Predicting Cuisines of Recipes

Francesco Ercole
s2135797
s2135797@ed.ac.uk

Dan Lyth
s2008827
s2008827@ed.ac.uk

Mariel Reyes Salazar
s2123659
s2123659@ed.ac.uk

Haorui Tao
s2039924
s2039924@ed.ac.uk

Abstract

We explore a dataset of recipes that we analyze, visualize and pre-process before utilizing in the context of two different predictive tasks. We also explore a variety of dimensionality reduction techniques to help visualize relationships within the data. The first of the predictive tasks is to classify the cuisine of a recipe given its ingredients. For this task, we explore the use of both the full feature representation of the dataset and two different representations that have been dimensionally reduced. The second task is to explore collaborative filtering, specifically, to suggest suitable ingredients for a recipe that has had some of its ingredients removed. For both tasks we explore a variety of methods and demonstrate their efficacy.

1 Introduction

Meat patty, chips, cheddar cheese, tomato and a bread bun. What is it? It is most likely a cheeseburger. Where is this recipe from? America, most likely. Humans are able to tell which cuisine a dish may come from using "cues" from the ingredients or by visual inspection.

Can a machine be able to replicate this inference? Cuisine is a strong identity factor that makes people in a certain region connect and share traditions. Each culture has their own style of cooking that is distinctive to others by the usage of different ingredients and cooking techniques.

In this work, we use the dataset from Bellosi [2] to predict the cuisines of recipes. This dataset includes 4236 recipes from 12 cuisines with 709 distinct ingredients. The objectives for this project are:

- Predict the cuisine type when given a list of ingredients in a recipe
- Collaborative filtering to predict ingredients in a partial recipe

Cuisine prediction is not a new task. Several authors built different models and also there was a competition in the Kaggle site [9]. Kumar et al. [11] used Extreme Boosting Algorithm and Random forest, achieving an accuracy of 0.80 for that dataset. Sharma et al. [15] instead applied several machine learning algorithms, i.e Naive Bayes, Logistic Regression and Support Vector Machines.

Regarding the recommendation system, and Ueda et al. [18] proposed a personalized method based on user's food preferences, Teng et al. [17] developed recipe recommendation methods by constructing 2 ingredient networks. Wang et al. [19] model cooking procedures of Chinese recipes as directed graphs and proposed a substructure similarity measurement based on the frequent graph mining.

All those papers, although referring to a different dataset, are useful to know the state-of-art and its results can be used as a benchmark.

2 Data preparation

Prior to exploring data, it was required to prepare the dataset to ensure it is in the appropriate format to work with it.

1. Rename ingredients columns since many ingredients had whitespaces, e.g. soy sauce. Whitespaces can cause trouble when handling these columns, and therefore all whitespaces in ingredient columns were replaced by an underscore

2. Split data into training set (for exploration and model building), validation set (for adjusting model hyperparameters) and finally a test set for evaluating the generated models using a new dataset. More details are described in section 5.1

3. Translate by one unit the cuisine type identifier for consistency across the recipe and the cuisine datasets.

3 Exploratory data analysis

To uncover initial patterns in the data, we used built-in libraries [7, 20, 22] and developed functions for exploration. The exploration was done in the training set.

3.1 Data description

The dataset contains only categorical variables with the ingredient used in a recipe. Each observation is a recipe, and the last column describes the cuisine in which it belongs. The ingredients variables are binary, describing the presence or absence of it in a recipe.

The matrix sparsity is defined as the number of non-zero elements divided by the total number of elements in the matrix. Considering a matrix A has dimensions $m \times n$; then the sparsity of A:

$$\text{sparsity} := 1 - \frac{\#\{a_{ij} : a_{ij} \neq 0\}}{n.m}$$

Due of the high number of ingredients, the data has a sparsity coefficient of 98.5%.

3.2 Ingredient usage

The top 5 ingredients used in all cuisines are garlic, onion, salt, olive oil and chicken. These ingredients appear in average in 1191 recipes (with a median of 948 recipes). Salt is a popular ingredient; however, this ingredient may not be relevant for predicting a recipe, as salt can be added up to taste for most of the recipes, except for sweet dishes or desserts.

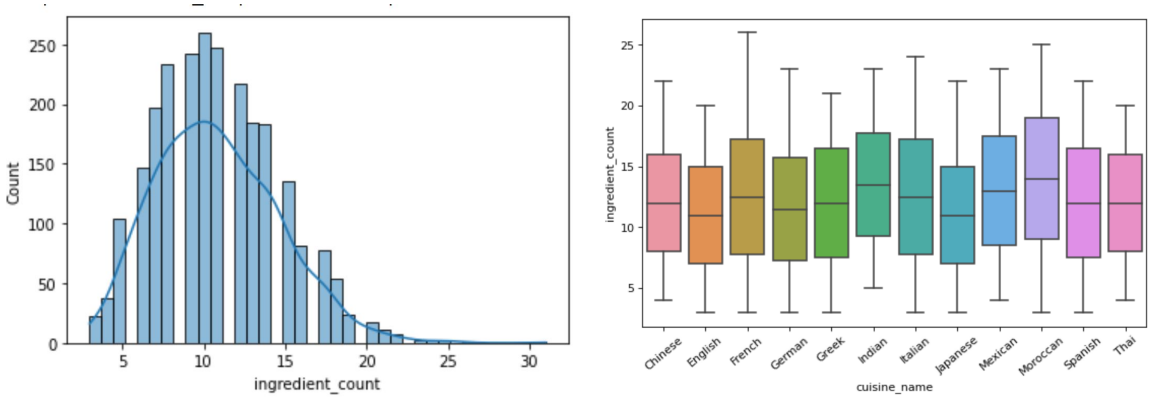
Recipes are balanced across the dataset for each cuisine type. The median number of ingredients per recipe are relatively similar in all cuisine types, with a median value of 12 ingredients per recipe. However, there are examples of recipes that require a higher number of ingredients. In particular, there is one recipe in French cuisine that uses 26 ingredients. This can be seen in figure 1.

Indian cuisine uses many spices and German cuisine relies on pork products, and this is extended to all cuisines. Cuisine is a cultural trait and as such, it has its own identity. Therefore, it is expected that the most popular ingredients per cuisine will differ. This is confirmed by exploring the 5 most popular ingredients per cuisine. The top 5 ingredients are listed in table 1.

Table 1: Top 5 most popular ingredients per cuisine

Chinese	English	French	German	Greek	Indian
Soy sauce	Onion	Garlic	Onion	Olive oil	Onion
Garlic	Butter	Butter	Pepper	Garlic	Garlic
Ginger	Potato	Wine	Salt	Onion	Ginger
Cornstarch	Garlic	Onion	Flour	Oregano	Cumin
Green onion	Flour	Olive oil	Water	Tomato	Turmeric

Italian	Japanese	Mexican	Moroccan	Spanish	Thai
Garlic	Soy sauce	Onion	Onion	Garlic	Garlic
Olive oil	Rice wine	Tortilla	Garlic	Olive oil	Fish sauce
Parmesan cheese	Sugar	Garlic	Olive oil	Onion	Chicken
Onion	Ginger	Cumin	Cumin	Tomato	Coconut cream/milk
Pasta	Garlic	Salt	Cinnamon	Chicken	Lime



(a) Ingredients per recipe for all cuisines histogram

(b) Ingredients per cuisine boxplot

Figure 1: Ingredients per recipe analysis

Part of the exploration is to understand how ingredients are used in different cuisines. There are specific ingredients that are only present in certain cuisines. For instance, pecorino cheese is only in Italian recipes. The number of specific ingredients per cuisine are displayed in the heatmap in figure 2.

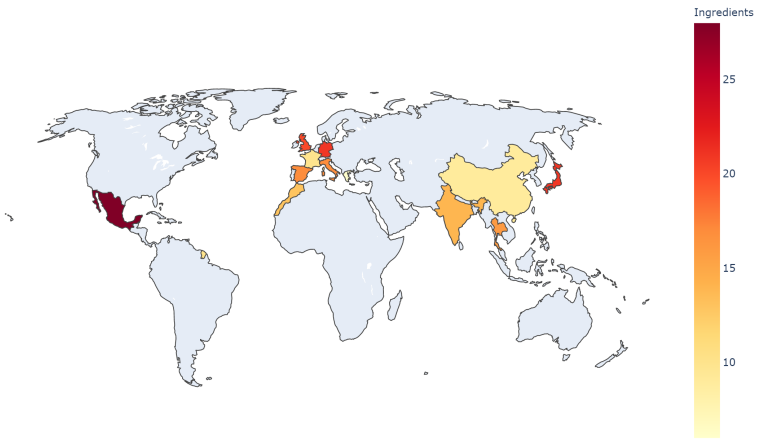


Figure 2: Number of specific ingredients per cuisine

In addition to these findings, it was found there are several ingredients that appear only in one recipe or in two. These ingredients may add noise to the data, as they do not seem to be meaningful in describing a recipe or predicting the cuisine.

Furthermore, it was analyzed the most common matches between ingredients using the `best_match` function developed for this project. For each cuisine only the elements that appear in at least 3% of the recipes of the same group are taken into consideration. This percentage is a parameter that can be tuned for removing infrequent ingredients. Using this function, it is possible to distinguish famous combinations of ingredients, such as cod, flour and potato in the English cuisine for the preparation of 'fish and chips' and tortilla with cheese in the Mexican cuisine.

4 Dimensionality reduction

Considering this is a high-dimension dataset, it is crucial to reduce the dimensions for a better understanding in 2d or 3d plots, identify patterns or clusters and use this information for building models.

Prior to running any dimensionality reduction technique, the least popular ingredients were removed. The least popularity was defined as an ingredient that appears in only 2 recipes, at most. This reduces the dataset to 538 ingredients.

The explored techniques were PCA, kernel PCA, Isomap, metric MDS, t-SNE and UMAP; however only UMAP provided more insight for this particular dataset. The results of UMAP are detailed in section 4.1. The results from all the techniques explored can be found in appendix C.

4.1 UMAP

Unifold Manifold Approximation and Projection (UMAP) [12] technique was explored. UMAP is a nearest-neighbor dimensionality reduction method.

Different metrics to compute distances between neighbors (Jaccard, Dice, Rogers & Tanimoto, Yule, Bray & Curtis, Cosine) and number of neighbours (5, 10, 15, 20, 30, 50, 100) hyperparameters were trialed. Jaccard, Dice, cosine and Bray & Curtis metrics show similar patterns with the same number of neighbors. This is due of such metrics does not take in account the negative matches [4], and this is relevant in present of sparse matrices. Instead, Rogers & Tanimoto and Yule metrics consider the negative matches, and this is reflected in their results of UMAP reduction. All UMAP reduction plots can be found in appendix C.6. A UMAP representation is obtained using Jaccard metric with 15 n-neighbors in figure 3.

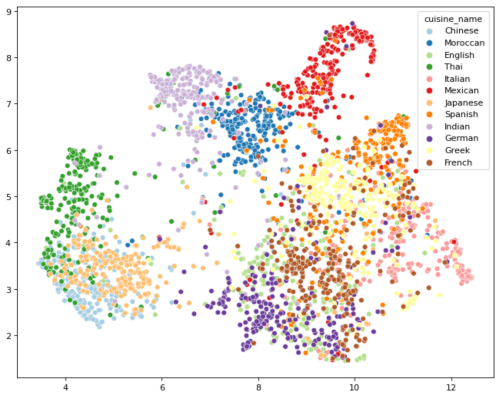


Figure 3: UMAP reduction using Jaccard metric and 15 n-neighbors

Using the reduction shown in figure 3, it can be inferred the dataset has 3 major clusters. A cluster comprising East Asia cuisines (Chinese, Japanese and Thai) on the left side of the plot, European cuisines (English, French, German, Greek, Italian and Spanish), and spiced cuisine types in the top center (Indian, Moroccan and Mexican).

5 Classification Task Methodology

We now explore a variety of approaches to the task of classifying the cuisine of a recipe given its ingredients. As seen in the exploratory data visualizations in section 3, it would appear that the different cuisines should indeed be separable given their list of ingredients. We train machine learning models on both the full feature representation and two dimensionally reduced feature representations and investigate the effect these different representations have on classification accuracy. In some real-world applications, the volume of data collected for inference is extremely large. Thus, exploring whether data can be stored in a lower-dimensional representation and still effectively be used for inference is a worthy area of research.

5.1 Data Preparation

We split the data into training, validation and test subsets using 70%, 20% and 10% of the dataset respectively. When creating these splits, we ensure that the distribution of the labels in the full dataset is reflected in each subset. For each model, the training dataset is used for training and fitting parameters, the validation dataset is used to tune the hyperparameters, and the test set is used to evaluate the generalization performance. Another common approach to creating a validation dataset for tuning hyperparameters is to use k-fold cross-validation, a method most commonly used with smaller datasets. While the number of observations in this dataset (4236) is not especially large, we found that this approach did not yield significantly improved performance over the training and validation split method described above.

For training and testing these models using dimensionally reduced forms of the dataset, we take two approaches. The first uses PCA with 71 components; 10% of the number of features compared to the original dataset. When applied to the training data, these components contain 73% of the explained variance. The second uses UMAP with only two components; 0.3% of the number of features compared to the original dataset.

5.2 Classification Models

We use scikit-learn [14] and TensorFlow [1] as machine learning frameworks for our prediction task. We explore the use of logistic regression, random forests, k-nearest neighbors, a support vector machine and a simple, shallow neural network. Due to the balanced nature of the classes, we use accuracy as our performance metric.

It is briefly worth discussing why we choose to include a neural network as one of our classifiers. The widespread adoption of neural networks for classification tasks has been driven by their performance on unstructured data such as images and audio. This is predominantly because deep neural networks are able to learn feature representations of this unstructured (and often highly correlated) data and effectively use these representations for classification. However, structured data such as the kind that we are exploring in this report typically benefits less from these learned representations as the data already has clearly defined and separable features.

Despite this fact, we wanted to experiment with using a simple neural network and compare its performance to the other classical machine learning methods. In particular, we explore whether adding more hidden layers to such a network - a technique used to help learn features in unstructured data - has any positive effect when modeling structured data.

6 Classification Task Experiments

Given our data analysis and visualization, we hypothesize that this dataset should be able to make reasonably accurate predictions using straightforward discriminative classifiers. We are less confident of how these models will perform on the dimensionally reduced data. However, the number of components and explained variance of the PCA reduced data is relatively high and given the clearly visualized clusters in the two-dimensional UMAP data, we do hypothesize that even with reduced dimensionality, we should be able to make reasonably effective predictions.

The generalization abilities of these models are tested using the held-out test set and the results can be seen in Table 3.

Model	Accuracy (%)	Accuracy - PCA (%)	Accuracy - UMAP (%)
Logistic Regression	76.7	71.2	57.5
Random Forest	73.1	**72.8**	59.7
K-Nearest Neighbors	61.6	61.3	**60.4**
Support Vector Machine	74.5	71.0	59.9
Shallow neural network	**79.2**	71.5	59.0

Table 2: Recipe classification results using a variety of machine learning methods on both the regular test set and dimensionally reduced test sets.

6.1 Logistic Regression

After some hyperparameter tuning and experimentation, we find that using L2 regularization with a C value of 0.8 and 300 maximum iterations produces the best result on the full training and validation sets. We then train models on the PCA reduced data (C value of 0.4) and the UMAP reduced data (C value of 0.5).

6.2 Random Forest

We find that using 300 estimators, a maximum depth of 50 and a minimum sample split of 3 yields good results on the validation set. For the PCA reduced data we use the same number of estimators, a maximum depth of 30 and a minimum sample split of 5. For the UMAP reduced data, we again use the same number of estimators, a maximum depth of 10 and a minimum sample split of 10.

6.3 K-Nearest Neighbors

We use the ball tree algorithm [13] for all three datasets and only modify the number of nearest neighbors; 12 for the full dataset, 15 for the PCA dataset, and 20 for the UMAP dataset. The results can be seen in Table 3. For the first time, we see very similar performance across all three data representations.

6.4 Support Vector Machine

After experimenting with various hyperparameters we find that scikit-learn's default support vector machine hyperparameters are suitable for this task.

6.5 Shallow Neural Network

We use Tensor Flow and the Keras API [5] to build a shallow neural network. This network has just one hidden layer consisting of 128 units followed by a dropout layer [16], followed by the final output layer. The hidden layer and output layer have an L2 kernel regularizer applied with a regularization value of 1e-5 and the dropout layer uses a dropout value of 0.5. The model is optimized using cross-entropy loss and the Adam optimizer. Early stopping is applied to help prevent over-fitting.

We also experiment with deeper networks with several more hidden layers and larger numbers of hidden units. However, we found that these extra layers do not build up a richer feature representation, witnessed by equivalent or poorer performance than that of the shallow network.

6.6 Discussion

As we can see in Table 2, the shallow neural network performs best on the dataset containing the full feature representation. Interestingly, with the PCA dataset that contains only 10% as many features, the accuracy of the best model (random forest) is not drastically worse. We see a further performance drop with the UMAP data, which is perhaps unsurprising given that it has only 0.3% as many features as the full dataset (only two dimensions). Looking at the UMAP visualization in Figure 4, while we can see relatively distinct cuisine clusters, there is also considerable overlap between the classes. Interestingly, the k-nearest neighbor model (which performs most strongly here) has very similar results across all three datasets. Depending on the domain and the model performance requirements,

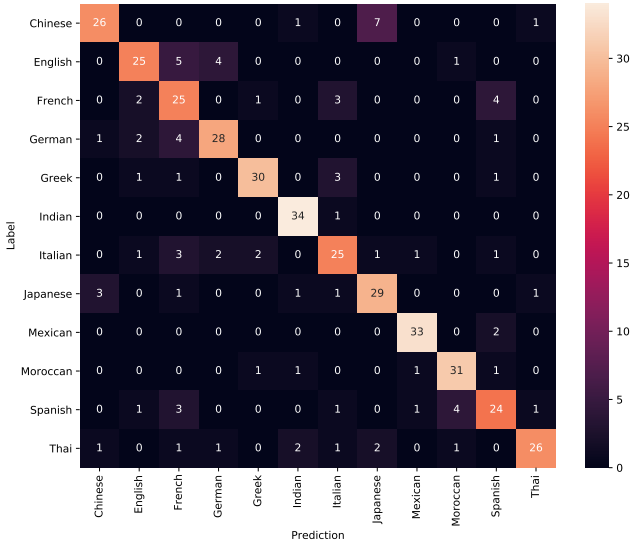


Figure 4: Confusion matrix for the best performing model

it would appear that dimensionality reduction as a method of reducing the storage requirements for huge datasets could be worth exploring.

Looking at the confusion matrix in Figure 4, we can see that the way in which the best-performing model confuses the classes is fairly interpretable and broadly follows global geography. For example, Chinese and Japanese cuisines are occasionally "confused" by the model, likewise French and Spanish. By contrast, Greek cuisine is never confused with Thai, Indian, Chinese or Moroccan cuisines.

7 Collaborative Filtering Methodology

In many domains (especially since the advent of the internet), a user may be faced with numerous options to choose from, and a recommendation system is likely to be valuable. Collaborative filtering (CF) is one such approach to recommendation and is motivated by the desire to recommend items to users effectively. In this report, the "users" are recipes, and the "items" are ingredients. We remove ingredients from a recipe and use collaborative filtering to make recommendations for these missing ingredients.

There are broadly two approaches to collaborative filtering; the first can be described as user-based and the second as item-based. In user-based CF, we group similar users who like the same items. We can then recommend an item to a user based on what other similar users have "liked". In item-based CF, we group similar items and recommend items to a user based on other items the user has "liked".

The feedback that shows whether a user has "liked" an item is either explicit or implicit. Explicit feedback measures how much a user likes an item (how highly it is rated). For example, if a user gives a movie a rating between 0 and 5. Implicit feedback only shows whether a user has used an item, not how highly rated the item is. In our case, we are dealing with implicit feedback as we only have binary labels of whether an ingredient (item) is used or not in a recipe (user).

There are several types of collaborative filtering methods. Broadly, these are memory-based, model-based, hybrid, and, more recently, deep learning-based. Memory-based CF involves computing a similarity metric between neighboring users or items and using this metric to make recommendations. Model-based CF often involves dimensionality reduction (such as singular value decomposition) to create low dimensional latent factors that can then be used to group similar items and users. Hybrid CF uses a combination of these approaches deep-learning CF feed existing methods into non-linear networks or introduce new approaches such as variational autoencoders.

We explore the use of item-based CF with a memory-based model. In this method of CF, the similarity measure used to define the relationship between items is a crucial hyperparameter. We compare the

SIMILARITY MEASURE	RECALL@10 (%)	RECALL@10 - PCA (%)
COSINE SIMILARITY	31.7	37.9
ASYMMETRIC COSINE SIMILARITY	0.0	44.7
JACCARD SIMILARITY	34.8	36.2
POINTWISE MUTUAL INFORMATION	0.0	**48.7**

Table 3: Recall@10 results for different similarity methods on the full dataset and the PCA dataset.

use of several different similarity measures; cosine similarity (CS), asymmetric cosine similarity (ACS), Jaccard similarity (JS), and pointwise mutual information (PMI).

In [3], it is demonstrated that dimensionality reduction techniques can be very effective in CF settings. We apply PCA across "users" (recipes) with the number of components equal to the number of "items" (ingredients). This significantly reduces the dimensionality of the data, and we explore whether it also increases the quality of recommendations compared to the full-dimensional data.

We divide the dataset into a 20%-80% split, with the smaller subset used for selecting the hyperparameters (the "training" dataset) and the larger subset used a test dataset. The proportions of these splits are not typical in machine learning settings. However, in this case, we are not 'training' the model but simply evaluating different metrics across a representative sample of the dataset.

Our metric for evaluation is the percentage of recipes for which the missing ingredient is included in the CF model's top ten recommendations, i.e. recall@10.

8 Collaborative Filtering Experiments

Using the "training" dataset, we perform a grid-search to choose optimal values of the asymmetry parameter alpha for the asymmetric cosine similarity and the number of neighbors k for all similarity measures. We then compare the recall@10 for the different similarity measures using the full-dimensional test dataset and the dimensionally reduced PCA test dataset. The results can be seen in Table 2.

8.1 Discussion

The results for the asymmetric cosine similarity and pointwise mutual information measures on the regular test dataset may be the result of a bug in the code. It is unclear why their recall@10 score is so poor. However, comparing the remaining two similarity measures across the regular and PCA reduced versions of the test set, it is clear that the PCA reduction significantly improves performance.

9 Conclusion

We have demonstrated a variety of data cleaning, analysis, dimensionality reduction, and visualization methods that help to explore and understand this recipes dataset. This exploration fed into the decisions made when designing the experiments for the two predictive tasks. In the classification task, we successfully demonstrate various machine learning methods for predicting the cuisine given the ingredients and also explore the use of dimensionally reduced data in this context. In the collaborative filtering task, we successfully recommend ingredients and note that in this case, the dimensionality reduction was a key element to producing favorable results. An area of future work would be to explore this aspect of dimensionality reduction in the context of collaborative filtering and experiment with other dimensionality reduction techniques. Moreover, the task of classifying cuisine based on ingredients is similar to topic modeling. It would be interesting to explore topic modeling techniques such as Latent Dirichlet allocation to model cuisines.

Contribution statement

- Francesco Ercole:
  - Exploratory data analysis: Best match, specific ingredients,
  - Dimensionality reduction: UMAP metrics comparison, References lookup
- Dan Lyth:
  - Classification Models
  - Collaborative filtering
  - Conclusions
- Mariel Reyes Salazar
  - Exploratory data analysis: Specific ingredients heatmap, top n ingredients, least popular ingredients
  - Dimensionality reduction: All techniques
  - Interim report write-up with preliminary results
- Haorui Tao
  - Related works
  - Further works

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. [2015], 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

[2] Bellosi, F. [2012], Machine learning for cuisine discovery, Master's thesis, University of Edinburgh.

[3] Candillier, L., Jack, K., Fessant, F. and Meyer, F. [n.d.], 'State-of-the-art recommender systems'.

[4] Choi, S., Cha, S.-H. and Tappert, C. [2009], 'A survey of binary similarity and distance measures', *J. Syst. Cybern. Inf.* **8**.

[5] Chollet, F. et al. [2015], 'Keras', `https://keras.io`.

[6] Hug, N. [2020], 'Surprise: A python library for recommender systems', *Journal of Open Source Software* **5**(52), 2174.
URL: *https://doi.org/10.21105/joss.02174*

[7] Hunter, J. D. [2007], 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

[8] Inc., P. T. [2015], 'Collaborative data science'.
URL: *https://plot.ly*

[9] Kaggle [2021], 'What's cooking?',
`https://www.kaggle.com/c/whats-cooking`. (Accessed: 10 April 2021).

[10] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S. and Willing, C. [2016], Jupyter notebooks – a publishing format for reproducible computational workflows, *in* F. Loizides and B. Schmidt, eds, 'Positioning and Power in Academic Publishing: Players, Agents and Agendas', IOS Press, pp. 87 – 90.

[11] Kumar, R., Kumar, M. and Kp, S. [2016], 'Cuisine prediction based on ingredients using tree boosting algorithms', *Indian Journal of Science and Technology* **9**.

[12] McInnes, L., Healy, J. and Melville, J. [2020], 'Umap: Uniform manifold approximation and projection for dimension reduction'.

[13] Omohundro, S. M. [1989], *Five balltree construction algorithms*, International Computer Science Institute Berkeley.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. [2011], 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

[15] Sharma, T., Upadhyay, U. and Bagler, G. [2020], 'Classification of cuisines from sequentially structured recipes'.

[16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. [2014], 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), 1929–1958.

[17] Teng, C.-Y., Lin, Y.-R. and Adamic, L. A. [2012], Recipe recommendation using ingredient networks, *in* 'Proceedings of the 4th Annual ACM Web Science Conference', WebSci '12, Association for Computing Machinery, New York, NY, USA, p. 298–307.
URL: *https://doi.org/10.1145/2380718.2380757*

[18] Ueda, M., Takahata, M. and Nakajima, S. [n.d.], 'User's food preference extraction for personalized cooking recipe recommendation'.

[19] Wang, L., Li, Q., Li, N., Dong, G. and Yang, Y. [2008], Substructure similarity measurement in chinese recipes, ACM, Beijing, China, pp. 979–988.
URL: *https://dl.acm.org/doi/10.1145/1367497.1367629*

[20] Waskom, M. and the seaborn development team [2020], 'mwaskom/seaborn'.
URL: *https://doi.org/10.5281/zenodo.592845*

[21] Wattenberg, M., Viégas, F. and Johnson, I. [2016], 'How to use t-sne effectively', *Distill* . (Accessed: 13 March 2021).
URL: *http://distill.pub/2016/misread-tsne*

[22] Wes McKinney [2010], Data Structures for Statistical Computing in Python, *in* Stéfan van der Walt and Jarrod Millman, eds, 'Proceedings of the 9th Python in Science Conference', pp. 56 – 61.

Appendices

A Project setup

This work was done entirely using `Python` in a Jupyter notebook [10]. Several Python libraries were used:

- matplotlib [7]
- Pandas [22]
- plotly [8]
- Sci-kit learn [14]
- seaborn [20]
- Surprise [6]
- Tensorflow[1]
- UMAP [12]

B Supplementary exploratory data analysis results

B.1 Recipe ingredients

The top 10 most popular ingredients in all cuisine types are listed in the following table

Table 4: Top 10 most popular ingredients

Ingredient	Occurrence in recipes
Garlic	1622
Onion	1502
Salt	948
Olive oil	944
Chicken	938
Pepper	806
Tomato	670
Water	612
Ginger	596
Butter	545

B.2 Specific ingredients

Table 5: Specific ingredients per cuisine

Chinese	**English**	**French**	**German**
Fermented black bean	Ale	Blackberry	Bechamel
French beans	Baked beans	Brie cheese	Black forest ham
Golden syrup	Campbell	Calvados	Bratwurst
Hot bean paste	Cheshire cheese	Confit	Brick cheese
Lotus	Chestnut	Filet mignon	Chantelle mushroom
Pink bean	Dripping	Parchment paper	Corned beef
Vegetable oil	English muffin	Pastry dough	Emmenthaler cheese
Red pepper sauce	French styled green bean	Sole	Gingersnap cookie
Sea cucumber	Gelatine	Sorrel	Gingersnaps
Yellow food coloring	Gloucester	Truffle	Granulated garlic
	Gumbo		Hoagie rolls
	Horseradish		Kohlrabi
	Horseradish sauce		Oatmeal
	Lancashire cheese		Pickle
	Long grain and wld rice		Pickling spices
	Marmite		Pierogi
	Rhubarb		Rye bread
	Rutabaga		Sauerkraut
	Stilton		Thousand island dressing
	Stuffing		Vital wheat gluten
			White cheese

Greek	**Indian**	**Italian**	**Japanese**
Aleppo pepper	Amchoor powder	Alfredo sauce	Azuki bean
Celery soup	Asafetida	Bagel	Buckwheat
Greek salad dressing	Bitter melon	Emeril	Burdock root
Mizithra cheese	Black-eyed pea	Garlic butter	Dashi
Spearmint	Curry leaf	Genoa salami	Dashi stock
Wheat bread	Fenugreek	Gorgonzola	Eeel
	Ghee	Ground round	Enoki mushroom
	Ginger garlic paste	Italian bread	Hash browns
	Panir	Manicotti	Japanese breadcrumbs
	Soybean	Marinara sauce	Kamaboko

	Toor dal	Mascarpone	Nori
	Urad dal	Meatloaf mix	Rapeseed oil
	Watermelon	Muscovado sugar	Shimeji mushroom
	Yellow split pea	Pecorino cheese	Shirataki noodles
		Radiccio	Shoyu
		Sliced pepperoni	Sriracha hot sauce
		Wild rice	Sunflower seed
		Sushi rice	
		Tatsoi	
		Teriyaki marinade	
		Umeboshi	

Mexican	**Moroccan**	**Spanish**	**Thai**
Asadero	Alfalfa sprouts	Andouille sausage	Cantaloupe
Black truffle	Chicken base	Crab claw	Coconut oil
Cheddar cheese soup	Craisin	Crayfish	Edamame
Chipotle pepper	Harissa	Crusty bread	Fresh button mushroom
Coleslaw mix	Mixed spice	Fava bean	Galangal
Cocoa powder	Moroccan seasoning	Grand marnier	Kaffir lime
Corn chips	Pomegranate	Grapefruit	Ketjap manis
Cornbread	Preserved lemon	Lambs kidney	Nectarine
Epazote	Pumpkin pie spice	Pace picante sauce	Palm sugar
Fajita seasoning mix	Ras El Hanout	Pastis	Papaya
Fat free refried beans	Smen	Piquillo pepper	Peanut sauce
French dressing	Sumac	Sourdough bread	Sambal
Grating cheese	Zaatar	Spaghetti squash	Shrimp paste
Guacamole		Spanish chorizo	Straw mushroom
Havarti		Strawberry	Thai curry paste
Hominy		Truffle oil	Yellow curry paste
Miracle whip dressing		Vegetable juice	
Picante sauce			
Pinto bean			
Poblano pepper			
Polenta			
Queso			
Refried beans			
Sazon			
Taco seasoning			
Tomatillos			
Unsweetened chocolate			
Velveeta			

C Dimensionality reduction techniques

In this appendix, we provide the results obtained from all the dimensionality reduction techniques explored. While not all the techniques provided insightful results, they are detailed below for the readers interest.

C.1 PCA

In order to run PCA, the number of components was chosen to be 3. The explained variation per principal component is displayed in the below table:

Table 6: Fraction of variance explained per principal component

Direction	1	2	3
Fraction	0.054	0.043	0.037

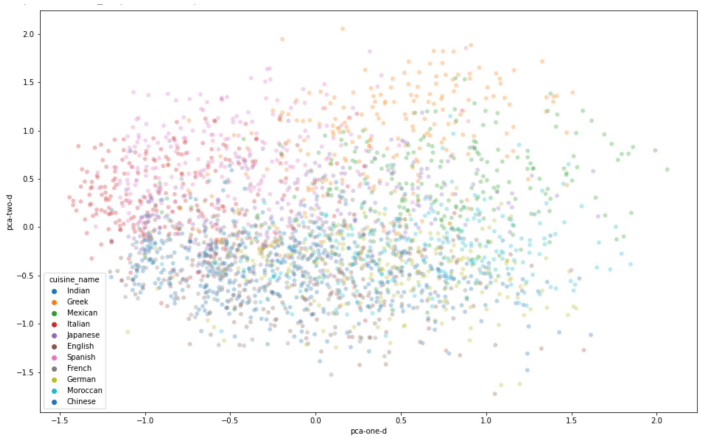


Figure 5: PCA scores visualization

The first 3 principal components explain only the 13.55% variance of the data. Figure 5 shows that PCA does not seem to unveil anything useful for understanding the data. This could be due to the small variance explained with these components.

C.2 Kernel PCA

Kernel PCA was also surveyed and several kernels were attempted; however, this technique did not help to unveil more information about this dataset.

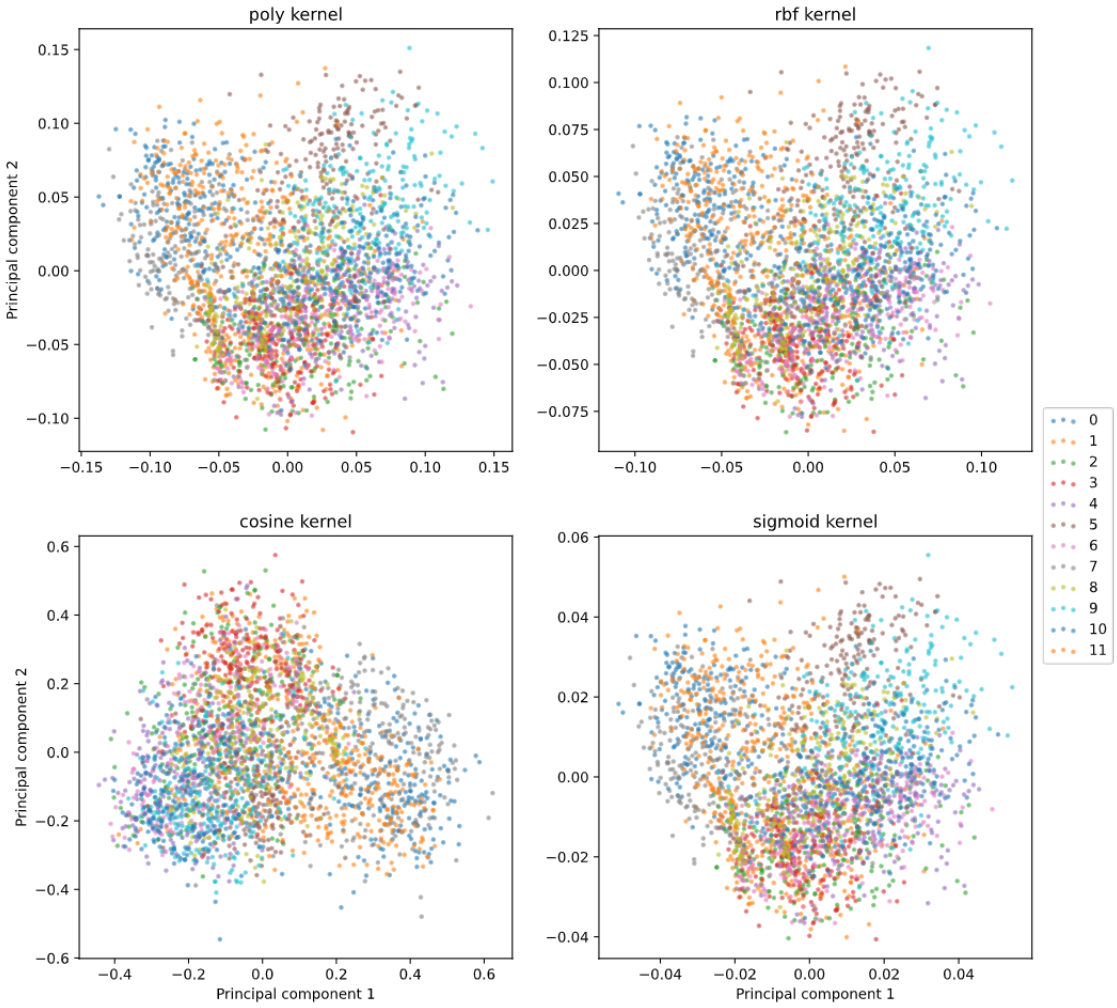


Figure 6: Kernel PCA reduction using different kernels

C.3 Metric MDS

As in the previous techniques surveyed, metric MDS failed to unveil insightful clusters.

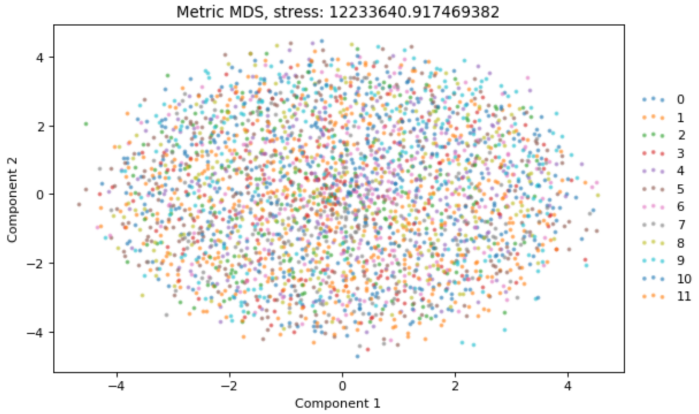


Figure 7: Metric MDS representation

C.4 Isomap

Isomap was also surveyed by experimenting with the number of neighbors. The below figure shows the results.

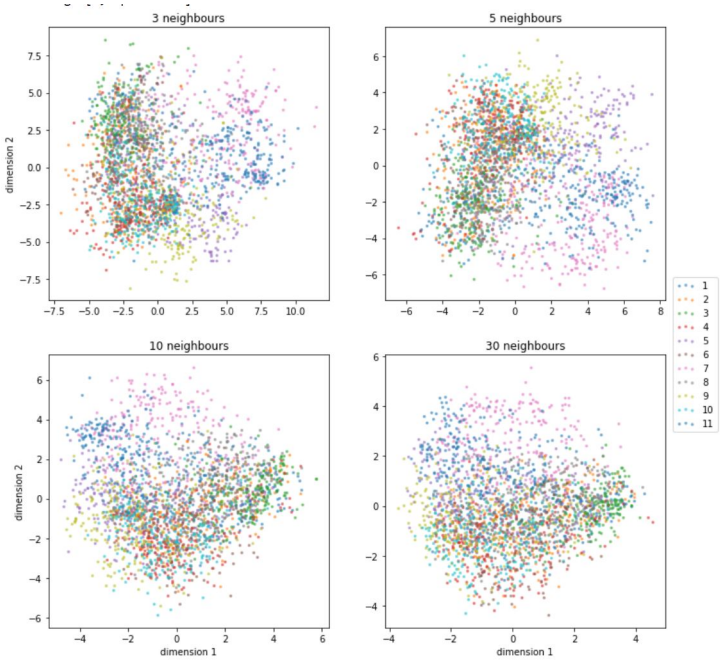


Figure 8: Isomap results

Overall, all techniques utilized are perhaps not as insightful as expected, due to the high-dimension of the dataset. Despite, 171 ingredients were not considered for dimensionality reduction, there are other approaches to be explored, such as removing near-zero variance features. This will be explored in the coming weeks.

C.5 t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding) is a cutting edge technique that is explored by experimenting with the perplexity hyperparameter, as per Wattenberg et al. [21]. While this

technique threw different results than the previous techniques, the results are not insightful enough for understanding clusters nor for prediction purposes.

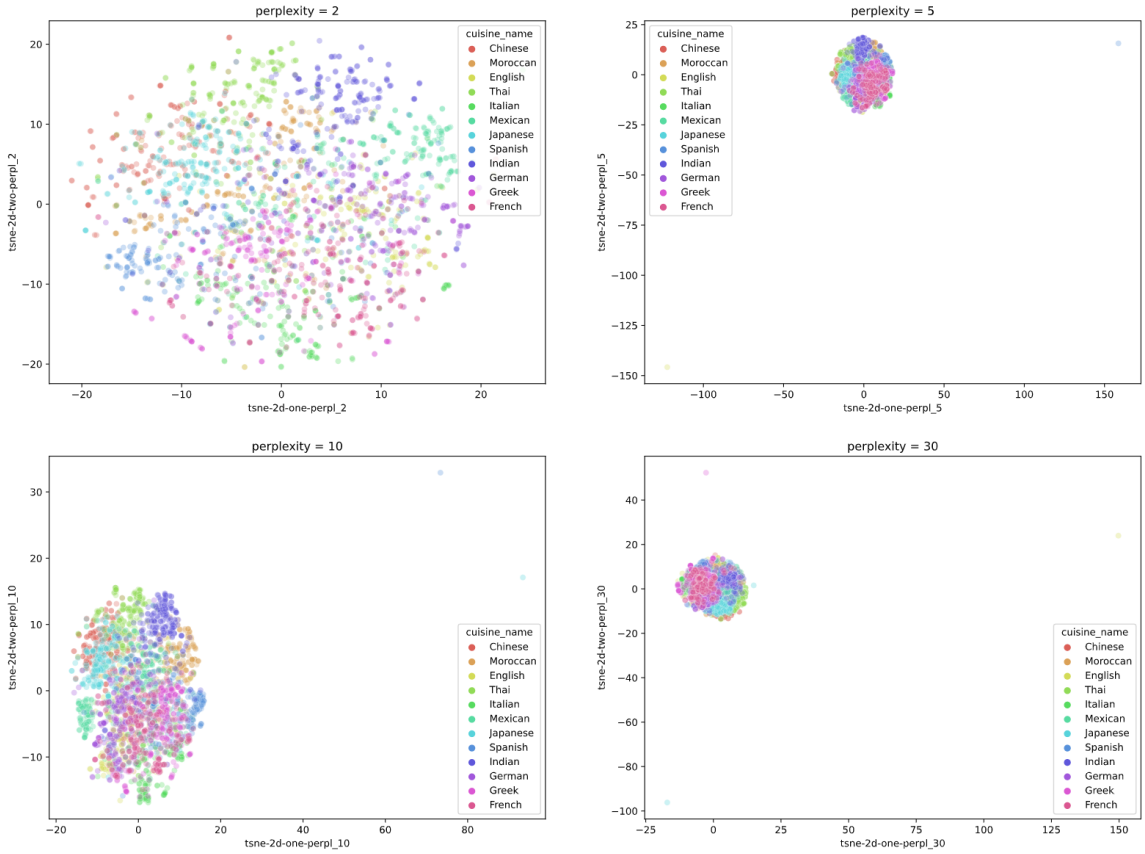


Figure 9: t-SNE representations with different perplexity values

C.6 UMAP

As per the discussion in section 4.1, UMAP was the technique that offered an insightful representation of the data, despite using only 2 components. The metrics and the n-neighbors (nearest neigbors) hyperparameters were explored for assessing the quality of the dimensionality reduction. In the below subsections the results are shown for each hyperparameter survey.

C.6.1 Metrics

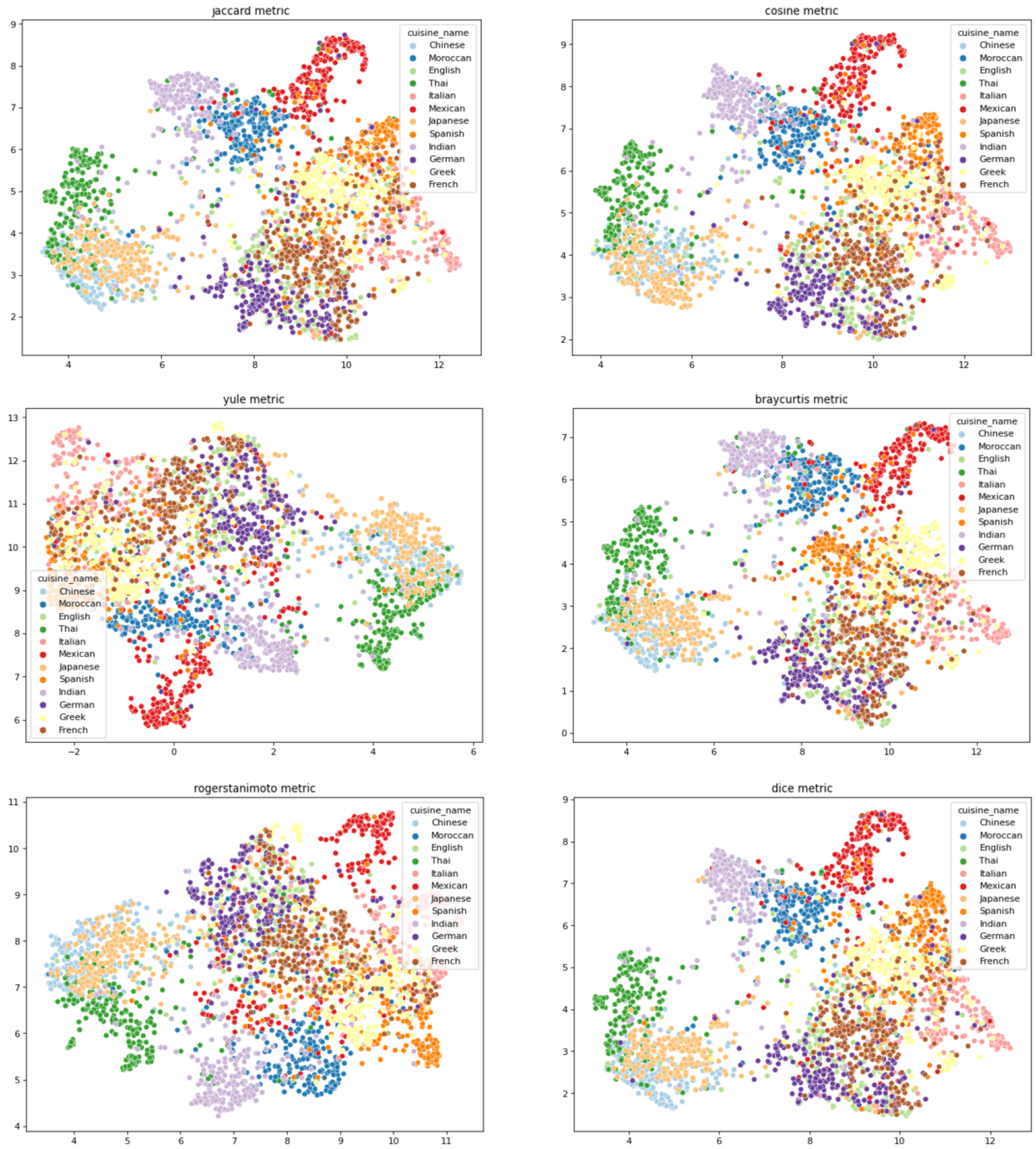


Figure 10: UMAP reduction using different metrics

C.6.2 Nearest neighbors

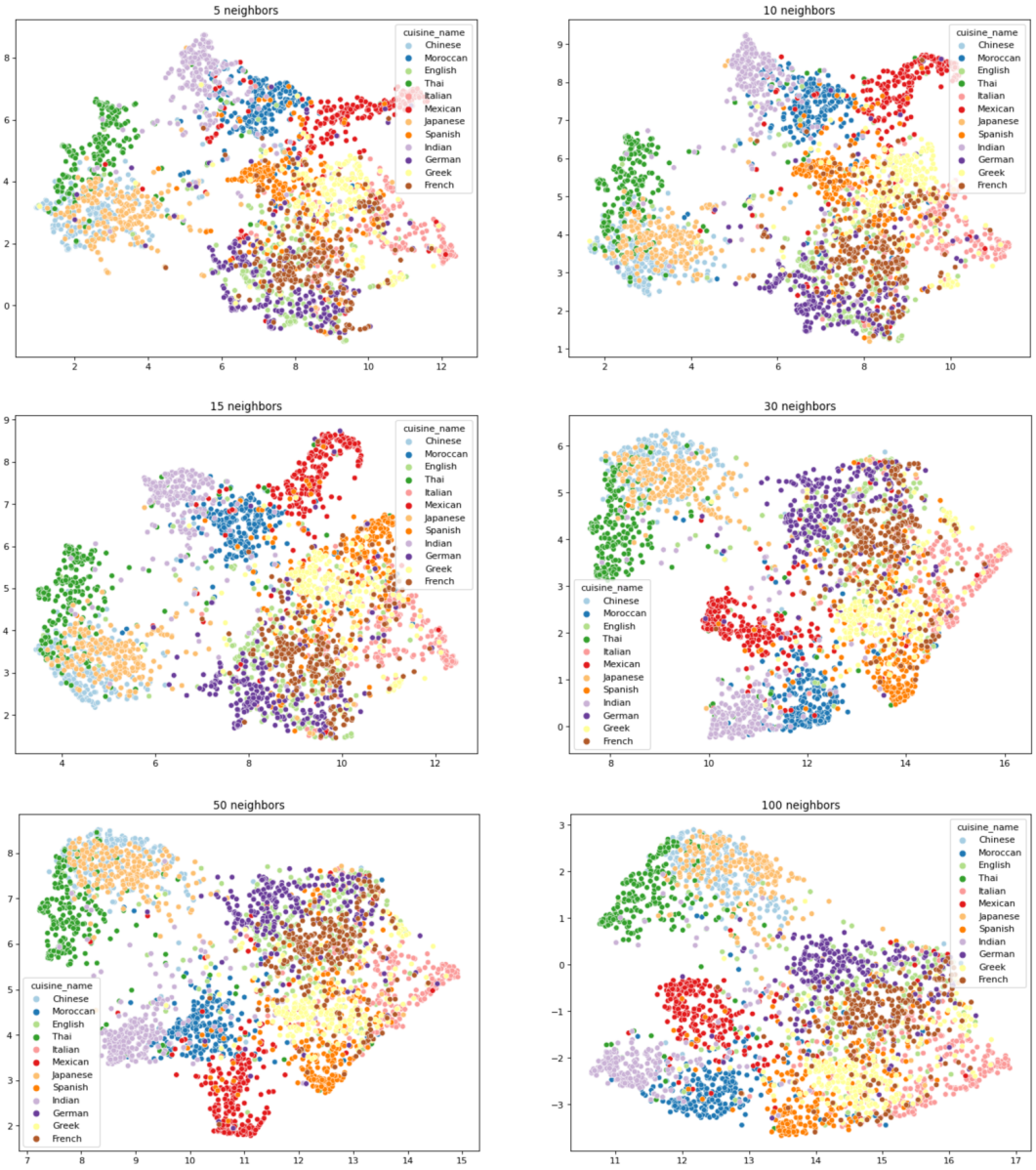


Figure 11: UMAP reduction using different nearest neighbors