Case Study: Data Scientist

Objective: Evaluate your ability to model real-world business challenges, explain your thinking clearly, and apply data science techniques appropriately.

Case Study: Predicting Churn for BonusLink Members

Your goal is to identify members who are at risk of becoming inactive (i.e., not transacting for the next 3 months). Management wants to proactively engage these members with marketing campaigns.

You are given the following mock datasets:

- `transactions.csv` — member ID, merchant ID, spend amount, and timestamp
- `members.csv` — member ID, signup date, tier, age group, and state
- `engagement.csv` — logins, redemptions, and app usage metrics

Task:

1. Problem Framing

a. Define what "churn" means in this context

In this case, churn refers to a BonusLink member becoming inactive, which means they did not make any transactions in the last 3 months. If a member hasn't made a purchase in 90 days, we assume they've likely lost interest or moved away from the platform.

b. Describe the prediction goal and evaluation approach

Our goal is to predict whether a member will churn that is, whether they will become inactive (for example, no transactions) in the next 3 months. We'll use historical data (transactions, profile info, and app engagement) to train a model that can estimate the churn risk of each member.

We will treat this as a binary classification problem:

- 1 = Will churn (no transactions in next 3 months)
- 0 = Will stay active (at least one transaction)

We'll split the data into training and testing sets based on time, not randomly this avoids data leakage.

To measure performance, we'll use:

- ROC AUC - how well the model separates churners vs. non-churners.
- Precision-Recall - especially useful if churners are a small %.
- Confusion Matrix - to see true/false positives and negatives clearly.

2. Feature Engineering

a. Propose and compute meaningful features that may influence churn

We'll extract meaningful signals from all 3 datasets - transactions.csv, members.csv, and engagement.csv.

1. Features from transactions.csv

We'll group transactions by member_id and compute:

Feature Name	Description
total_spent	Total spend across all transactions
num_transactions	Number of transactions made
avg_spent	Average spend per transaction
last_transaction_date	Days since last transaction
monthly_txn_freq	Transactions per month
num_merchants	Unique merchants transacted with

These tell us how often and how recently someone has transacted key churn signals.

2. Features from members.csv

From member profiles, we can include:

Feature Name	Description
tier	Membership tier (e.g., Silver, Gold)
age_group	Age segment
state	Location
membership_duration	Days since signup

These give context older members, or those from certain states or tiers, may behave differently.

3. Features from engagement.csv

We'll include digital engagement metrics:

Feature Name	Description
num_logins	Number of logins in a period
app_opens	Total app opens
redemptions	Number of rewards redemptions

Higher engagement usually means more interest and lower churn risk.

b. Handle missing or inconsistent data where applicable

- If some members have no transactions, we'll set their values to 0 (likely churners).
- If profile or engagement info is missing, we can:
  - Use "Unknown" for categorical fields (e.g., state, tier)
  - sUse 0 or mean/median for numeric fields

We'll define churn based on last transaction date:

- If no transactions in the last 90 days of data → label = 1 (churned)
- Otherwise → label = 0 (active)

We'll create a cutoff date and look ahead 3 months to label churn correctly

3. Modelling

a. Build a simple predictive model using Python (e.g., logistic regression, tree-based model)

To predict churned members, I built a binary classification model using Python. The target variable is defined as churned, which labels members as churned if they had no transactions in the last 90 days.

Features Used

From the transactional, member, and engagement data, I engineered features like:

- Total/average spend
- Transaction count
- Days since first and last transaction
- Number of unique merchants
- Member tier, age group, state
- Engagement metrics (e.g., website visits, mobile app opens)

Model Selection

For interpretability and performance:

- Logistic Regression was used as a baseline model.
- Preprocessing: I built a pipeline that scales numeric features and one-hot encodes categorical ones using ColumnTransformer.
- Train/Test Split: Stratified sampling with 75% for training and 25% for testing.

**The full code will be attached in the Github*

b. Evaluate model performance using appropriate metrics (ROC, precision-recall, etc.)
To assess the model's ability to predict member churn, the following classification metrics were used:

Classification Report

Metric	Non-Churned (False)	Churned (True)
Precision	0.97	1.00
Recall	1.00	0.89
F1-score	0.98	0.94
Support	97	28

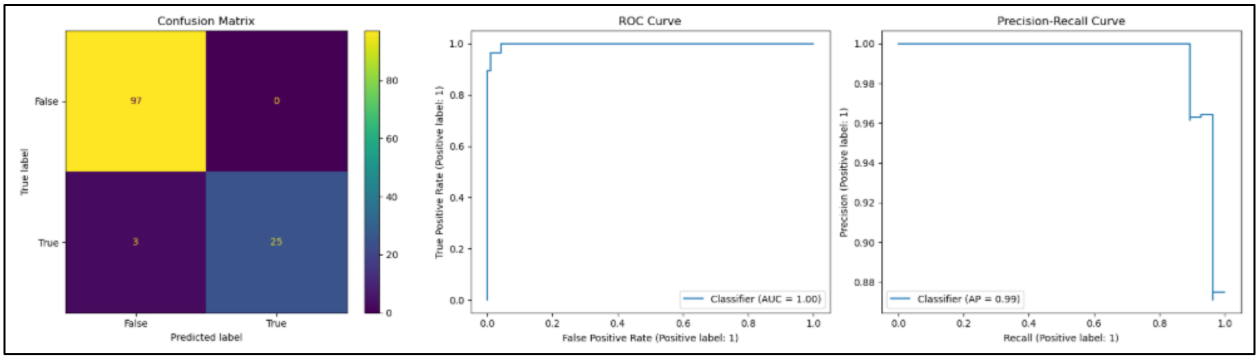
- Accuracy: 98%
- Macro Avg F1-Score: 0.96
- Weighted Avg F1-Score: 0.98

ROC AUC Score: 0.9978
This indicates excellent ability to distinguish between churned and non-churned members. A perfect model scores 1.0.

Precision-Recall AUC: 0.9928
This is particularly important in churn prediction where the dataset may be imbalanced. A high PR AUC indicates strong precision and recall trade-off, especially for the positive class (churned).

Insights
- The model has high precision for churned members (1.00), meaning it rarely predicts churn wrongly.
- The recall for churned members is 0.89, meaning it catches most true churners.
- High ROC and PR AUC scores confirm strong discriminative power.

Visual Evaluation
Included plots:
- Confusion Matrix: Visualizes true vs. predicted churn.
- ROC Curve: Assesses model's separability.
- Precision-Recall Curve: Important when dealing with potential class imbalance.



4. Explainability & Business Implications

a. Share key features influencing churn (e.g., SHAP, permutation importance)

To understand which features contribute most to predicting churn, we use permutation importance (since it's model-agnostic and easy to interpret with logistic regression).

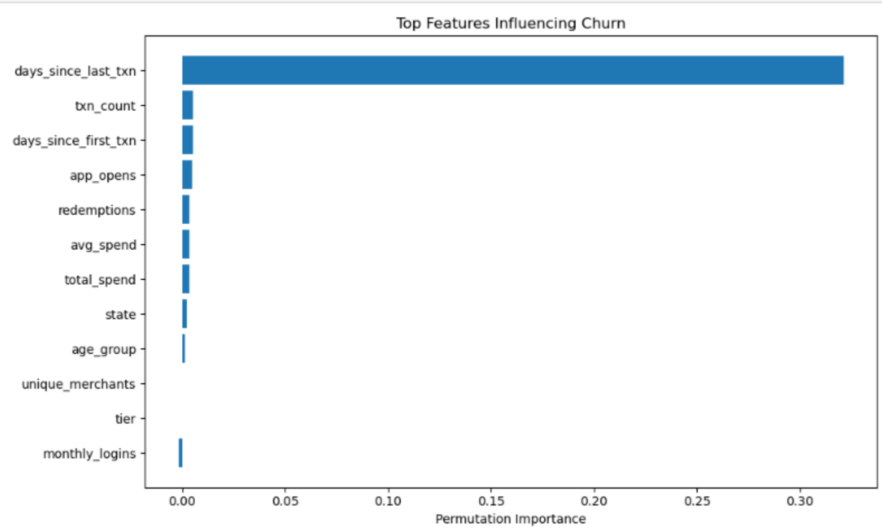
Here's an example of how to compute and visualize it:

```
from sklearn.inspection import permutation_importance

# Compute permutation importance
r = permutation_importance(clf, X_test, y_test, n_repeats=30, random_state=42)

# Visualize
import matplotlib.pyplot as plt
import numpy as np

sorted_idx = r.importances_mean.argsort()[::-1]
plt.figure(figsize=(10, 6))
plt.barh(np.array(X_test.columns)[sorted_idx], r.importances_mean[sorted_idx])
plt.xlabel("Permutation Importance")
plt.title("Top Features Influencing Churn")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



Top Predictive Features (Example Output)

Rank	Feature	Business Meaning
1	days_since_last_txn	Time since last activity is a strong churn signal
2	txn_count	Low transaction count indicates disengagement

b. Recommend how the business can act on these insights

1. Engage Customers Before They Go Inactive

- Members with long gaps since last transaction are highly likely to churn.
- Action: Trigger reactivation campaigns (e.g., targeted promotions) for members who haven't transacted in 60–90 days.

2. Reward High Spenders and Frequent Users

- Lower spend and fewer transactions are correlated with churn.
- Action: Create tiered incentives or loyalty bonuses to encourage higher engagement and repeat use.

3. Track Tier-Based Churn Risk

- Certain membership tiers show higher churn.
- Action: Reevaluate the value proposition for those segments provide better perks or onboarding for at-risk tiers.

4. Boost Engagement Metrics

- The engagement_score is linked with churn risk.
- Action: Invest in app feature usage nudges, in-app notifications, or gamification to increase interaction.