AI Problem Analysis Report

1. Problem Definition

In today's competitive business environment, customer retention is critical to long-term profitability. For a large retail chain, one key challenge is identifying customers who are likely to stop shopping—commonly referred to as "churn." To address this, we propose developing an AI-based predictive system that can accurately forecast whether a customer is likely to churn within the next three months based on their purchase history and engagement behavior. This system will allow the marketing team to intervene early, re-engage at-risk customers, and improve overall retention.

The primary objectives of this AI system are threefold. First, it aims to improve customer retention by identifying high-risk churn customers through data-driven insights. Second, it enables the marketing department to personalize communication and campaigns to maximize the chance of customer re-engagement. Finally, the system supports the company's broader strategic goal of increasing customer lifetime value by targeting and preserving its existing customer base.

Two main stakeholders will benefit from this project. The **marketing team** will rely on the system's predictions to guide outreach strategies and promotional efforts. The **data science team**, on the other hand, will be responsible for building, testing, and maintaining the model, ensuring it remains accurate and effective over time.

To measure the success of the predictive model, we propose using **customer retention rate** as a key performance indicator (KPI). This metric tracks the percentage of identified at-risk customers who continue to shop with the brand after targeted marketing interventions, providing a direct measurement of the model's business impact.

2. Data Collection and Preprocessing

Accurate churn prediction requires rich, high-quality data. For this model, two main data sources will be utilized. The first is **transaction history**, which includes data such as how frequently a customer shops, the amount they spend, and the types of products they purchase. The second is **customer engagement logs**, including digital interactions like email opens, app usage, website visits, and support requests. Together, these datasets provide a comprehensive view of customer behavior and engagement.

However, the use of historical data can introduce potential biases. One significant concern is **recency bias**, where newer customers may not have had enough time to generate meaningful behavior patterns. As a result, the model might inaccurately classify them as likely to churn due to insufficient activity history, even though they are simply in the early stages of their customer journey.

Before model training, several preprocessing steps are necessary to ensure clean, usable data. First, **missing data must be handled** carefully. For instance, if engagement data is missing for a particular customer, this could either indicate low activity or a logging error. Depending on the context, we might fill these values with zeros (indicating no activity) or use mean or median imputation. Second, **normalization** will be applied to continuous variables like purchase frequency and total spend to bring them into the same scale, which helps improve model performance. Third, we must **encode categorical variables**, such as product category or customer region, into numerical formats. This is often done using one-hot encoding, which transforms categories into binary features suitable for most machine learning algorithms.

3. Model Development

For the task of predicting customer churn, we recommend using a **Random Forest Classifier**. This ensemble learning algorithm is well-suited for classification problems, especially when the data contains a mix of numerical and categorical variables. Random forests are known for their robustness against overfitting, their ability to handle missing values, and their strong performance across a variety of use cases. An added benefit is that they offer feature importance metrics, allowing the business team to understand which factors most influence churn predictions.

To train and evaluate the model effectively, we will split the dataset into three parts: **70% for training**, **15% for validation**, and **15% for testing**. The training set will be used to fit the model, the validation set will be used to fine-tune hyperparameters and prevent overfitting, and the test set will provide an unbiased evaluation of the final model's performance.

During model tuning, we will focus on two important hyperparameters. The first is **n_estimators**, which defines the number of trees in the forest. Increasing this number can improve accuracy but also increases computational cost. The second is **max_depth**, which limits how deep each decision tree can grow. This helps control model complexity and prevents overfitting, especially when working with noisy data.

---

4. Evaluation and Deployment

To assess the model's effectiveness, we will use two main evaluation metrics. The first is the **F1 score**, which is the harmonic mean of precision and recall. This metric is particularly useful in churn prediction where false positives (identifying a loyal customer as at-risk) and false negatives (missing a customer who is truly at risk) both have business consequences. The second is the **AUC-ROC score**, which measures the model's ability to distinguish between churn and non-churn cases across various threshold levels. A high AUC score indicates strong predictive discrimination, even when classes are imbalanced.

After deployment, one of the primary challenges will be **monitoring for concept drift**. Concept drift refers to the phenomenon where the statistical properties of the target variable (in this case, customer churn) change over time. This could be due to market shifts, new product launches, economic changes, or evolving customer preferences. If not addressed, concept drift can degrade model performance significantly. To detect drift, we will regularly monitor prediction accuracy and compare it to actual churn outcomes. A sudden drop in performance will trigger an investigation and, if needed, model retraining with updated data.

One technical hurdle during deployment is ensuring **scalability**. The model needs to serve predictions for potentially millions of customers in real-time or near-real-time. This requires an infrastructure that can handle large-scale inference with low latency. Possible solutions include deploying the model using cloud-based tools with autoscaling features (such as AWS SageMaker or GCP AI Platform), using batch prediction pipelines during low-traffic periods, or compressing the model using tools like ONNX or TensorFlow Lite to reduce memory footprint.