Sake Recommendation System

Objective

The objective of this project is to build a recommendation system for sake bottles. The system takes as input the names of the sakes that a user likes and dislikes, and outputs a prediction of a sake that the user will like, along with recommendations for similar sakes.

Methodology

The recommendation system uses a combination of Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN) algorithms. The SVM model is used to predict a sake that the user will like based on the user's input, and the k-NN model is used to find similar sakes to the predicted sake.

The system also uses a weighting scheme to give more importance to features that are common in sakes that the user likes and less importance to features that are common in sakes that the user dislikes. The weights are calculated based on the mean value of each feature for the sakes that the user likes and dislikes. A small constant (1e-10) is added to the weights to avoid division by zero.

The features are then scaled by these weights, so that features that are more important (i.e., have higher weights) have a greater influence on the distance calculation in the k-NN algorithm.

Benefits

This recommendation system provides a personalized experience for users, as it takes into account the user's individual preferences when making recommendations. It also provides a diverse set of recommendations, as it uses a k-NN model to find similar sakes to the predicted sake.

Moreover, the use of a weighting scheme allows the system to give more importance to features that are more relevant to the user's preferences, and less importance to features that are less relevant. This makes the recommendations more tailored to the user's individual tastes.

Code Explanation

The code first loads and preprocesses the dataset, which includes converting price and SMV to float, one-hot encoding categorical columns, filling missing values with the mean of the column, and encoding the 'Name' column.

The dataset is then split into training and testing sets, and the SVM model is trained with a linear kernel and C=1.0.

The system then takes as input the names of the sakes that the user likes and dislikes. It calculates weights for each feature based on these inputs, scales the features by these weights, and fits a k-NN model to the scaled data.

The SVM model is then used to predict a sake that the user will like based on the user's input. The k-NN model is used to find the 5 sakes that are most similar to the predicted sake. These are the sakes that the system recommends to the user.

Code Output

```
Enter the names of the sakes you like, separated by commas: WAKAZE “Yuzu”,Okunomatsu “Ihei”
Enter the names of the sakes you don't like, separated by commas: Suehiro “Gensai”,Rihaku “Dreamy Clouds”
Enter the name of the sake: Narutotai “Ginjo” Nama Genshu
Predicted Sake: Narutotai “Ginjo” Nama Genshu
Recommended Sakes:
Narutotai “Ginjo” Nama Genshu
Kamotsuru “Sokaku”
Momokawa “Diamond”
Kikumasamune “Shiboritate” Kojo
Kamikokoro “Umakuchi” Hiyaoroshi
```

Example

- The model first takes the names of the sakes that the user likes and dislikes as input. It then calculates weights for each feature based on these inputs. The weights are higher for features that are common in sakes that the user likes and lower for features that are common in sakes that the user dislikes.
- The model then scales the features by these weights. This means that features that are more important (i.e., have higher weights) will have a greater influence on the distance calculation in the k-NN algorithm.
- The SVM model is trained to predict the sake based on the user's input. The predicted sake is the one that the SVM model thinks the user will like the most based on the user's input.
- The k-NN model then finds the 5 sakes that are most similar to the predicted sake. These are the sakes that have the most similar features to the predicted sake, taking into account the weights that were calculated earlier. These are the sakes that the model recommends to the user.
- The output of the model is the name of the predicted sake and the names of the recommended sakes, all in string format.

Simple explanation of how the weights and the k-NN algorithm and SVM model Works

1. **User Input**: Let's say we input that we like "Sake A" and dislike "Sake B". we also input that we are interested in "Sake C".

2. **Weight Calculation**: The model first calculates weights for each feature (like price, brewery, acidity, etc.) based on our likes and dislikes. For example, if "Sake A" (which we like) is expensive and "Sake B" (which we dislike) is cheap, the 'Price' feature might get a high weight. This means the model will pay more attention to the price when making predictions.

3. **SVM Prediction**: The SVM model then uses these weights to predict a sake that we would like, based on the features of "Sake C" (the sake we interested in). Let's say it predicts "Sake D". This means that "Sake D" has similar features to "Sake C" and aligns with our preferences (it's more like "Sake A" and less like "Sake B").

4. **k-NN Recommendation**: Now, the k-NN model takes over. It uses "Sake D" (the sake predicted by the SVM model) as a reference and finds other sakes that are similar to "Sake D". Let's say it finds "Sake E", "Sake F", and "Sake G". These sakes are similar to "Sake D" (and therefore also similar to "Sake C") in terms of their features.

5. **Output**: The model then outputs "Sake D" as the predicted sake (the sake it thinks you would like based on your input) and "Sake E", "Sake F", and "Sake G" as the recommended sakes (the sakes that are similar to the predicted sake).

Conclusion

This sake recommendation system provides a personalized and diverse set of recommendations for users, taking into account their individual preferences. The use of SVM and k-NN algorithms, along with a weighting scheme, allows the system to make accurate and relevant recommendations.