Capstone 3: Recipe Recommendation System

Executive Summary

1. A recommendation system was created to improve user satisfaction and increase user interactions with a website containing recipes

2. Several model approaches were attempted, but ultimately a neural network with separate retrieval and ranking models was used

3. The model performed well based on RMSE compared to a baseline calculation. This is promising for an initial model

4. The models should be refined based on additional user interaction data after deployment

Problem Statement and Approach

Problem Statement: *How can a website which generates user activity by providing access to recipes increase user satisfaction and user engagement through the implementation of a recommendation system that offers personalized recipe recommendations to its users?*

Approach:

Problem Identification

Recommendation systems are a proven method to increase user engagement and satisfaction. I seek to develop a personalized recommendation system for a website which posts recipes to promote user engagement / satisfaction and ultimately increase website revenue.

Data Wrangling

Obtained and cleaned data from the following sources:

- A compendium of 231,637 recipes and associated information such as minutes, steps, nutrition, ingredients, etc
- A log of 1,132,367 user interactions, ratings, and reviews of the recipes contained within the compendium

Exploratory Data Analysis

Gathered insights about the datasets

Key considerations:

- Over 70% of ratings are 5 stars
- Majority of recipes only have 1 or 2 reviews
- Over 73% of users only have written one review
- There are 226,570 unique users
- Sparcity = 99.9978%

Preprocessing & Training

Created two neural network models to function as a single recommendation system:

- **Retrieval** – inputs a user id and outputs a short list of relevant recipes which the user has not previously interacted with
- **Ranking** – inputs a user id and short list of relevant recipes to predict recipe ratings for the user

Modeling

Models were trained separately using different features. They were evaluated using separate metrics and were separately tuned with various hyperparameters. Final model evaluation produced a RMSE of 1.22 compared to a baseline RMSE of 1.39.

Additional user interaction data should be used to train new model and refine recommendation outputs

Data Wrangling

Obtained and cleaned data from the following datasets:

1. **Recipe compendium (231,637 recipes):**
   - Each recipe contained the following features: name, id, minutes, contributor id, date submitted, tags, nutrition, n_steps, description, ingredients, n_ingredients

2. **User ratings and reviews (1,132,367 interactions):**
   - Each interaction contained the following features: user_id, recipe_id, date, rating, review (169 interactions did not contain any review text)

The following actions were taken to clean the data:

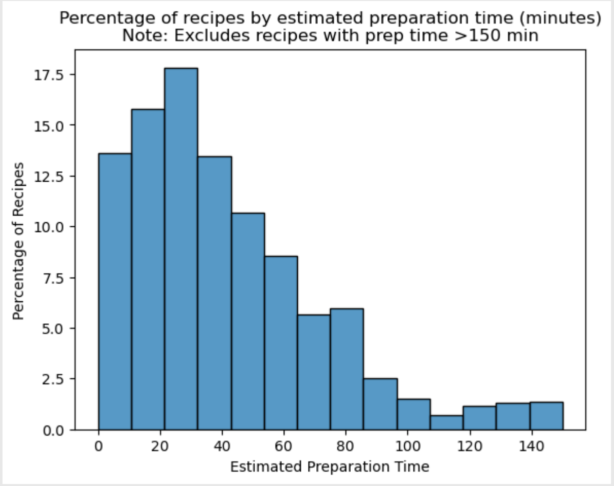
- Manually filled in null values for recipe names
- Null values for open text fields such as reviews were left alone
- Imputed null values for n_steps with the mean value of similar recipes
- Modified outlier values for minutes

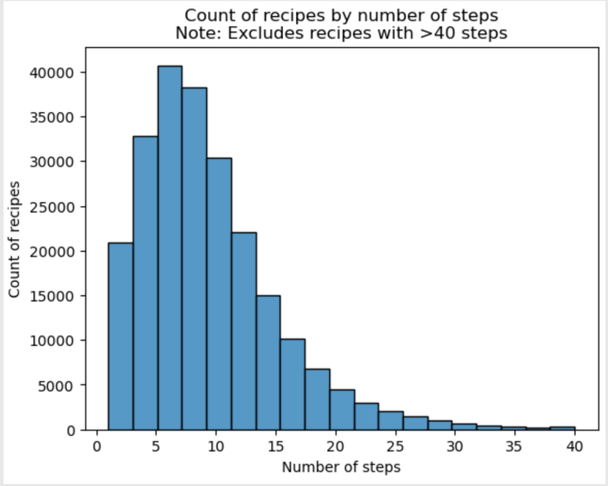
EDA: Recipe Dataframe

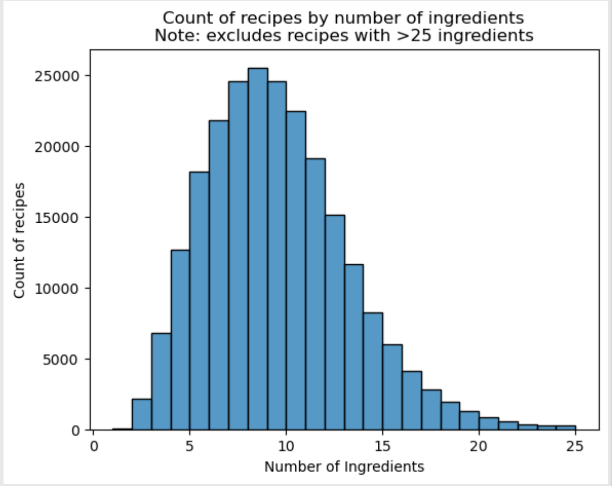
Recipe Dataframe Summary Statistics for numerical features:

	count	mean	std	min	25%	50%	75%	max
minutes	231637.0	9.398546e+03	4.461963e+06	0.0	20.0	40.0	65.0	2.147484e+09
n_steps	231637.0	9.765499e+00	5.995128e+00	0.0	6.0	9.0	12.0	1.450000e+02
n_ingredients	231637.0	9.051153e+00	3.734796e+00	1.0	6.0	9.0	11.0	4.300000e+01

Histograms of recipe data set







- These histograms give us a sense of the overall makeup of the recipe compendium:
  - a majority of recipes are completed in 60 minutes or less
  - most recipes are 10 steps or less
  - most recipes have between 10 – 12 ingredients
- In terms of developing a recommendation system, these insights aren't necessarily directly helpful, but can be used to inform which features should be included in training the model

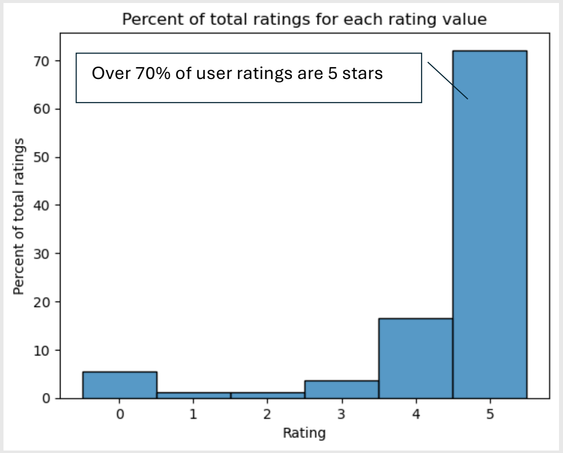
EDA: Interactions Dataframe

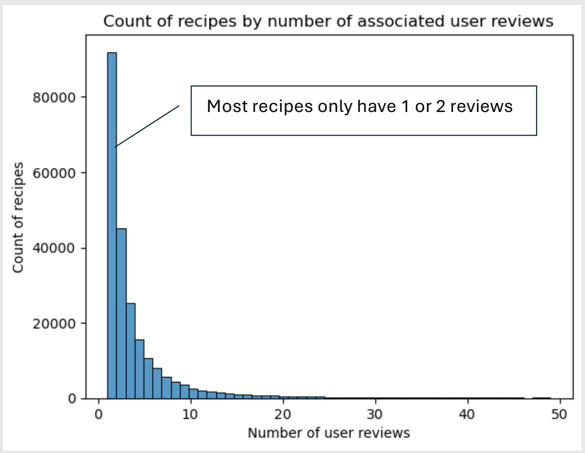
Interactions Dataframe Summary Statistics for numerical features:

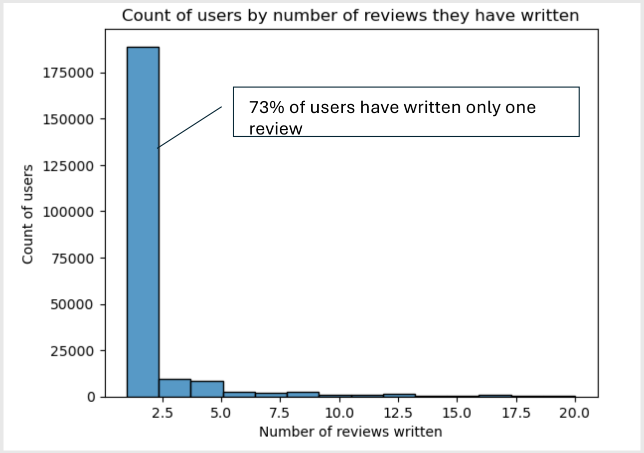
	count	mean	std	min	25%	50%	75%	max
rating	1132367.0	4.411016	1.264752	0.0	4.0	5.0	5.0	5.0

$$Sparcity = 1 - \frac{Number\ of\ Interactions}{Total\ Possible\ Interactions} = 1 - \frac{length\ of\ interactions\ df}{n_{users} * n_{recipes}} = 99.9978\%$$

Histograms of Interactions data set







- These histograms give us a sense of the nature of the user ratings of the recipes:
  - Over 70% of ratings are 5 stars, ratings are skewed towards the higher end of the spectrum
  - The vast majority of recipes have less than 5 reviews while very few have large amounts (the max number of reviews for 1 recipe is >1600 reviews)
  - 73% of users have only written 1 review, the max number of reviews written by one user is 7671 reviews

- In terms of developing a recommendation system it seems as if there is more information / usable features than there are regarding users. That, along with the high level of sparcity, **an item-based collaborative filter recommendation system seems to be the most promising approach.**

Model Build – Approach 1: Item-based collaborative filter

Initial approach involved building an item-based collaborative filter

1. **Item – based collaborative filter was chosen for the following reason:**
    - more robust recipe dataset will make it more feasible to identify similar recipes vs similar users
    - better recommendations could be produced even for a cold-start user
    - can be implemented well even with a highly sparse dataset

2. **I intended to execute the following plan:**
    - Construct a user-recipe interaction table
    - Calculate item similarities (e.g., cosine similarity, pearson correlation, or implement a KNN model)
    - Calculate predicted ratings of recipes
    - Generate recommendations - select the highest predicted ratings for the recipes for the user

Due to memory constraints, a user-recipe interaction was not able to be constructed on my machine despite mitigation efforts. A new approach needed to be implemented.

Model Build – Approach 2: Neural Network Based 2-stage System (1/2)

A new approach was needed to deal with memory constraints, yet still produce quality recommendations

A two stage neural network based model will be built to provide personalized user recipe recommendations:

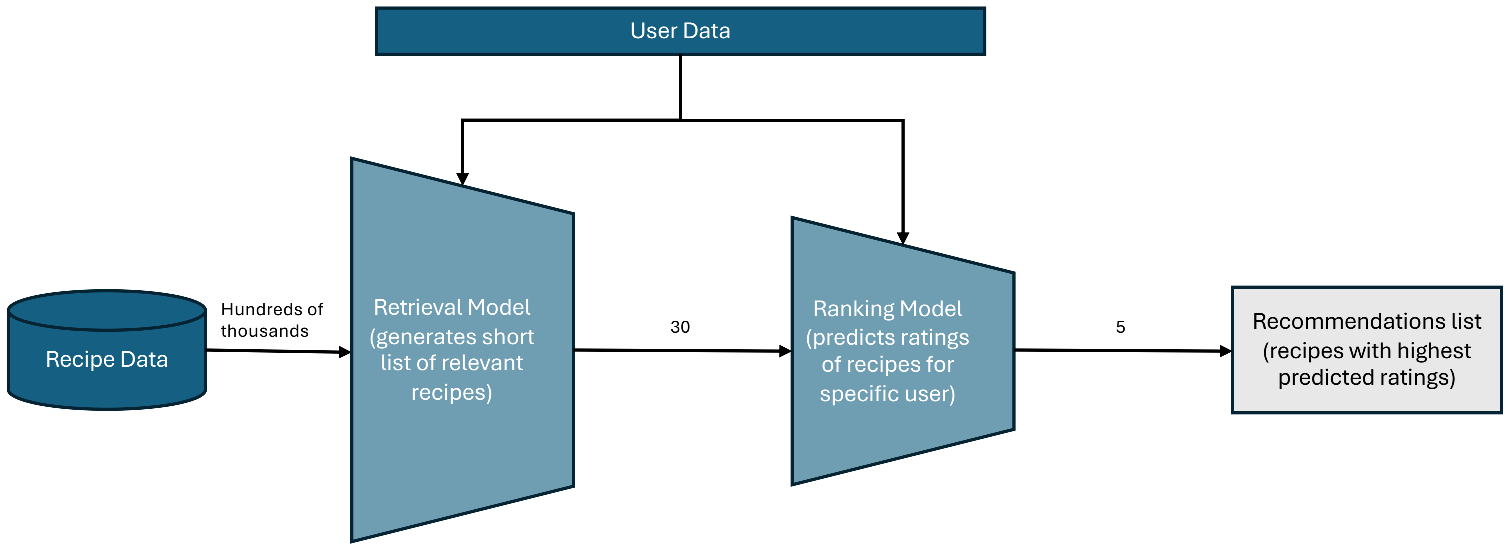
1. **A two-stage neural network based recommendation system was chosen for the following reasons:**
   - Handles large datasets efficiently
   - Handles high levels of dimensionality well
   - Easier customization

2. **I intended to execute the following plan:**
   - **Preprocess data:** convert categorical features in both recipe and user dfs into numerical values using tokenization and embeddings. I will also convert the datasets into TensorFlow datasets.
   - **Split the data:** split the data into training and test sets
   - **Define and create the model architecture:** Create input layers for both user interactions and recipes, define how many hidden layers we want, which activation function to use, the loss function to optimize, which optimizer we want to use
   - **Train the model:** Train the model on the training and validation data
   - **Evaluate the model:** Evaluate the model using the test set using a pre-defined evaluation metric such as RMSE
   - **Optimize the model:** Tune the parameters to improve evaluation metrics
   - **Deploy the model:** Deploy the model to be able to recommend recipes to new users

Model Build – Approach 2: Neural Network Based 2-stage System (2/2)

A new approach was needed to deal with memory constraints, yet still produce quality recommendations

A two-stage neural network-based recommendation system works in the following manner:


User Data
Recipe Data
Hundreds of thousands
Retrieval Model (generates short list of relevant recipes)
30
Ranking Model (predicts ratings of recipes for specific user)
5
Recommendations list (recipes with highest predicted ratings)

The following parameters were used in the neural network models:

Model	Features	Dense layers (# nodes)	Epochs	Loss Function	Optimizer	Learning Rate	Activation Function	Evaluation Results
Retrieval	User_id, recipe_id	2 embedding layers	5	Categorical Entropy	Adagrad	0.1	N/A	Top K 50 = 0.0126
Ranking	user_id, recipe_id, rating	1 (256), 2 (64)	5	MSE	Adagrad	0.1	Rectified Linear Unit (Relu)	RMSE = 1.22 (baseline RMSE = 1.39)

Next Steps

1. Deploy the model – this model could be deployed easily. A new user could answer a few questions regarding their recipe preferences (e.g., choose which recipe most appeals to you) to use as a baseline input. That user id and recipe preferences could be fed into the retrieval and ranking models to produce a list of 5 recommended recipes.

2. Track user behavior / interaction – an experiment should be run to see if / how the recommendation system changes user interactions. Does it have a positive impact? Are users more likely to interact with the website of recipes if they have these recommendations presented to them? These experiments can help refine the model. Key indicators could be engagement metrics like clicks or recipe saves and how they change with recommendations

3. Refine the model as more user interaction data is obtained – there is much room for improvement on the model. We can include more features to train the ranking model. We can refine the model, deploy new models, and run experiments to determine which model is most effective at improving user interactions. Incorporating additional features like tags, review texts, nutrition may help greatly improve the model. However, this would take computational time and power.

Questions?

Model Builds and Results

The following neural networks were built:

#	Name	Features	Dense layers (# nodes)	Epochs	Loss Function	Optimizer	Learning Rate	Beta1	Evaluation Results
1	Model1	All	1 (64)	10	MSE	Adam	0.001	0.9	RMSE = 6.5
2	Model2	All	1 (64), 2 (32)	10	MSE	Adam	0.001	0.9	RMSE = 38.7
3	Model3	All	1 (64)	10	MSE	Adam	0.0001	0.5	RMSE = 8.42
4	Model Crossentropy1	All	1 (64)	3	Sparse Categorical Crossentropy	Adam	0.0001	0.5	Accuracy = 71%
5	Model Crossentropy2	All	1 (64)	3	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 69%
6	Model Crossentropy3	All	1 (64), 2 (64)	3	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 69%
7	Model Crossentropy4	All	1 (64), 2 (64)	5	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 68%
8	Model Crossentropy5	Excludes tags	1 (64), 2 (64)	5	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 70%
9	Model Crossentropy6	Excludes nutrition	1 (64), 2 (64)	3	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 66%
10	Model Crossentropy7	Excludes user, recipe, contributor	1 (64), 2 (64)	3	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 72%
11	Model Crossentropy8	Excludes user, recipe, contributor	1 (64), 2 (64)	10	Sparse Categorical Crossentropy	Adam	0.001	0.9	Accuracy = 72%
12	Model Crossentropy9	Excludes user, recipe, contributor	1 (64), 2 (64)	10	Sparse Categorical Crossentropy	Adam	0.0001	0.5	Accuracy = 72%
13	Binary crossentropy1	All	1 (64), 2 (64)	3	Binary Crossentropy	Adam	0.001	0.9	Accuracy = 71%
14	Two Stage	User Id, Recipe ID	1 (256), 2 (64)	5	Categorical Crossentropy	Adagrad	0.1	N/A	RSME = 1.22