Project: Bank Customer Churn Modeling

---

1. Project Overview (Explain First)

This project focuses on predicting whether a bank customer will leave the bank in the future. Customer churn is a serious problem for banks because acquiring new customers is more expensive than retaining existing ones.

---

2. Business Problem

The bank wants to **identify customers who are at high risk of leaving** so that the bank can:

- Offer personalized discounts
- Improve customer service
- Reduce revenue loss

This makes the project **business-oriented**, not just technical.

---

3. Dataset Description

The dataset contains **historical customer information** such as:

Customer Demographics

- Age
- Gender
- Geography

Banking Details

- Credit Score
- Account Balance
- Number of Products
- Estimated Salary

Behavioral Information

- Is Active Member
- Has Credit Card

Target Variable

- **Exited (0 = Not Churned, 1 = Churned)**

---

4. Data Cleaning & Preprocessing

I performed the following steps:

Handling Missing Values

- Checked for null values
- Filled or removed them based on importance

Encoding Categorical Variables

- Converted **Gender** and **Geography** into numerical format using encoding techniques

Feature Scaling

- Applied scaling to numerical columns to improve model performance

Class Imbalance Handling

- The dataset had fewer churned customers
- Used **SMOTE** to balance the classes

---

5. Exploratory Data Analysis (EDA)

EDA helped me understand **why customers leave**.

Key Insights Found

- Customers with **low activity** churn more
- Customers with **high balance but fewer products** are more likely to leave
- Older customers show higher churn rate

Visualization Used

- Bar charts
- Histograms
- Correlation heatmap

EDA ensured I made **data-driven decisions** before modeling.

---

6. Model Building

I trained multiple machine learning models to compare performance:

Models Used

1. **Logistic Regression** – Baseline, easy to interpret
2. **Random Forest** – Handles non-linear relationships
3. **XGBoost** – Best performing model
4. **SMOTE + Models** – To handle imbalance

---

7. Model Evaluation

I evaluated models using:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

Why Recall is Important Here

Missing a churn customer is costly, so recall is more important than accuracy.

---

8. Model Explainability (SHAP)

To make the model **trustworthy and interpretable**, I used **SHAP**.

SHAP Explained Simply

SHAP tells **why** the model predicted a customer will churn.

Top Churn Factors Identified

- Low activity status
- High balance with low engagement
- Fewer products
- Age

This helps banks **take targeted actions**.

---

9. Best Model Selection

XGBoost gave the best performance due to:

- Handling complex patterns
- Better performance on imbalanced data
- High recall and F1-score

---

10. Model Saving & Deployment Readiness

- Saved the final trained model using **Pickle (.pkl)**
- Model is ready for deployment using **FastAPI + Streamlit**

---

11. Tools & Technologies

- **Programming:** Python
- **Libraries:** Pandas, NumPy, Scikit-Learn
- **ML Models:** Logistic Regression, Random Forest, XGBoost
- **Explainability:** SHAP
- **Visualization:** Matplotlib, Seaborn
- **Deployment:** FastAPI, Streamlit

---

12. Real-World Use Case (Interview Gold ⭐)

If a customer is predicted to churn, the bank can:

- Offer special interest rates
- Provide relationship manager support
- Suggest better banking products

This **reduces churn and increases customer lifetime value**.

---

13. Challenges Faced

- Handling imbalanced data
- Choosing the right evaluation metric
- Explaining model decisions to non-technical users

---

14. What I Learned

- How to build an **end-to-end industry-level ML project**
- Importance of business understanding
- Model interpretability using SHAP

---

15. Final Strong Interview Closing Line

This project helped me understand how machine learning can directly solve real banking business problems by predicting customer churn and improving retention strategies.

Technical Deep Dive – Bank Customer Churn Modeling (ML Round)

---

1. Problem Formulation

- This is a **binary classification problem**
- Target variable:
  Exited
    - `1` → Customer churned
    - `0` → Customer stayed
- Objective: **Maximize recall for churn class** while maintaining good overall performance.

---

2. Data Understanding

- Dataset contains **numerical + categorical features**
- Typical size: ~10K records
- Key challenges:
    - **Class imbalance**
    - Mix of linear and non-linear relationships

---

3. Data Preprocessing (Technical Explanation)

3.1 Handling Missing Values

- Checked null distribution using `.isnull().sum()`
- Since missing values were minimal, used:
    - Mean / median for numerical features
    - Mode for categorical features

3.2 Encoding Categorical Variables

- **Gender** → Label Encoding
- **Geography** → One-Hot Encoding
  Reason:

Tree models handle one-hot features better and avoid ordinal bias.

3.3 Feature Scaling

- Applied **StandardScaler**
- Scaling required for:
    - Logistic Regression
- Not mandatory for tree-based models, but kept pipeline consistent.

3.4 Train-Test Split

- Used **stratified split**
- Ratio: `80% train / 20% test`
  Reason:

Maintains churn ratio in both sets.

---

4. Handling Class Imbalance

- Churn class was under-represented (~20%)
- Used **SMOTE (Synthetic Minority Over-sampling Technique)**

Why SMOTE?

- Creates synthetic churn samples
- Prevents model bias toward majority class
- Improved recall for churn customers

---

5. Exploratory Data Analysis (EDA – Technical View)

Univariate Analysis

- Distribution of Age, Balance, Credit Score

Bivariate Analysis

- Churn vs Age
- Churn vs IsActiveMember
- Churn vs NumOfProducts

Multivariate Analysis

- Correlation heatmap
- Found:
  - Negative correlation between activity and churn
  - Age positively correlated with churn

---

6. Feature Engineering

- Dropped non-informative columns:
  - `CustomerId`
  - `Surname`
- Created clean, model-ready feature matrix `X`
- Target vector `y = Exited`

---

7. Models Implemented

7.1 Logistic Regression

- Baseline linear model
- Helps understand:
  - Feature coefficients
  - Direction of impact
- Limitation:
  - Cannot capture non-linear patterns

7.2 Random Forest

- Ensemble of decision trees
- Advantages:

    - Handles non-linearity
    - Robust to outliers
- Tuned parameters:
    - `n_estimators`
    - `max_depth`
    - `min_samples_split`

7.3 XGBoost (Final Model)

- Gradient boosting framework
- Sequential tree learning
- Handles:
    - Complex interactions
    - Imbalanced datasets well

Key Hyperparameters Tuned

- `n_estimators`
- `learning_rate`
- `max_depth`
- `subsample`
- `colsample_bytree`

---

8. Model Evaluation Metrics

Used **multiple metrics**, not just accuracy.

Why Accuracy is NOT Enough

- Dataset is imbalanced
- High accuracy can hide poor churn detection

Metrics Used

- **Precision** → How many predicted churns were correct
- **Recall** → How many actual churns were captured ⭐
- **F1-Score** → Balance of precision and recall
- **Confusion Matrix**

Recall was prioritized because missing a churn customer is costly.

---

9. Model Explainability using SHAP

Used **SHAP (SHapley Additive exPlanations)** for interpretability.

Why SHAP?

- Explains predictions at:
  - Global level (feature importance)
  - Local level (individual customer)

Top Features Contributing to Churn

1. IsActiveMember
2. Age
3. Balance
4. NumOfProducts
5. CreditScore

This makes the model **business-trustworthy**.

---

10. Model Selection

- Compared all models on:
  - Recall
  - F1-Score
- **XGBoost selected** because:
  - Highest recall for churn class
  - Stable performance
  - Better generalization

---

11. Model Saving

- Saved trained model using **pickle**

```
import pickle
pickle.dump(model, open("churn_model.pkl", "wb"))
```

---

12. Deployment Architecture (Brief)

- Backend: **FastAPI**
- Frontend: **Streamlit**
- User inputs customer data
- Model returns:
  - Churn probability
  - Churn decision

13. Production Considerations

- Data drift monitoring
- Periodic retraining
- Threshold tuning based on business cost

14. Final ML-Round Closing Statement (IMPORTANT)

This project demonstrates my ability to build an end-to-end machine learning solution—from data preprocessing and model selection to explainability and deployment—while keeping business impact as the primary focus.

Model Selection Reasoning (Sequential – Interview Ready)

1️⃣ Logistic Regression – Baseline Model

Why I Started with Logistic Regression

- It is a **simple and interpretable** model
- Helps establish a **baseline performance**
- Shows **direction and importance** of features

What I Learned from It

- Identified which features increase or decrease churn
- Understood linear relationships
- Easy to explain to business stakeholders

Limitation

- Assumes **linear decision boundary**
- Could not capture **complex customer behavior**

📌 Interview Line

"I used Logistic Regression first to build a baseline and understand feature impact."

2️⃣ Random Forest – Handling Non-Linearity

Why I Moved to Random Forest

- Customer churn is **non-linear**
- Random Forest:
  - Handles complex feature interactions
  - Reduces overfitting using ensemble learning

Improvements Over Logistic Regression

- Better recall
- More stable predictions
- Handles outliers well

Limitation

- Less interpretable
- Performance plateaus after a point

📌 Interview Line

"Random Forest helped capture non-linear patterns that Logistic Regression couldn't."

3️⃣ XGBoost – Performance Optimization

Why XGBoost

- Gradient Boosting builds trees **sequentially**
- Focuses more on **hard-to-predict churn cases**
- Known for **high performance in tabular data**

Advantages Over Random Forest

- Better generalization
- Higher recall and F1-score
- Handles imbalance more effectively

Why It Became Final Model

- Best churn recall
- Lowest false negatives
- Consistent performance on test data

📌 Interview Line

“XGBoost gave the best balance between performance and generalization, especially for churn customers.”

4️⃣ SMOTE + Models – Handling Class Imbalance

Why SMOTE Was Needed

- Churn class was under-represented
- Models were biased toward non-churn customers

How SMOTE Helped

- Created synthetic churn samples
- Balanced the dataset
- Improved recall significantly

Applied With

- Logistic Regression + SMOTE
- Random Forest + SMOTE
- XGBoost + SMOTE

Final Observation

- **XGBoost + SMOTE** gave the highest recall

📌 Interview Line

“SMOTE helped the model focus on minority churn cases, which is critical from a business perspective.”

Model	Purpose	Outcome
Logistic Regression	Baseline & interpretability	Simple, low recall
Random Forest	Non-linear patterns	Better recall
XGBoost	Performance optimization	Best overall
SMOTE + XGBoost	Handle imbalance	Highest churn detection