Task 1: Cook Prediction

Predict given a (user, recipe) pair from 'stub Made.txt' whether the user would make a recipe.

METHOD/APPROACH

For this task, I implemented the Collaborative Filtering, specifically the Alternating Least Squares (ALS) to compute how compatible a user is to recipe, then using this score to decide whether the user will cook the recipe or not.

Step 1 - Data Import & Preparation

I started by using panda to fetch the csv file into a dataframe with 4 columns: user_id, recipe_id, rating, timestamp. Split data into 2 sets: training and validation

Optional, we compute a numeric code for user_id and recipe_id just to make it our model run faster. Using *cat.codes*, we can easily retrieve back to our original id later.

```
data['user'] = data['user_id'].astype("category")
data['userID'] = data['user'].cat.codes
```

Step 2 - Build a Matrix and Fit the Model

I have an original matrix R of size *u x i* with our users, recipes, and rating. To find a way to turn that into one matrix with users and hidden features of size *u x k* and one with items and hidden features of size *k x i*, I calculate U and V so that their product approximates R as closely as possible: R ≈ U x V. By randomly assigning the values in U and V and using least squares iteratively, I can get at what weights yield the best approximation of R. This approach is similar to the one outlined in [Collaborative Filtering]() by Hu, Koren and and Volinsky

$$\underbrace{\begin{bmatrix} \quad \mathbf{R} \quad \end{bmatrix}}_{|U|\times|I|} = \underbrace{\begin{bmatrix} \gamma_U \end{bmatrix}}_{|U|\times K} \times \underbrace{\begin{bmatrix} \quad \gamma_I^T \quad \end{bmatrix}}_{K\times|I|}.$$

For the score calculated the preference and confidence score using these formulas:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \qquad\qquad c_{ui} = 1 + \alpha r_{ui}$$

The rate of which our confidence increases is set through a linear scaling factor α (here I set α = 40).

To calculate the compatible score, I take the dot product between the user vector and the transpose of the recipe vector.

$$Score = U_i \cdot V^T$$

Here is my flow to the program which also handle the cold-start problem (new user and recipes that have not been seen in training):

```
   Compute a list of the 60% most popular recipes
        If  u is new user:
            Predict 1 if recipe is in the popular list
            Predict 0 otherwise
        If  u is an existing user:
            If r is existing recipe:
                  Calculate the compatible score using the formula and predict
                  1 if score > 0.6, predict 0 otherwise
            Else predict 1 if  the number of recipes that  user has cooked >
1500, predict 0 otherwise
```

Step 3 - Validation

Test the model on validation test and use the AUC to calculate model accuracy