Domain Background

In this project, we will train and deploy a model that **predicts the completion probability of a customer offer**. For this purpose, we will use simulated data from the **Starbucks Reward mobile app** that contains information about customers, offers, and transactions. One possible use of this model would be a **recommendation system that displays to each customer the offer with the highest probability of being redeemed**. The business value of deploying a recommendation system like this can be huge for marketing-intensive companies like Starbucks. By **tailoring coupons to individual customers**, which generate the highest likelihood of purchase with the lowest incentive, the **expected return on all marketing activities can be maximized**.

Problem Statement

To build the recommendation system we must follow **two consecutive steps**:

Step 1: Build A Predictive Model

This part will be the major focus of this project. We will train and **deploy a model that predicts the completion probability** for a given customer and offer combination. Our ultimate goal is to **create an endpoint for providing these predictions** to other teams.

Step 2: Build A Recommendation System Upon The Predictive Model

This is a potential use case of our endpoint and will be secondary in this project. For all available offers, we predict the completion probability of a given customer and **choose the offer with the highest completion probability** for this customer.

Datasets and Inputs

The data is contained in **three files**:

- **portfolio.json** - containing offer ids and meta data about each offer (duration, type, etc.)
- **profile.json** - demographic data for each customer

- **transcript.json** - records for transactions, offers received, offers viewed, and offers completed

Here is the schema and explanation of each variable in the files:

portfolio.json

- **id** (string) - offer id
- **offer_type** (string) - type of offer ie BOGO, discount, informational
- **difficulty** (int) - minimum required spend to complete an offer
- **reward** (int) - reward given for completing an offer
- **duration** (int) - time for offer to be open, in days
- **channels** (list of strings)

profile.json

- **age** (int) - age of the customer
- **became_member_on** (int) - date when customer created an app account
- **gender** (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- **id** (str) - customer id
- **income** (float) - customer's income

transcript.json

- **event** (str) - record description (ie transaction, offer received, offer viewed, etc.)
- **person** (str) - customer id
- **time** (int) - time in hours since start of test. The data begins at time t=0
- **value** - (dict of strings) - either an offer id or transaction amount depending on the record

Solution Statement

We will build a **classification model** to predict the completion probabilities of customer offers. Since the transaction data is only available in raw form, a main challenge is to **create the target variable** for the classification model in the first place. To do this, we will look at each offer roll

out to see if a redemption has occurred within the intended validity of the offer. We will then **prepare customer and offer characteristics** as possible model features for each of these observations. We will do a rough **pre-screening of different classification models** (Logistic Regression, KNN and Random Forest) and select the best model for final training and **deployment on Amazon SageMaker**. Finally, we will demonstrate how other teams might use the provided endpoint for building a **recommendation system** on top of it that selects the best offer for a customer.

Benchmark Model

The benchmark model for the recommender system is **a model that randomly chooses an offer for each customer**. In order to be successful, the completion rate of our recommender system must be higher than that of the random model.

Evaluation Metrics

For the **model selection**, we will compare classifiers on the basis of the **AUC (Area Under Curve)** of the ROC Curve. We want to achieve the **best possible discrimination between true positives and false positives** for our recommendation system. The better the classifier is able to discriminate, the better the decisions of the recommendation system will be. Therefore, the AUC metric is a suitable evaluation metric for model selection, because it represents exactly this quality of a classifier.

For the final **recommendations**, we can choose the **completion rate** as evaluation metric. We will simulate the completion rate on the test set if we choose one offer per customer, both with our own recommendation system and the random benchmark model.

Project Design

We will proceed in **three steps**:

1. **Data Exploration And Preprocessing**

First, we will **sift through the three input data sets** and prepare them so that they can be used for model training. However, before we start with the actual model training, we will **investigate correlations** between the features and the completion probability of offers.

2. **Model Comparison**

With this basis, we will **compare three different classification models** (Logistic Regression, KNN, Random Forest) in the second step. The goal here is not to train a final model with optimal hyperparameters, but to make a general decision about which algorithm to choose.

3. **Model Deployment And Application**

In the last step, we will **train the final model pipeline** and **deploy it on Amazon SageMaker**. We will use the provided endpoint to illustrate how the predictions can be used for a **recommendation system** and how its recommendations perform compared to the random model.