**Customer Churn Prediction — SQL + Random Forest**

* **Project Overview**

This project focuses on predicting **customer churn** (whether a bank customer will exit) using a combination of **SQL for data cleaning, exploration, and preparation** and **Machine Learning (Random Forest)** for modeling.

The dataset (Churn\_Modelling.csv) contains **10,000 customer records** with demographic, financial, and account activity details. The goal is to understand churn drivers, prepare the dataset in SQL, and train a predictive model.

**🔹 Business Understanding**

* **Problem**: Retaining customers is critical for banks. High churn leads to lost revenue and higher acquisition costs.
* **Objective**: Identify customers likely to churn and analyze key factors influencing their decision.
* **Success Metric**: Achieve high recall (catch as many churners as possible) while maintaining good accuracy.

**🔹 Data Understanding**

**Columns**: CustomerId, Gender, Geography, Age, CreditScore, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited.  
**Target Variable**: Exited → 1 = Churned, 0 = Stayed.

**Inspect dataset in SQL**

USE churn\_predection;

-- Look at first few rows

SELECT \* FROM churn\_modelling LIMIT 10;

-- Count rows

SELECT COUNT(\*) AS total\_rows FROM churn\_modelling;

-- Count columns

SELECT COLUMN\_NAME, DATA\_TYPE, IS\_NULLABLE

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE table\_name = 'churn\_modelling';

-- Describe table

DESCRIBE churn\_modelling;

-- Unique customers

SELECT COUNT(DISTINCT CustomerId) AS unique\_customers FROM churn\_modelling;

**🔹 Summary Statistics**

SELECT

MIN(CreditScore) AS min\_credit,

MAX(CreditScore) AS max\_credit,

AVG(CreditScore) AS avg\_credit,

STD(CreditScore) AS std\_credit,

MIN(Age) AS min\_age,

MAX(Age) AS max\_age,

AVG(Age) AS avg\_age,

STD(Age) AS std\_age,

MIN(Balance) AS min\_balance,

MAX(Balance) AS max\_balance,

AVG(Balance) AS avg\_balance,

STD(Balance) AS std\_balance,

MIN(EstimatedSalary) AS min\_salary,

MAX(EstimatedSalary) AS max\_salary,

AVG(EstimatedSalary) AS avg\_salary,

STD(EstimatedSalary) AS std\_salary

FROM churn\_modelling;

**🔹 Churn Distribution**

SELECT Exited, COUNT(\*) AS cnt, -- count of churn

ROUND(COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM churn\_modelling),2) AS pct -- percentage

FROM churn\_modelling

GROUP BY Exited;

**Observation**: About ~20% customers have churned.

**🔹 Churn by Demographics**

-- Gender vs Churn

SELECT Gender, Exited, COUNT(\*) AS cnt

FROM churn\_modelling

GROUP BY Gender ,Exited

ORDER BY Exited;

-- Geography vs Churn

SELECT Geography, Exited, COUNT(\*) AS cnt

FROM churn\_modelling

GROUP BY Geography, Exited;

-- Age groups vs Churn

SELECT CASE

WHEN Age BETWEEN 18 AND 25 THEN '18-25'

WHEN Age BETWEEN 26 AND 35 THEN '26-35'

WHEN Age BETWEEN 36 AND 45 THEN '36-45'

WHEN Age BETWEEN 46 AND 55 THEN '46-55'

ELSE '56+'

END AS age\_group,

Exited,

COUNT(\*) AS cnt

FROM churn\_modelling

GROUP BY age\_group, Exited

ORDER BY age\_group, Exited;

**Observation**: Churn is higher in Germany, among ages **30–45**, and for inactive customers.

**🔹 Feature Signals**

-- Average values for churned vs non-churned

SELECT Exited,

AVG(CreditScore) AS avg\_credit,

AVG(Age) AS avg\_age,

AVG(Balance) AS avg\_balance,

AVG(NumOfProducts) AS avg\_products,

AVG(EstimatedSalary) AS avg\_salary

FROM churn\_modelling

GROUP BY Exited;

* It helps identify **differences in feature behavior** between churned vs. retained customers.
* Example insights might be:
  + Churned customers are slightly **older**.
  + They may have **higher balances** but **fewer products**.
  + Credit scores of churned customers could be lower on average.

This is a **step in feature exploration** — like groupby("Exited").mean() in pandas.

**🔹 Data Preparation in SQL**

**One-hot encoding categorical features**

-- Gender → numerical

SET sql\_safe\_updates = 0;

ALTER TABLE churn\_modelling ADD COLUMN Gender\_Female TINYINT;

ALTER TABLE churn\_modelling ADD COLUMN Gender\_Male TINYINT;

UPDATE churn\_modelling

SET Gender\_Female = CASE WHEN Gender = 'Female' THEN 1 ELSE 0 END,

Gender\_Male = CASE WHEN Gender = 'Male' THEN 1 ELSE 0 END;

SET sql\_safe\_updates = 1;

-- Geography → numerical

ALTER TABLE churn\_modelling ADD COLUMN Geo\_France TINYINT;

ALTER TABLE churn\_modelling ADD COLUMN Geo\_Spain TINYINT;

ALTER TABLE churn\_modelling ADD COLUMN Geo\_Germany TINYINT;

UPDATE churn\_modelling

SET Geo\_France = CASE WHEN Geography = 'France' THEN 1 ELSE 0 END,

Geo\_Spain = CASE WHEN Geography = 'Spain' THEN 1 ELSE 0 END,

Geo\_Germany = CASE WHEN Geography = 'Germany' THEN 1 ELSE 0 END;

**Age bucketing**

ALTER TABLE churn\_modelling ADD COLUMN AgeGroup VARCHAR(10);

UPDATE churn\_modelling

SET Age = CASE

WHEN Age BETWEEN 18 AND 25 THEN 1

WHEN Age BETWEEN 26 AND 35 THEN 2

WHEN Age BETWEEN 36 AND 45 THEN 3

WHEN Age BETWEEN 46 AND 55 THEN 4

ELSE 5

END;

**🔹 Train / Test Split**

ALTER TABLE churn\_modelling ADD COLUMN rand\_val FLOAT;

UPDATE churn\_modelling SET rand\_val = RAND();

CREATE TABLE churn\_train AS

SELECT \* FROM churn\_modelling WHERE rand\_val <= 0.8;

CREATE TABLE churn\_test AS

SELECT \* FROM churn\_modelling WHERE rand\_val > 0.8;

-- Check distribution

SELECT COUNT(\*) AS train\_rows FROM churn\_train;

SELECT COUNT(\*) AS test\_rows FROM churn\_test;

**🔹 Feature Weights (Simple Proxy for Modeling)**

CREATE TABLE feature\_weights AS

SELECT 'CreditScore' AS feature, AVG(CASE WHEN Exited=1 THEN CreditScore ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN CreditScore ELSE NULL END) AS weight

FROM churn\_train

UNION ALL

SELECT 'Age', AVG(CASE WHEN Exited=1 THEN Age ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Age ELSE NULL END)

FROM churn\_train

UNION ALL

SELECT 'Balance', AVG(CASE WHEN Exited=1 THEN Balance ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Balance ELSE NULL END)

FROM churn\_train

UNION ALL

SELECT 'Gender\_Female', AVG(CASE WHEN Exited=1 THEN Gender\_Female ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Gender\_Female ELSE NULL END)

FROM churn\_train

UNION ALL

SELECT 'Gender\_Male', AVG(CASE WHEN Exited=1 THEN Gender\_Male ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Gender\_Male ELSE NULL END)

FROM churn\_train

UNION ALL

SELECT 'Geo\_France', AVG(CASE WHEN Exited=1 THEN Geo\_France ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Geo\_France ELSE NULL END)

FROM churn\_train

UNION ALL

SELECT 'Geo\_Spain', AVG(CASE WHEN Exited=1 THEN Geo\_Spain ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Geo\_Spain ELSE NULL END)

FROM churn\_train

UNION ALL

SELECT 'Geo\_Germany', AVG(CASE WHEN Exited=1 THEN Geo\_Germany ELSE NULL END) - AVG(CASE WHEN Exited=0 THEN Geo\_Germany ELSE NULL END)

FROM churn\_train;

SELECT \* FROM feature\_weights;

**🔹 Prediction & Evaluation**

ALTER TABLE churn\_test ADD COLUMN predicted\_score FLOAT;

UPDATE churn\_test

SET predicted\_score =

CreditScore \* (SELECT weight FROM feature\_weights WHERE feature='CreditScore') +

Age \* (SELECT weight FROM feature\_weights WHERE feature='Age') +

Balance \* (SELECT weight FROM feature\_weights WHERE feature='Balance') +

Gender\_Female \* (SELECT weight FROM feature\_weights WHERE feature='Gender\_Female') +

Gender\_Male \* (SELECT weight FROM feature\_weights WHERE feature='Gender\_Male') +

Geo\_France \* (SELECT weight FROM feature\_weights WHERE feature='Geo\_France') +

Geo\_Spain \* (SELECT weight FROM feature\_weights WHERE feature='Geo\_Spain') +

Geo\_Germany \* (SELECT weight FROM feature\_weights WHERE feature='Geo\_Germany');

ALTER TABLE churn\_test ADD COLUMN predicted\_churn TINYINT;

UPDATE churn\_test

SET predicted\_churn = CASE

WHEN predicted\_score >= 0 THEN 1

ELSE 0

END;

-- Accuracy

SELECT

SUM(CASE WHEN predicted\_churn = Exited THEN 1 ELSE 0 END)/COUNT(\*) AS accuracy

FROM churn\_test;

**🔹 Observations**

* Customers aged **30–45** are most likely to churn.
* **Germany** shows significantly higher churn than France or Spain.
* Inactive customers with high balance but fewer products churn more.
* Gender has relatively little effect compared to balance & geography.

**🔹 Tech Stack**

* **Database**: MySQL
* **Languages**: SQL, Python (for ML)
* **Visualization**: Power BI / Tableau
* **Model**: Random Forest