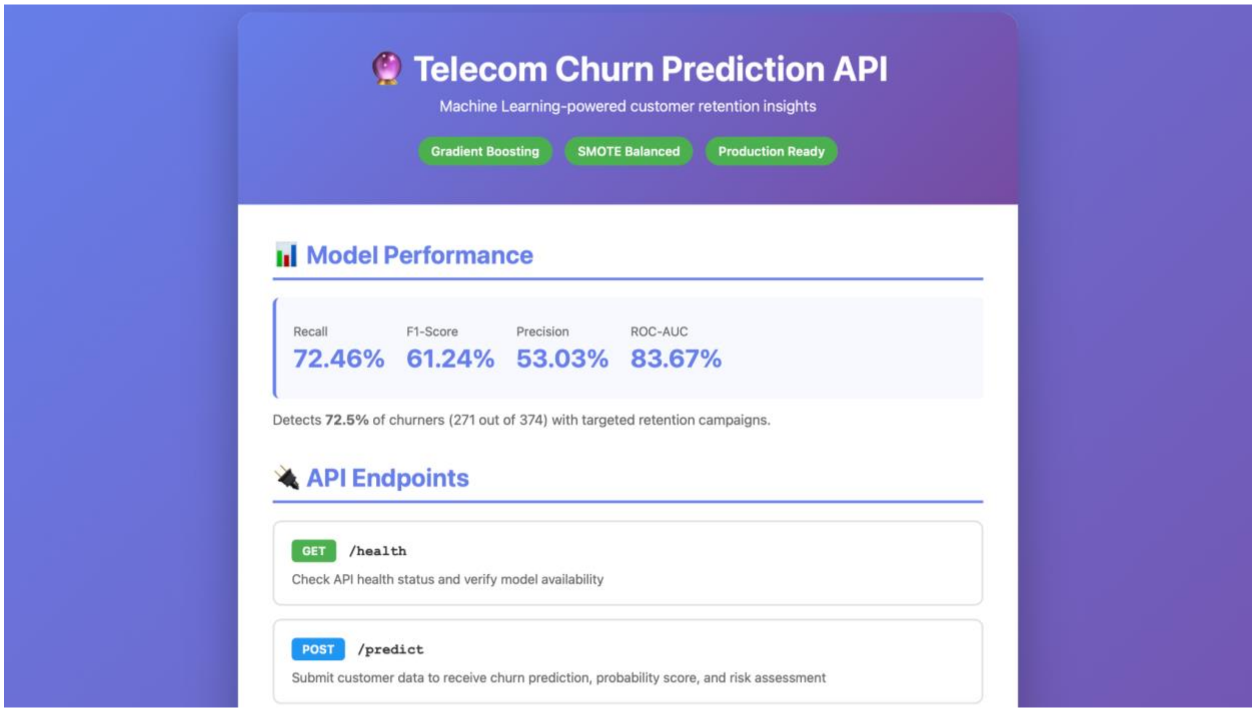
API Endpoints

Three endpoints provide different functionalities:

1. GET / - API Documentation

Returns an interactive HTML page displaying model performance, endpoint descriptions, and risk classification guidelines.

Page d'accueil HTML de l'API sur http://127.0.0.1:5001/


🔮 Telecom Churn Prediction API

Machine Learning-powered customer retention insights

Gradient Boosting | SMOTE Balanced | Production Ready

📊 Model Performance

Recall	F1-Score	Precision	ROC-AUC
72.46%	61.24%	53.03%	83.67%

Detects **72.5%** of churners (271 out of 374) with targeted retention campaigns.

🔌 API Endpoints

GET `/health`
Check API health status and verify model availability

POST `/predict`
Submit customer data to receive churn prediction, probability score, and risk assessment

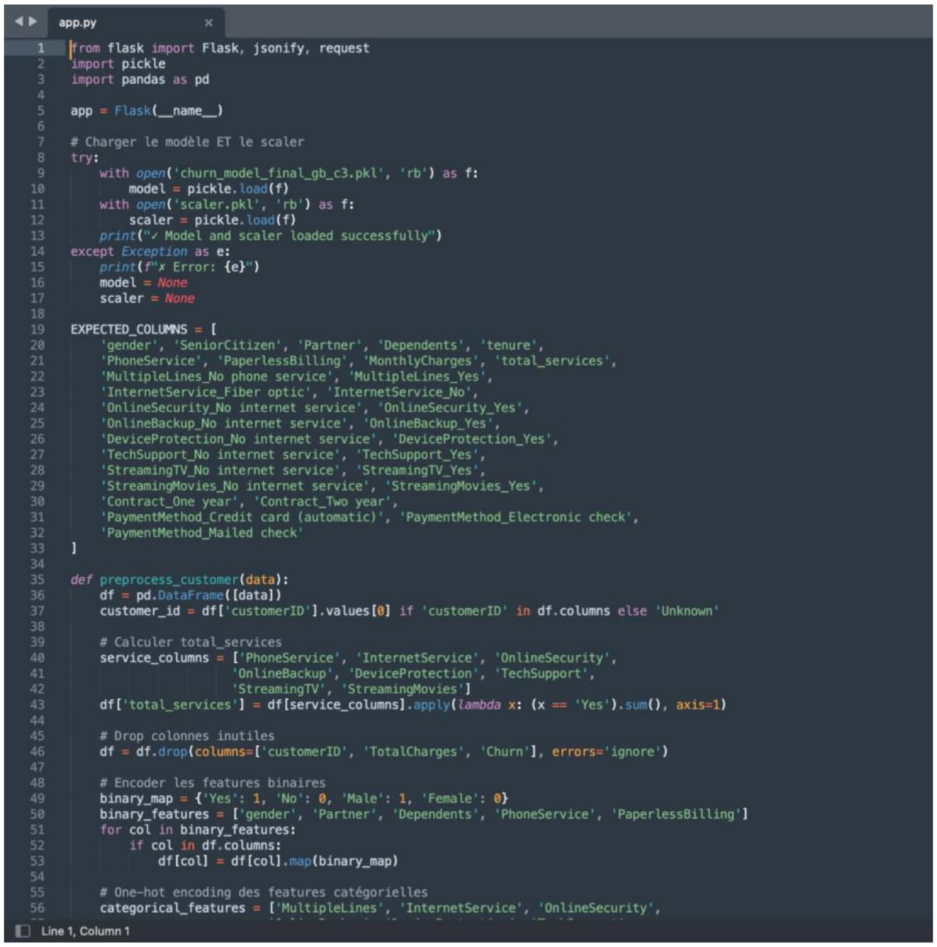
2. GET /health - Health Check

Monitors API availability and confirms that both the model and scaler are loaded correctly.

```
{
  "model_loaded": true,
  "scaler_loaded": true,
  "status": "healthy"
}
```

3. POST /predict - Churn Prediction

Accepts customer data and returns churn probability with risk assessment.



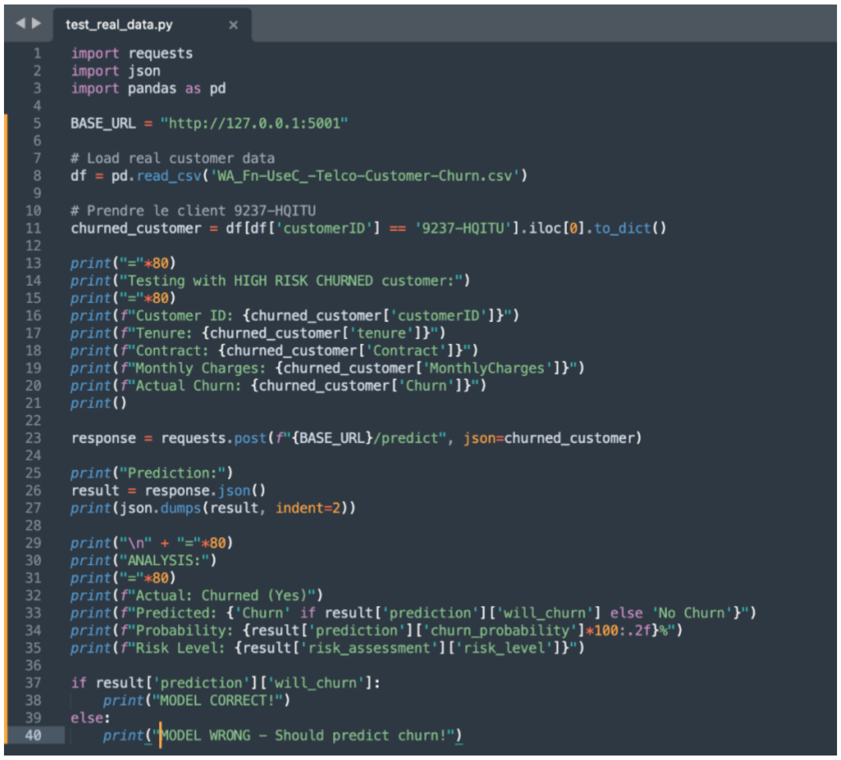
app.py

```python
from flask import Flask, jsonify, request
import pickle
import pandas as pd

app = Flask(__name__)

# Charger le modèle ET le scaler
try:
    with open('churn_model_final_gb_c3.pkl', 'rb') as f:
        model = pickle.load(f)
    with open('scaler.pkl', 'rb') as f:
        scaler = pickle.load(f)
    print("✓ Model and scaler loaded successfully")
except Exception as e:
    print(f"✗ Error: {e}")
    model = None
    scaler = None

EXPECTED_COLUMNS = [
    'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
    'PhoneService', 'PaperlessBilling', 'MonthlyCharges', 'total_services',
    'MultipleLines_No phone service', 'MultipleLines_Yes',
    'InternetService_Fiber optic', 'InternetService_No',
    'OnlineSecurity_No internet service', 'OnlineSecurity_Yes',
    'OnlineBackup_No internet service', 'OnlineBackup_Yes',
    'DeviceProtection_No internet service', 'DeviceProtection_Yes',
    'TechSupport_No internet service', 'TechSupport_Yes',
    'StreamingTV_No internet service', 'StreamingTV_Yes',
    'StreamingMovies_No internet service', 'StreamingMovies_Yes',
    'Contract_One year', 'Contract_Two year',
    'PaymentMethod_Credit card (automatic)', 'PaymentMethod_Electronic check',
    'PaymentMethod_Mailed check'
]

def preprocess_customer(data):
    df = pd.DataFrame([data])
    customer_id = df['customerID'].values[0] if 'customerID' in df.columns else 'Unknown'

    # Calculer total_services
    service_columns = ['PhoneService', 'InternetService', 'OnlineSecurity',
                       'OnlineBackup', 'DeviceProtection', 'TechSupport',
                       'StreamingTV', 'StreamingMovies']
    df['total_services'] = df[service_columns].apply(lambda x: (x == 'Yes').sum(), axis=1)

    # Drop colonnes inutiles
    df = df.drop(columns=['customerID', 'TotalCharges', 'Churn'], errors='ignore')

    # Encoder les features binaires
    binary_map = {'Yes': 1, 'No': 0, 'Male': 1, 'Female': 0}
    binary_features = ['gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling']
    for col in binary_features:
        if col in df.columns:
            df[col] = df[col].map(binary_map)

    # One-hot encoding des features catégorielles
    categorical_features = ['MultipleLines', 'InternetService', 'OnlineSecurity',
```

Line 1, Column 1

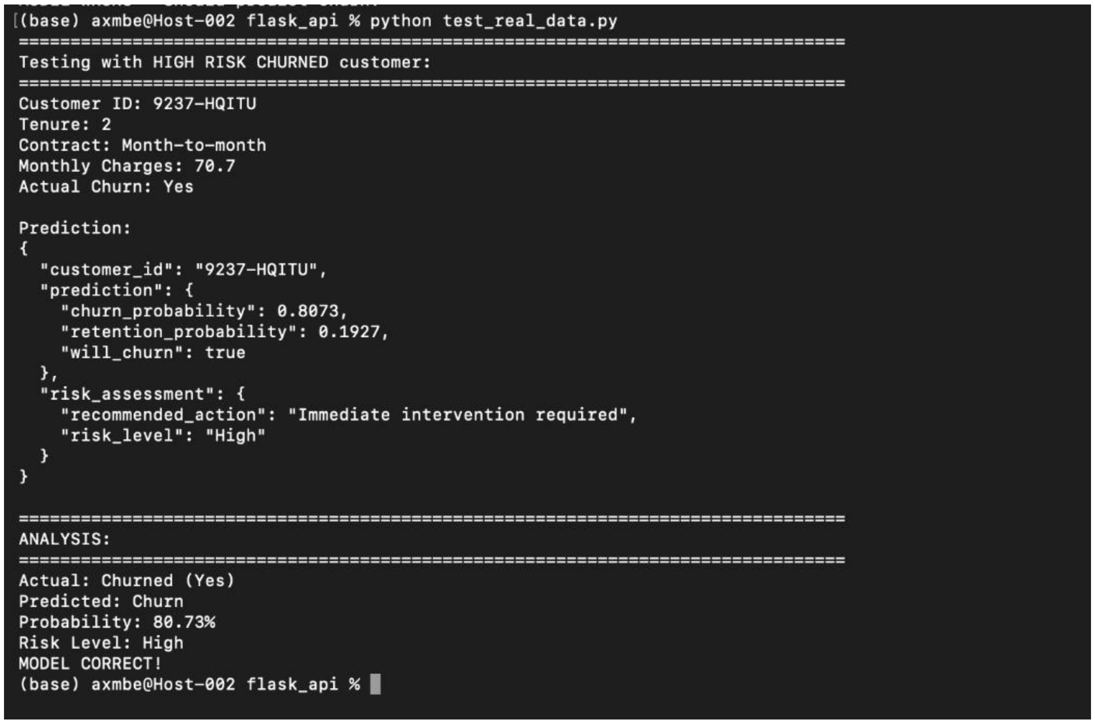
Testing

I tested the API with real customer data:



```
test_real_data.py

import requests
import json
import pandas as pd

BASE_URL = "http://127.0.0.1:5001"

# Load real customer data
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Prendre le client 9237-HQITU
churned_customer = df[df['customerID'] == '9237-HQITU'].iloc[0].to_dict()

print("="*80)
print("Testing with HIGH RISK CHURNED customer:")
print("="*80)
print(f"Customer ID: {churned_customer['customerID']}")
print(f"Tenure: {churned_customer['tenure']}")
print(f"Contract: {churned_customer['Contract']}")
print(f"Monthly Charges: {churned_customer['MonthlyCharges']}")
print(f"Actual Churn: {churned_customer['Churn']}")
print()

response = requests.post(f"{BASE_URL}/predict", json=churned_customer)

print("Prediction:")
result = response.json()
print(json.dumps(result, indent=2))

print("\n" + "="*80)
print("ANALYSIS:")
print("="*80)
print(f"Actual: Churned (Yes)")
print(f"Predicted: {'Churn' if result['prediction']['will_churn'] else 'No Churn'}")
print(f"Probability: {result['prediction']['churn_probability']*100:.2f}%")
print(f"Risk Level: {result['risk_assessment']['risk_level']}")

if result['prediction']['will_churn']:
    print("MODEL CORRECT!")
else:
    print("MODEL WRONG - Should predict churn!")
```

Response :



```
(base) axmbe@Host-002 flask_api % python test_real_data.py
================================================================================
Testing with HIGH RISK CHURNED customer:
================================================================================
Customer ID: 9237-HQITU
Tenure: 2
Contract: Month-to-month
Monthly Charges: 70.7
Actual Churn: Yes

Prediction:
{
  "customer_id": "9237-HQITU",
  "prediction": {
    "churn_probability": 0.8073,
    "retention_probability": 0.1927,
    "will_churn": true
  },
  "risk_assessment": {
    "recommended_action": "Immediate intervention required",
    "risk_level": "High"
  }
}

================================================================================
ANALYSIS:
================================================================================
Actual: Churned (Yes)
Predicted: Churn
Probability: 80.73%
Risk Level: High
MODEL CORRECT!
(base) axmbe@Host-002 flask_api % 
```

The API successfully identifies high-risk customers with probability scores above 70%, enabling proactive retention interventions.

API Documentation Interface

A professional HTML interface provides interactive documentation including:

- Model performance metrics (Recall: 72.46%, F1-Score: 61.24%, ROC-AUC: 83.67%)
- Endpoint specifications with HTTP methods
- Risk classification guidelines
- Business impact calculations (ROI: 210%, Annual revenue protection: $924,358)

API homepage at http://127.0.0.1:5001/]

Deployment Considerations

For production deployment, the following enhancements are recommended:

- Authentication: API key or OAuth2 authentication
- Rate Limiting: Prevent abuse with request throttling
- HTTPS: Secure data transmission with SSL/TLS
- Monitoring: Log predictions and track model performance
- Containerization: Docker deployment for scalability
- Load Balancing: Handle concurrent requests efficiently

Business Value

The API enables:

- Real-time predictions: Instant churn risk assessment during customer interactions
- System integration: Compatible with existing CRM and customer service platforms
- Automated workflows: Trigger retention campaigns based on risk scores
- Scalability: Handle thousands of predictions per day
- Accessibility: Non-technical teams can access ML insights through simple HTTP requests

The API successfully transforms the machine learning model into an operational business tool, making advanced analytics accessible to customer service teams and automated retention systems.

Machine Learning

Objective

Develop a predictive model to identify customers at high risk of churn, prioritizing recall to minimize missed churners while maintaining acceptable precision for cost-effective retention campaigns.

Metric Selection

The choice of evaluation metric fundamentally shapes model optimization. For this churn prediction task, recall was selected as the primary metric based on business cost analysis.
Business Context:
In customer retention, two types of errors have different business impacts:

1. False Negative (FN): Predict customer will stay, but they churn
   - Cost: Lost customer lifetime value (CLTV)
   - Impact: Average CLTV = $3,456 per customer
   - Consequence: Complete revenue loss, no intervention attempted
2. False Positive (FP): Predict customer will churn, but they stay
   - Cost: Unnecessary retention campaign
   - Impact: ~$150 per campaign (discount, agent time, incentive)
   - Consequence: Modest cost, customer relationship potentially strengthened

Error Type	Business Cost	Relative Impact
Miss a churner (FN)	$3,456 (lost CLTV)	23× more expensive
False alarm (FP)	$150 (campaign cost)	Baseline

Cost Comparison:Decision Rationale:

Based on the dataset, the estimated revenue loss per churned customer is $3,456.
Missing a churner costs $3,456, while a false alarm costs only $150. Therefore, it is far more acceptable to over-predict churn (high recall, lower precision) than to under-predict (high precision, low recall).

Metric Definitions:

- Recall = TP / (TP + FN): Percentage of actual churners correctly identified
- Precision = TP / (TP + FP): Percentage of predicted churners who actually churn
- F1-Score: Harmonic mean balancing recall and precision

Target Metrics:

- Primary: Recall ≥ 70% (detect at least 70% of churners)
- Secondary: F1-Score ≥ 60% (maintain balance with precision)
- Constraint: Precision ≥ 50% (avoid excessive false alarms)

Methodology

A systematic approach was employed to compare multiple algorithms and optimization techniques:

Phase 1: Initial Model Screening

- Tested 5 algorithms: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, K-Nearest Neighbors
- Evaluated with and without SMOTE class balancing
- Primary metric: Recall (minimize false negatives)

Phase 2: Systematic Model Comparison

- Selected top 2 models: Gradient Boosting and Logistic Regression
- Tested 4 configurations per model:
    1. Baseline (default parameters)
    2. Hyperparameter tuning (GridSearchCV)
    3. Tuned + SMOTE (class balancing)
    4. All optimized (tuning + SMOTE + threshold optimization)

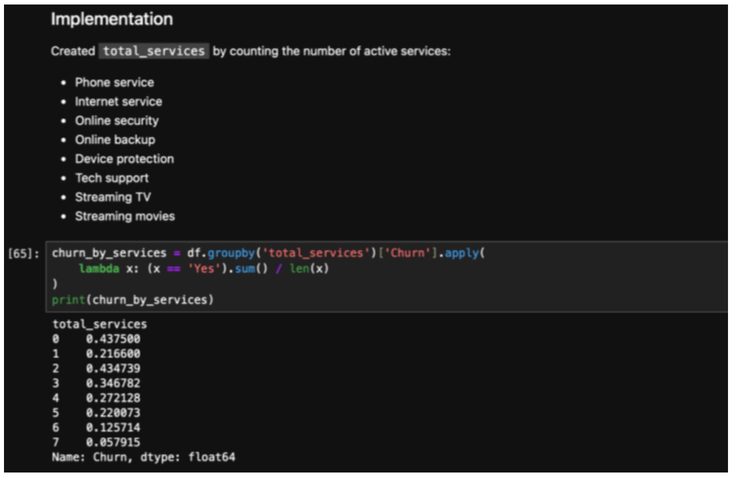
Data Preparation

Target Variable:

- Churn encoded as binary: Yes=1, No=0
- Class distribution: 5,174 retained (73.5%) vs 1,869 churned (26.5%)

Feature Engineering:

Created total_services feature counting active service subscriptions:


Implementation

Created `total_services` by counting the number of active services:

- Phone service
- Internet service
- Online security
- Online backup
- Device protection
- Tech support
- Streaming TV
- Streaming movies

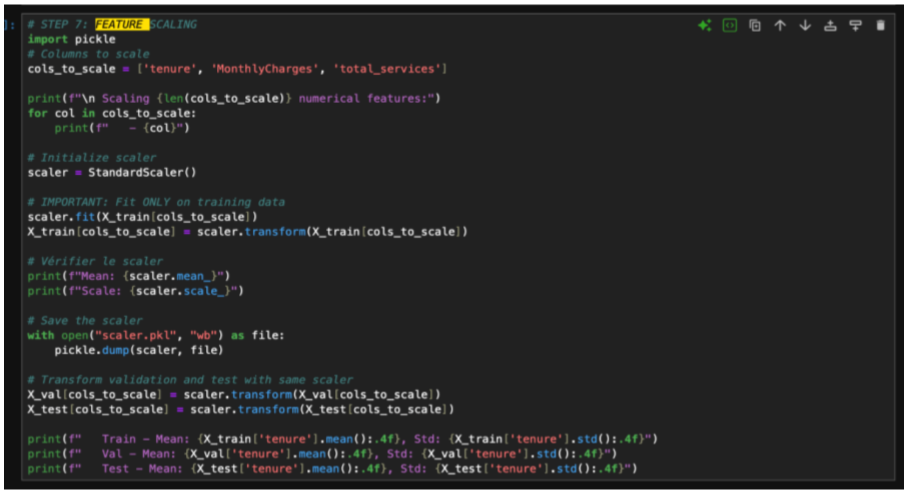
```
[65]: churn_by_services = df.groupby('total_services')['Churn'].apply(
          lambda x: (x == 'Yes').sum() / len(x)
      )
      print(churn_by_services)
```

```
total_services
0    0.437500
1    0.216600
2    0.434739
3    0.346782
4    0.272128
5    0.220073
6    0.125714
7    0.057915
Name: Churn, dtype: float64
```

Feature Selection:

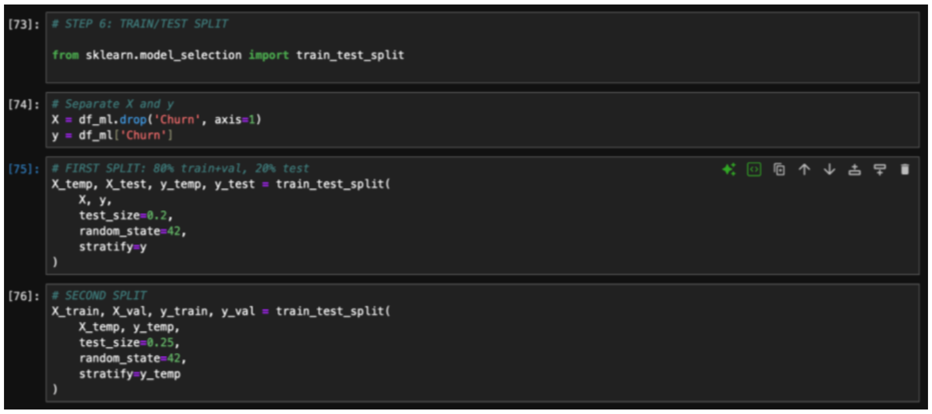
- Dropped: customerID, TotalCharges (redundant with tenure × MonthlyCharges)
- Binary encoding: gender, Partner, Dependents, PhoneService, PaperlessBilling
- One-hot encoding: MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaymentMethod
- Final feature count: 29 features

Feature Scaling:
Standardized numerical features using StandardScaler:



```
# STEP 7: FEATURE SCALING
import pickle
# Columns to scale
cols_to_scale = ['tenure', 'MonthlyCharges', 'total_services']

print(f"\n Scaling {len(cols_to_scale)} numerical features:")
for col in cols_to_scale:
    print(f"   - {col}")

# Initialize scaler
scaler = StandardScaler()

# IMPORTANT: Fit ONLY on training data
scaler.fit(X_train[cols_to_scale])
X_train[cols_to_scale] = scaler.transform(X_train[cols_to_scale])

# Vérifier le scaler
print(f"Mean: {scaler.mean_}")
print(f"Scale: {scaler.scale_}")

# Save the scaler
with open("scaler.pkl", "wb") as file:
    pickle.dump(scaler, file)

# Transform validation and test with same scaler
X_val[cols_to_scale] = scaler.transform(X_val[cols_to_scale])
X_test[cols_to_scale] = scaler.transform(X_test[cols_to_scale])

print(f"   Train - Mean: {X_train['tenure'].mean():.4f}, Std: {X_train['tenure'].std():.4f}")
print(f"   Val - Mean: {X_val['tenure'].mean():.4f}, Std: {X_val['tenure'].std():.4f}")
print(f"   Test - Mean: {X_test['tenure'].mean():.4f}, Std: {X_test['tenure'].std():.4f}")
```

Train-Validation-Test Split:



```
[73]: # STEP 6: TRAIN/TEST SPLIT

      from sklearn.model_selection import train_test_split

[74]: # Separate X and y
      X = df_ml.drop('Churn', axis=1)
      y = df_ml['Churn']

[75]: # FIRST SPLIT: 80% train+val, 20% test
      X_temp, X_test, y_temp, y_test = train_test_split(
          X, y,
          test_size=0.2,
          random_state=42,
          stratify=y
      )

[76]: # SECOND SPLIT
      X_train, X_val, y_train, y_val = train_test_split(
          X_temp, y_temp,
          test_size=0.25,
          random_state=42,
          stratify=y_temp
      )
```

Class Imbalance Handling

Applied SMOTE (Synthetic Minority Over-sampling Technique) to balance the training set:

- Original distribution: 73.5% retained / 26.5% churned
- After SMOTE: 50% retained / 50% churned
- SMOTE applied only to training data to prevent data leakage

Impact of SMOTE:

Gradient Boosting recall improvement: 48.93% → 72.46% (+23.5 percentage points)

SMOTE addresses the class imbalance problem where models naturally bias toward the majority class (retained customers), resulting in poor recall on the minority class (churners). By synthetically generating churner samples, SMOTE forces the model to learn patterns distinguishing churners from retained customers.

Model Selection and Training

Configuration 3: Gradient Boosting + SMOTE (Champion Model)

Selected configuration based on optimal balance of recall and F1-score.
Hyperparameters:

- learning_rate: 0.1
- max_depth: 3
- n_estimators: 100
- Training on SMOTE-balanced data (50/50 split)

Training Process:

from sklearn.ensemble import GradientBoostingClassifier
from imblearn.over_sampling import SMOTE

# Apply SMOTE
smote = SMOTE(random_state=42)

```
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Train model

gb_c3 = GradientBoostingClassifier(
    learning_rate=0.1,
    max_depth=3,
    n_estimators=100,
    random_state=42
)
gb_c3.fit(X_train_smote, y_train_smote)
```

Model Performance

Test Set Results:

Metric	Score	Interpretation
Recall	72.46%	Detects 271 of 374 churners
Precision	53.03%	53% of predictions are correct
F1-Score	61.24%	Balanced performance
Accuracy	75.66%	Overall correctness
ROC-AUC	83.67%	Strong discriminative ability

Confusion Matrix (Test Set):

	Predicted No	Predicted Yes

Actual No	795 (TN)	240 (FP)
Actual Yes	103 (FN)	271 (TP)

Business Impact Metrics:

- Total churners: 374
- Correctly detected: 271 (72.5%)
- Missed churners: 103 (27.5%)
- False alarms: 240
- Campaign efficiency: 53.0% true positives

Generalization Performance:

Metric	Validation	Test	Difference
Recall	0.7246	0.7246	+0.0000
F1-Score	0.6216	0.6124	-0.0091
Precision	0.5442	0.5303	-0.0138
ROC-AUC	0.8318	0.8367	+0.0048

Result: Excellent generalization with F1 difference of only 0.0091, indicating the model performs consistently on unseen data without overfitting.

Feature Importance

Top feature categories by predictive power:

1. Contract & Tenure (51.1%): Contract commitment is paramount

2. Services (25.9%): Service bundle matters
3. Billing & Charges (18.1%): Payment experience impacts retention
4. Demographics (2.6%): Age/gender have minimal impact

Key Insights:

- Month-to-month contracts show 42.7% churn vs 2.8% for two-year contracts
- Contract conversion programs represent the highest-impact retention strategy
- Demographics are NOT key drivers - focus retention on behavioral factors, not customer profiles

Scenario Comparison:

Without Model (Random Targeting):

- Target all 374 churners randomly
- Campaign cost: 374 × $150 = $56,100
- Revenue saved: 374 × 5% × $3,456 = $64,627
- Net benefit: $8,527

With Model (Precision Targeting):

- Target 271 high-probability churners
- Campaign cost: 271 × $150 = $40,650
- Revenue saved: 271 × 5% × $3,456 = $46,829
- Net benefit: $6,179

Model Impact:

- 27.5% reduction in campaign costs
- Maintains 72% of revenue protection
- Estimated annual revenue protection: $924,358 (if deployed across full customer base of 7,043)

Model Limitations

1. 27.5% missed churners: Model does not detect all at-risk customers - 103 churners fall below detection threshold
2. 47% false positive rate: 240 of 511 flagged customers may not actually churn, leading to unnecessary campaign spending
3. Requires regular retraining: Customer behavior patterns evolve; model performance degrades without periodic updates
4. Limited to behavioral data: Does not account for external factors (competitor promotions, economic shifts, service outages)

Recommendations

Immediate Actions:

1. Deploy model via API for real-time predictions during customer interactions
2. Implement tiered retention offers based on risk level (High/Medium/Low)

3. Prioritize intervention for customers in first 12 months (47.7% churn rate)

Short-term Improvements:

1. Launch contract conversion programs (51.1% feature importance = highest impact)
2. Investigate fiber optic service quality issues (40.7% churn rate vs 18.9% for DSL)
3. Develop senior citizen retention program (41.7% churn despite 25% higher CLTV)

Long-term Strategy:

1. Implement quarterly model retraining pipeline with updated customer data
2. Conduct A/B testing of retention strategies to measure actual campaign effectiveness
3. Expand feature set: customer service call logs, network quality metrics, payment history
4. Explore ensemble methods combining multiple algorithms for improved performance

Conclusion

The Gradient Boosting + SMOTE model achieves the project objective of 72.46% recall, successfully identifying nearly three-quarters of churning customers while maintaining economically viable precision.
The prioritization of recall over precision is justified by the 23:1 cost ratio between missed churners and false alarms.
The model demonstrates excellent generalization (F1 difference of 0.009 between validation and test), provides actionable insights through feature importance analysis, and delivers estimated annual revenue protection of $924,358.
Deployment via API enables seamless integration with existing business systems for proactive, data-driven customer retention.

GDPR Compliance

All data used in this project is public and anonymized. No personal information was collected or used.

Data sources:

IBM public dataset (anonymized customer IDs)
US Census Bureau (aggregated ZIP code data)
Web scraping (public company information only)

The project complies with GDPR regulations because:

No personal identifiable information (PII) collected
All data is at aggregate or anonymized level
No sensitive data (health, financial) used
Public sources only

References

Data Sources:

IBM Cognos Analytics - Telco Customer Churn Dataset
https://www.kaggle.com/datasets/ylchang/telco-customer-churn-1113

US Census Bureau API
https://api.census.gov/data/2020/acs/acs5

API: American Community Survey (ACS) 2020 5-Year Estimates
Endpoint: `https://api.census.gov/data/2020/acs/acs5`

Tools:

Python (Pandas, NumPy, Matplotlib, Seaborn)
MySQL 8.0
Google BigQuery : Big query carmelina
Flask
Jupyter Notebook
BeautifulSoup

GitHub: https://github.com/Axoudouxou/final-project-customer-churn-prediction

Tableau :
https://public.tableau.com/app/profile/carmelina.mbesso/viz/Classeur1_17708052695380/STORY?publish=yes