Customer Churn and Win-Back Targeting

A. Business problem

You work for a subscription business. Leadership wants two things that can be used soon.

1. A reliable score for each active customer that estimates the chance they will cancel in the next period.
2. A simple, budget aware rule that tells the retention team whom to contact and how many to contact per 1,000 customers.

Your job is not only to produce a model. Your job is to deliver an end-to-end process that turns raw data into a calibrated probability and then into an action a manager can follow.

B. Data and release plan

You will receive two files on Canvas so every team works from the same schema.

- train.csv is available now. It contains features and labels. Treat it as the past.
- holdout_features.csv will be released on Monday, Nov 10,2025. It contains features only.

The holdout may differ from training in two ways:

1. the overall churn rate may shift by as much as five percentage points, and
2. the mix of add-on services may shift modestly.

C. Tools and where to work

Choose one path.

- **Python in Google Colab** as a single notebook.
- **R in Posit Cloud** as a single R Markdown.

Your notebook or document must run end-to-end from a clean runtime without manual fixes.

Suggested headings inside your notebook or document:

1. Title, team, date
2. Business problem
3. Leakage policy
4. Setup
5. Load train.csv

6. Features
7. Baseline model (logistic)
8. Other models and calibration
9. Decision rule and cost table
10. Save figures and files
11. Holdout scoring cell or chunk
12. Clear recommendation for managers
13. Appendix: AI use, prompts, and reflection

D. Leakage policy and how you will enforce it

Definition. Leakage is any use of information that would not exist at the time you decide whom to contact.

Assume you score customers at the end of a billing period to plan outreach for the next 30 days. Only information available up to that scoring time is allowed.

Your report must include one paragraph that lists the leakage risks you checked and how you prevented different sources of leakage.

E. Training, validation, and calibration

Work only on train.csv until the holdout is released. Use either 5-fold cross-validation or a simple time-aware split after sorting by tenure. Report:

- **Discrimination** on validation data (AUC).
- **Calibration** on validation data (Brier score).

Freeze your final pipeline before you touch the holdout file.

F. Models you may use

Train all three models so you can compare interpretability (logistic) with performance (RF, GBT).

- Baseline: **logistic regression** (you may use it with or without regularization, Regularization is optional).
- Two additional Models: random forest and gradient boosted trees. [**Caution:** The churn label is class imbalanced (the 'Yes' class is a minority).]

Calibrate the final model (Platt scaling) and include the reliability curve. If you compress many related fields with **PCA**, explain in plain words what each component roughly captures.

G. Required execution evidence

Your notebook or document must generate the following from code:

1. oof_predictions.csv built on train.csv.
2. figures/calibration.png with the reliability curve.
3. An ablation table[1] that shows how AUC and Brier[2] change when you drop each feature group you defined.

We will re-run your notebook and we should be able to reproduce your results.

H. Holdout protocol

On Monday, Nov 10, you will receive your team's holdout_features.csv.

- Run the single Holdout Cell to read the file, score it with your frozen pipeline, and write predictions.csv with two columns in the exact input order: customer_id,p_churn
- After labels are released, add a short "Holdout results" paragraph and a holdout-only calibration plot.

Important note on feature engineering and the holdout file: If you created new variables (engineered features) during training, for example by combining existing columns, creating ratios, or generating binary indicators, remember that the holdout file released on November 10 will contain only the original raw features, not your engineered ones.

You are responsible for ensuring that the same feature-engineering steps applied to the training data are also applied to the holdout data in exactly the same way and in the same order.
The simplest approach is to define your feature-engineering code as reusable functions or as part of a reproducible pipeline that transforms both the training and holdout data consistently.
Failure to do this will result in missing-column errors or, worse, mismatched features between training and prediction.

---

[1] See Appendix
[2] See Appendix

I. Turning scores into actions

A manager can spend a fixed amount per 1,000 customers. You must choose one threshold on p_churn that fits the budget.

- Present a small **cost table** at the chosen threshold. Show contacts per 1,000, expected saves, expected cost, and expected net value.
- State the rule in one sentence that a manager can follow. Example: "Contact anyone with p_churn ≥ 0.28, except customers with tenure under two months."

J. AI use, citation, and reflection

You may use an LLM for writing help, boilerplate code, or idea generation. You may not use it to fabricate results.

Include a final section titled **AI use, prompts, and reflection** with:

- what you used the model for,
- prompts or a brief prompt log,
- a citation such as:
  "OpenAI, ChatGPT (GPT-5 Thinking). Conversation with the author, November 2025. Prompts and excerpts documented in the appendix."
- a short reflection on what helped, what did not, and one rule you will follow next time.

Variable Description

ID — Unique customer identifier.

gender — Male or Female.

SeniorCitizen — 1 if the customer is a senior citizen, 0 otherwise.

Partner — Customer has a partner: Yes or No.

Dependents — Customer has dependents: Yes or No.

tenure — Number of months the customer has stayed with the company.

PhoneService — Customer has phone service: Yes or No.

MultipleLines — Customer has multiple phone lines: Yes, No, or “No phone service.”

InternetService — Type of internet connection: DSL, Fiber optic, or No.

OnlineSecurity — Online security add-on: Yes, No, or “No internet service.”

OnlineBackup — Online backup add-on: Yes, No, or “No internet service.”

DeviceProtection — Device protection add-on: Yes, No, or “No internet service.”

TechSupport — Tech support add-on: Yes, No, or “No internet service.”

StreamingTV — Streaming TV add-on: Yes, No, or “No internet service.”

StreamingMovies — Streaming movies add-on: Yes, No, or “No internet service.”

Contract — Contract term: Month-to-month, One year, or Two year.

PaperlessBilling — Billing is paperless: Yes or No.

PaymentMethod — Payment method: Electronic check, Mailed check, Bank transfer (automatic), or Credit card (automatic).

MonthlyCharges — Current monthly charge amount.

TotalCharges — Total amount charged to date (may contain blanks that must be coerced to numeric during cleaning).

Churn — Target label: Yes if the customer left during the target window; No otherwise.

Appendix: Glossary of Key Terms and Concepts

This glossary clarifies technical terms used throughout the project instructions. It was prepared with the understanding that students enter this course with varying levels of prior exposure to machine learning workflows. For some, these explanations may serve as a quick refresher; for others, they will provide a useful reference for unfamiliar concepts. Please use it as needed, according to your background and experience.

Ablation Table

1. What is an ablation table?

An **ablation table** is a way to understand how much each feature group contributes to your model's performance. You do this by **removing one group at a time**, retraining your model, and measuring how performance changes.

2. Why it matters in this project

In the churn project, you're building a model using a lot of input features:

- Internet services
- Contract terms
- Billing details
- Demographics
  ...and more.

But not all of these are equally helpful and some could even be hurting performance (especially if they are noisy).

The ablation table helps you:

- Understand which feature groups are most important
- Detect groups that might contain leakage
- Justify what features to keep in a final model
- Communicate value to stakeholders in plain terms

3. How to do an ablation test step-by-step

Let's say you grouped your features like this:

Feature Group	Example Variables
InternetServices	StreamingTV, OnlineSecurity, InternetService
ContractInfo	Contract, PaperlessBilling, PaymentMethod
BillingInfo	MonthlyCharges, TotalCharges
Demographics	gender, SeniorCitizen, Partner, Dependents

Now, here's what to do:

Step 1: Train your full model (baseline)

Train your best model on all features. Save the baseline performance:

- AUC (Area Under Curve)
- Brier Score (for calibration)

Let's say this gives:

AUC = 0.836
Brier = 0.187

Step 2: Remove one feature group

Remove the first group — say InternetServices. Then retrain the model using everything except those variables.

Record the new AUC and Brier score.

Repeat this for each group.

4. What the table looks like

Feature Group Removed	AUC	Change in AUC	Brier Score	Change in Brier
None (Full model)	0.836	—	0.187	—
InternetServices	0.801	−0.035	0.200	+0.013
ContractInfo	0.828	−0.008	0.188	+0.001
BillingInfo	0.810	−0.026	0.195	+0.008
Demographics	0.834	−0.002	0.187	0.000

5. How to interpret the table

- **Large drop in AUC** → this group matters a lot for ranking customers.
- **Big increase in Brier score** → this group helps calibrate probabilities better.

- **Small or no change** → the group might not help much and could be removed to simplify the model.
- **Improved performance after removing a group** → a possible sign of noise, overfitting, or leakage in that group.

In the hypothetical example above:

- InternetServices and BillingInfo are both important.
- Demographics doesn't help much and might be dropped or deprioritized.
- Removing InternetServices made calibration worse (+0.013 Brier), so that group helps produce more trustworthy probabilities.

Remember: For the ablation table, calculate AUC and Brier using cross-validation on the training data, not the holdout. Use the same validation folds across all ablation runs.

Do you need an ablation table if you already have feature importance from random forest or gradient-boosted trees?

Yes. Tree-based feature importance shows which individual variables contributed most to splitting decisions within a trained model. However, it does not account for feature redundancy, cannot detect harmful inputs such as leaky or overfit features, and does not reflect how the model would perform without those variables. An ablation table addresses this by measuring the global impact of removing an entire logical group of features and retraining the model. It reveals the true, non-redundant contribution of each group and uniquely identifies feature sets that, when excluded, improve model performance.

AUC (Area Under the ROC Curve)

A performance metric that measures how well the model ranks customers who churn above those who do not.

- AUC = 1.0 means perfect ranking; 0.5 means no better than random.
- Use it to report **discrimination** how well your model separates churners from non-churners.

Brier Score

Brier score checks how close your predicted chances are to what actually happened. For each case, take your predicted probability p and the real outcome y. $y = 1$ if it happened, 0 if not. Compute $(p - y)^2$. Average that number over all cases. That average is the Brier score.

- Lower scores mean better calibration.

- Unlike AUC, it rewards both accuracy *and* reliability of probabilities.

Calibration and Platt Scaling

1. Why calibration matters

When you train a machine-learning model (like a random forest or gradient-boosted tree), its "probabilities" are not always true probabilities, they are often just scores between 0 and 1 that *rank* customers but don't tell you real-world likelihoods.

2. What calibration does conceptually

Calibration learns a simple mapping between what the model predicts (raw score) and what actually happens (true outcome) on validation data.

3. What Platt scaling actually is

Platt scaling is the simplest form of calibration.
It fits a tiny logistic regression whose *input* is your model's raw score and whose *output* is the observed churn flag (1 = Yes, 0 = No).

Discrimination

How well a model ranks positive cases (churners) above negative ones.
Measured by **AUC**—good discrimination means churners consistently receive higher risk scores.

Reliability Curve (Calibration Plot)

A graph comparing predicted churn probabilities with actual churn rates.

- The 45° diagonal line represents perfect calibration.
- Points above the line mean the model underestimates churn; below the line means it overestimates.
  You will save this figure as figures/calibration.png.

Leakage / Data Leakage

Data leakage occurs when a predictor variable (feature) contains information that would not be available at the time a real prediction is made. Even a small amount of leakage can make a model appear far more accurate during training than it will be in real-world deployment.

Types of Leakage

1. Target Leakage (most relevant here)

Target leakage occurs when a feature is created using data recorded **after** the target outcome happened. In other words, the feature leaks information from the *future* back into the *past*.

Example (Churn project):
Imagine a feature named Post_Churn_Support_Calls is accidentally included in the training data. If a customer has a value in this column, it means they must have already churned (the target event). A model using this variable will look nearly perfect on training data, but it's cheating as it's using the answer key.

How to prevent it:

- Always check when each variable is recorded relative to the event you're predicting.
- Ensure features only use data available up to the scoring date (the observation window).

2. Train–Test Contamination

This type of leakage happens when preprocessing steps use information from the entire dataset including the test or validation portion before training the model.

Examples:

- Calculating scaling parameters (mean, standard deviation) using all rows before splitting the data.
- Imputing missing values using the overall median instead of computing it on the training subset.

When this happens, the model has indirectly "seen" information from the future (validation or holdout data), making validation scores unrealistically high.

How to prevent it:

- Always split the data first.
- Fit preprocessing steps (imputation, scaling, encoding, PCA, calibration) only on the training folds.
- Apply the fitted transformations to the validation and holdout sets.

Cross-Validation (CV)

A method to evaluate models on unseen data without using the holdout file.
In **5-fold CV**, the training data is split into five parts ("folds"); each fold is used once for validation while the model trains on the remaining four. The final metric is the average across folds.

Out-of-Fold Predictions (OOF)

Predictions made on validation folds during cross-validation, each row's prediction comes from a model that never saw that row. OOF predictions simulate how the model would behave on new data and are used to calculate honest AUC, Brier score, and calibration.

Feature Engineering

Creating or transforming variables to improve their predictive power. Feature engineering adds business logic to raw data.

Regularization

A technique used in logistic regression to prevent overfitting by shrinking large coefficients.

- **L1 (Lasso)** forces some coefficients to zero (feature selection).
- **L2 (Ridge)** keeps all coefficients but reduces their magnitude.
  Regularization stabilizes the model and improves generalization.

PCA (Principal Component Analysis)

A dimensionality-reduction method that compresses many correlated numeric variables into a few components while preserving most variation.

Baseline Model

Your simplest, interpretable model, here, a **logistic regression**, used as a reference to compare advanced models. Improvements in AUC or Brier relative to the baseline show added value from complexity.

Class Imbalance

A situation where one target class (e.g., churn = "Yes") is much smaller than the other. Imbalance makes accuracy misleading.

Holdout Set

The unseen dataset reserved for the final evaluation. You score it **once**, after freezing your final model, and never use it for tuning or feature comparisons.

Frozen Pipeline

A final version of your modeling workflow with no further code or parameter changes. Once frozen, it can be safely applied to new data (the holdout) for unbiased evaluation.

Cost Table

A small summary table showing how your chosen threshold translates into business impact:

- Contacts per 1,000 customers
- Expected saves
- Outreach cost
- Net value
  Used to justify your chosen threshold to management.

Threshold (Decision Cut-off)

A probability value that separates “contact” from “do not contact.”
Example: *Contact anyone with* $p\_churn \geq 0.28$*, except customers with tenure* < 2 *months.*
Thresholds are chosen using the cost table, not by maximizing accuracy.

Class Weights / Stratified Split

Techniques to handle imbalance.

- **Class weights:** give more importance to the minority class during training.
- **Stratified split:** ensures that both train and validation sets have the same churn proportion. (*Hint:* See class notes on Imbalanced data)

Setup Section (in your notebook)

Prepares the environment and data for modeling: import packages, set random seed, load files, fix missing values, recode categories, and verify shapes and column names.

Features Section (in your notebook)

Builds the actual model inputs: select variables, encode categorical features, create featured engineered fields, apply PCA if used, and define final training matrices `X_train` and `y_train`.

Train–Test Contamination

A form of data leakage that happens when preprocessing steps (scaling, encoding, imputation) use information from both training and validation data. Always fit these steps on the training fold only, then apply them to validation or holdout data.

AI Use Log

A short record describing when and how you used large-language-model tools (e.g., ChatGPT) for writing, coding, or idea generation. Must include a citation and brief reflection.