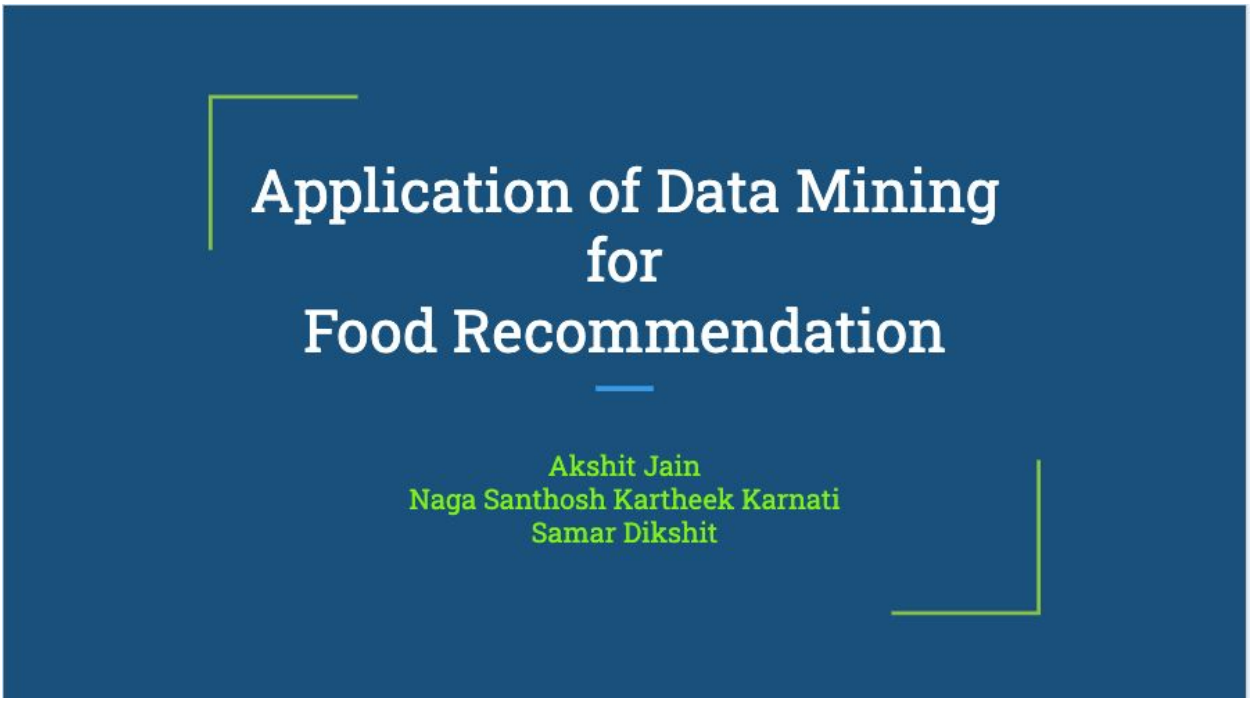
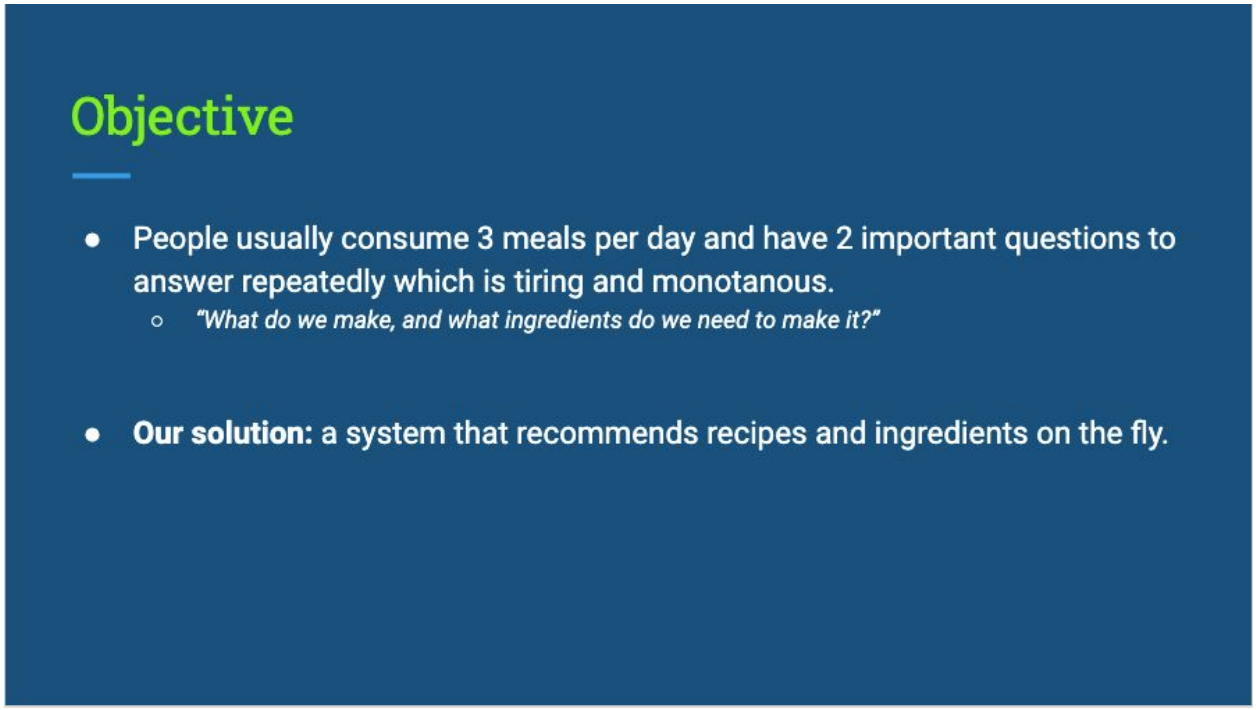
DS5230 - Project Report
08.12.2020



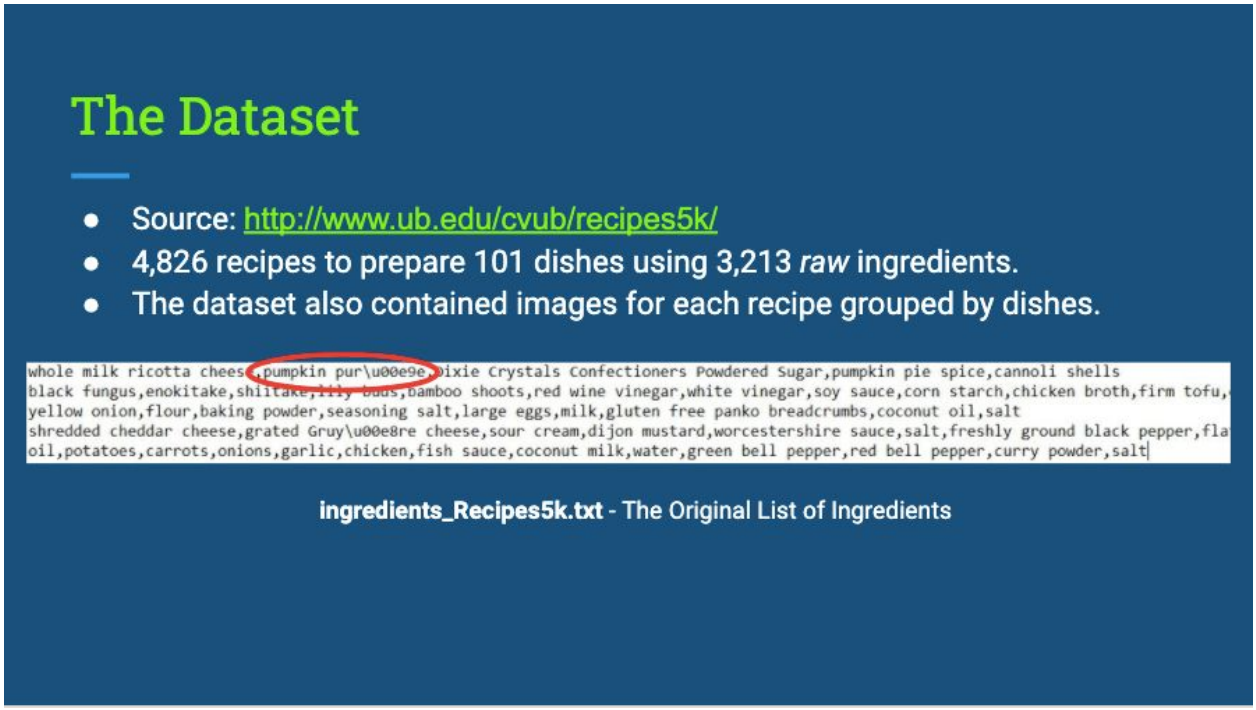


Normally, people have three meals a day. For each meal, answering the questions "*what do we make?*", and "*what ingredients do we need?"* can be repetitive and tiring.

Our solution is to build a recommendation system that can help with suggesting recipes and ingredients when required.

The procedure we followed for this was as follows:

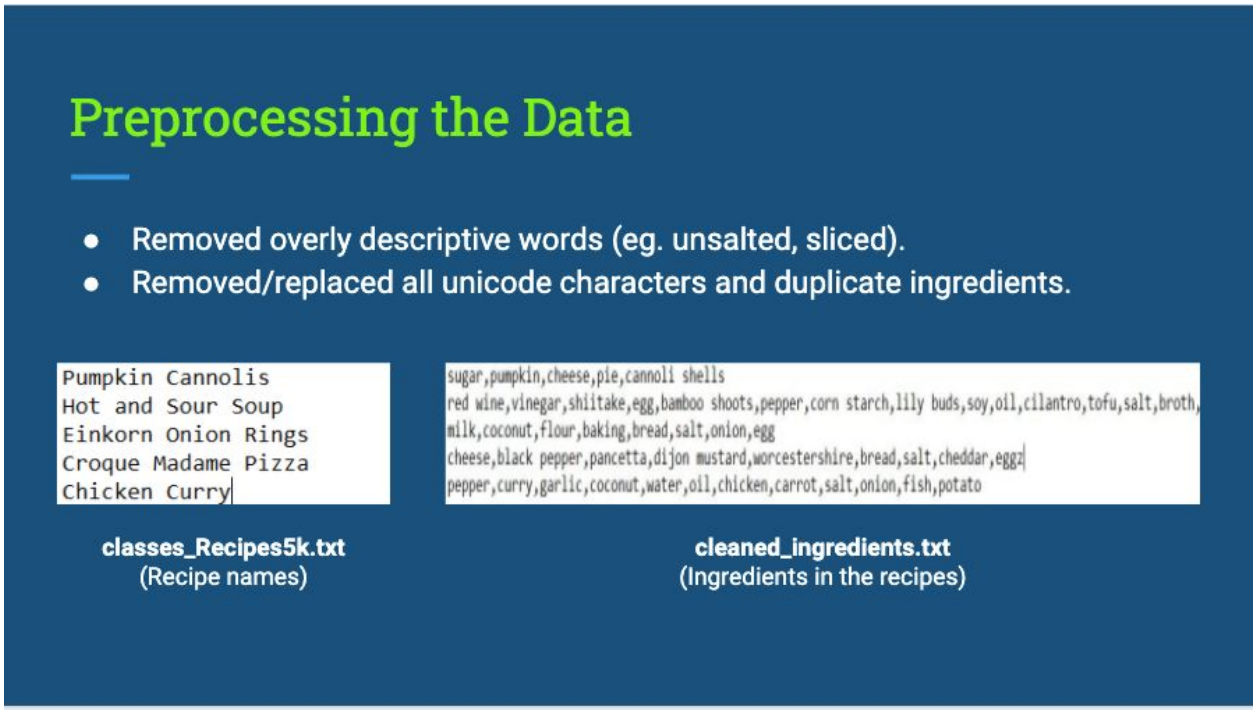
1. Obtain the dataset, clean the data, and prepare it for exploratory data analysis.
2. Perform EDA and generate plots to help summarise the data.
3. Represent all recipes as a network of ingredients.
4. Mine association rules to provide a probabilistic perspective on what ingredients tend to occur together across recipes.
5. Cluster recipes based on their constituent ingredients.
6. Build an ingredient-based recommendation system using two different vectorization techniques: Doc2Vec, and one-hot encoding.



In order to perform our analysis and build a recommendation system, we obtained food data from the web platform Yummly.

The dataset contained over **4,800** unique recipes to prepare **101** different dishes. On average, each dish had **50** alternative recipes.

The unique recipes contained more than **3,200** ingredients (10 per recipe on average), providing multiple ways of preparing a single dish. Hence, it captures at the same time the intra and inter-class variability of cooking recipes.



Each line of the dataset represents a different recipe, with comma-separated raw ingredients. However, due to the presence of unicode characters and overly descriptive words, along with duplicate ingredients, the dataset required preprocessing.

During preprocessing, these descriptors, such as *yellow* in *yellow onion* and *sliced* in *sliced tomato*, were removed to create a more generalised set of ingredients.

Unicode characters were either removed or replaced, depending on the case. For example, the *è* character in *Gruyère* was replaced with *e*, however non-Latin characters and symbols were removed all together. Duplicate ingredients were also removed from each recipe.

This brought down the number of ingredients from over **3,200** to **1,010**.

Preprocessed Dataset for Food Analysis

Removed recipes which don't map to any specific dish, resulting in **4,330 unique recipes, 668 unique ingredients and 101 unique dishes.**

	recipe_name	ingredients	dish
0	Freezer Breakfast Burritos_3	pepper,salsa,garlic,black	breakfast_burrito
1	Mobile-Style Oysters	pepper,parsley,garlic,cheese,black	oysters
2	Cheesecake	sugar,egg,cheese,corn	cheesecake
3	Fried Ravioli_6	ravioli,water,oil,bread,marinara,egg	ravioli
4	Poutine_10	gravy,pepper,cheese,butter,oil,flour,beef,salt...	poutine
...	...	...	...
4325	Death By Oreo Cupcakes	sugar,cheese,oreo	cupcakes
4326	Thai Green Chicken Curry	basil,coconut,gin,chicken,zucchini,mushroom,sp...	chicken_curry
4327	Mussels in White Wine	pepper,parsley,garlic,oil,white	mussels
4328	Low Carb Crab Cakes	pepper,parsley,mayonnaise,crab,oil,lemon,dijon	crab_cakes
4329	Tart Vanilla Frozen Yogurt	plain	frozen_yogurt

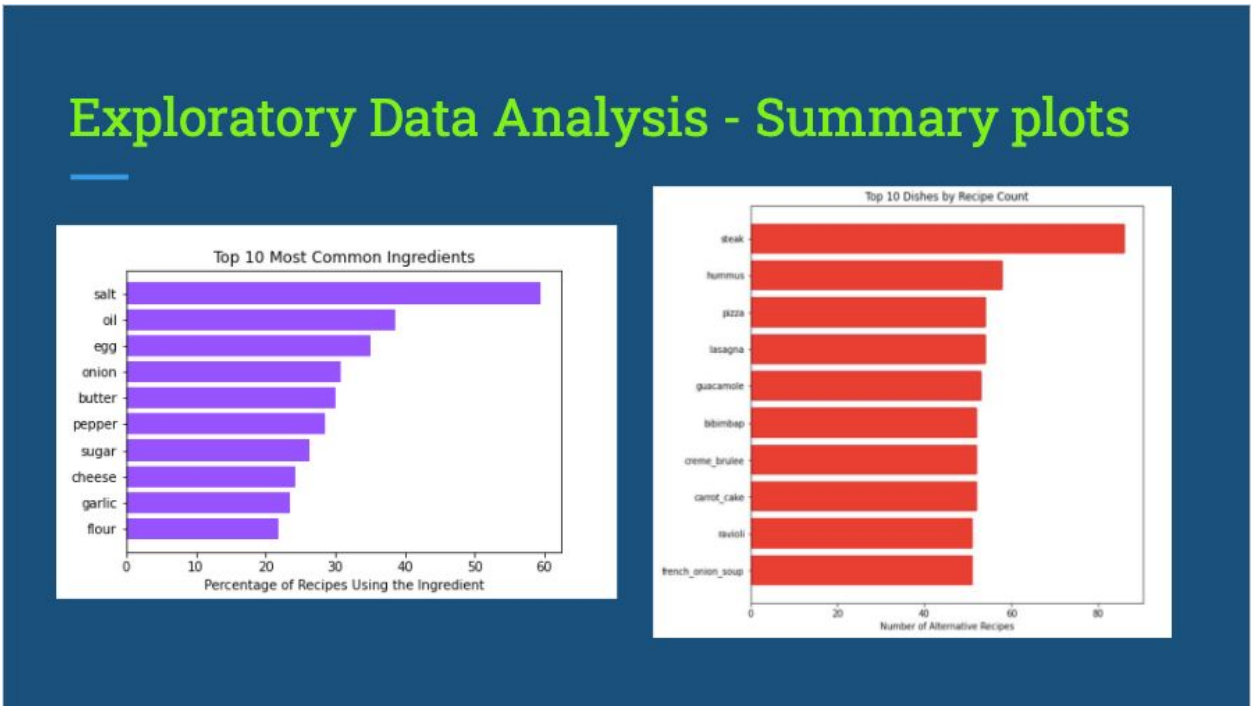
4330 rows × 4 columns

Final dataset

The final step of the preprocessing was to map recipes to the names of the dish.

Recipes without any mapping to a dish were removed. Doing so gave us the final dataset, which contained **4,330** recipes for **101** dishes using **668** different ingredients.

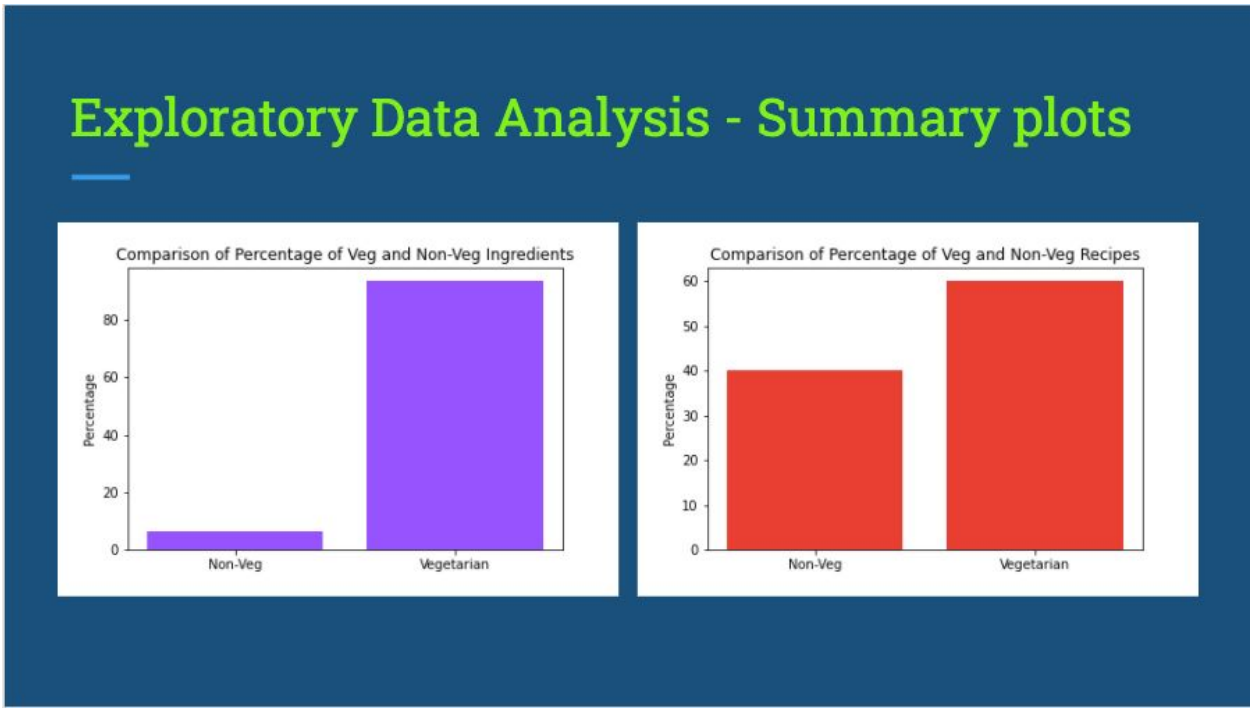
This dataset contains three columns. The first, *recipe_name*, is the name of the recipe. The *ingredients* column contains the list of ingredients required for that recipe, and the *dish* column contains the name of the dish.



In order to understand the data better, we found the most common ingredients used in dishes, and dishes with the most recipe count.

Salt, oil, egg, onion, butter, pepper, sugar, cheese, garlic and flour are the 10 most common ingredients. **Salt** is the only ingredient that is used in over **50%** of recipes, while **flour** is used in just over **20%** of all recipes. Looking at the top 10, none of the ingredients present are surprising as they are also ubiquitous in grocery stores and pantries.

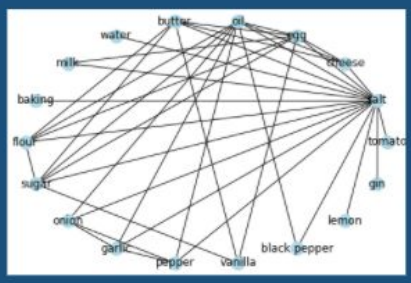
Steak, hummus, pizza, lasagna, guacamole, bibimbap, creme brulee, carrot cake, ravioli and French onion soup have the most number of alternative recipes. Hence, these 10 dishes have the most **intra-class variance**. These dishes belong to different cuisines and have varying preparation times. This indicates that there is no single cuisine that is preferred over the other and that there is no preconceived notion that time taking dishes aren't popular enough.



We categorized the numerous recipes for dishes into vegetarian and non-vegetarian recipes. The ingredients were also categorized into vegetarian and non-vegetarian ingredients.

We noticed that there are more vegetarian recipes and vegetarian ingredients when compared to non-vegetarian recipes and non-vegetarian ingredients. There is a greater imbalance in the ingredients **(7% non-veg vs 93% veg)** when compared to the imbalance in recipes **(40% non-veg vs 60% veg)**. This can be explained by the fact that any recipe with even a single non-vegetarian ingredient is categorized as a non-vegetarian recipe.

Graphs and Association Analysis



Pairs that co-occur the most:

1. Salt and oil
2. Salt and pepper
3. Salt and egg
4. Salt and butter
5. Salt and onion

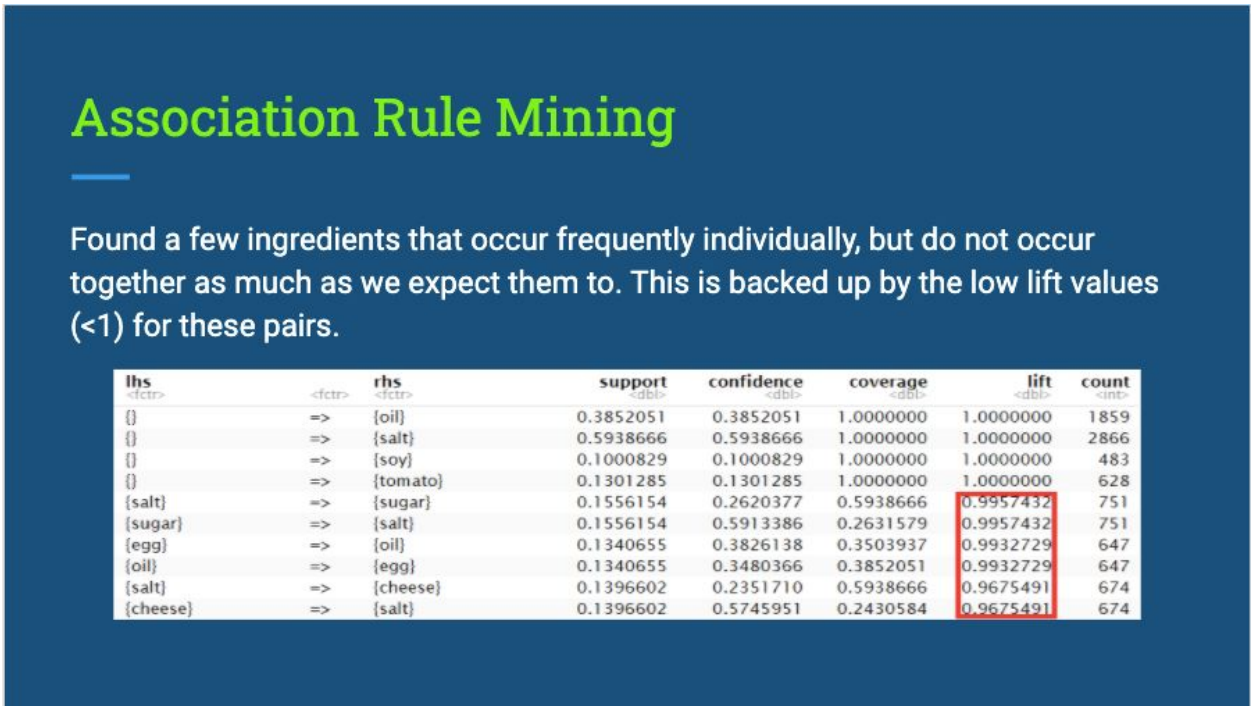
Market Basket Analysis backed up our network analysis

	items	support	transIdenticalToItemsets	count
[1]	{oil,salt}	0.2679237	0.0026937422	1293
[2]	{pepper,salt}	0.2206797	0.0060091173	1065
[3]	{egg,salt}	0.2186075	0.0024865313	1055
[4]	{butter,salt}	0.2028595	0.0012432656	979
[5]	{onion,salt}	0.2009946	0.0035225860	970
[6]	{flour,salt}	0.1732283	0.0002072109	836
[7]	{garlic,salt}	0.1647327	0.0014504766	795
[8]	{egg,sugar}	0.1603813	0.0103605470	774
[9]	{salt,sugar}	0.1556154	0.0010360547	751
[10]	{black pepper,salt}	0.1508496	0.0060091173	728

To represent recipes as a network of ingredients, we counted each pair of ingredients that occurred together and created a weighted undirected graph. The weight of each edge was set to the count of co-occurrences. Without thresholding, this network graph initially contained **668 nodes (one per ingredient)**.

To allow us to concentrate on the pairs that occur together more often, we used the edge weight as a threshold. Following a **bottom-up approach**, we slowly increased this threshold value, with the **final value set to 400**. Each pair of connected nodes in the final network occurs together at least 400 times, with **salt and oil** being the most frequently occurring pair **(1,293 times)**, followed by **salt and pepper (1,065 times)** and **salt and egg (1,055 times)**. In fact, **salt appears in 8 of the top 10** pairs. Since we had prior knowledge from our EDA that salt is the most common ingredient, it makes sense to see it co-occur often too.

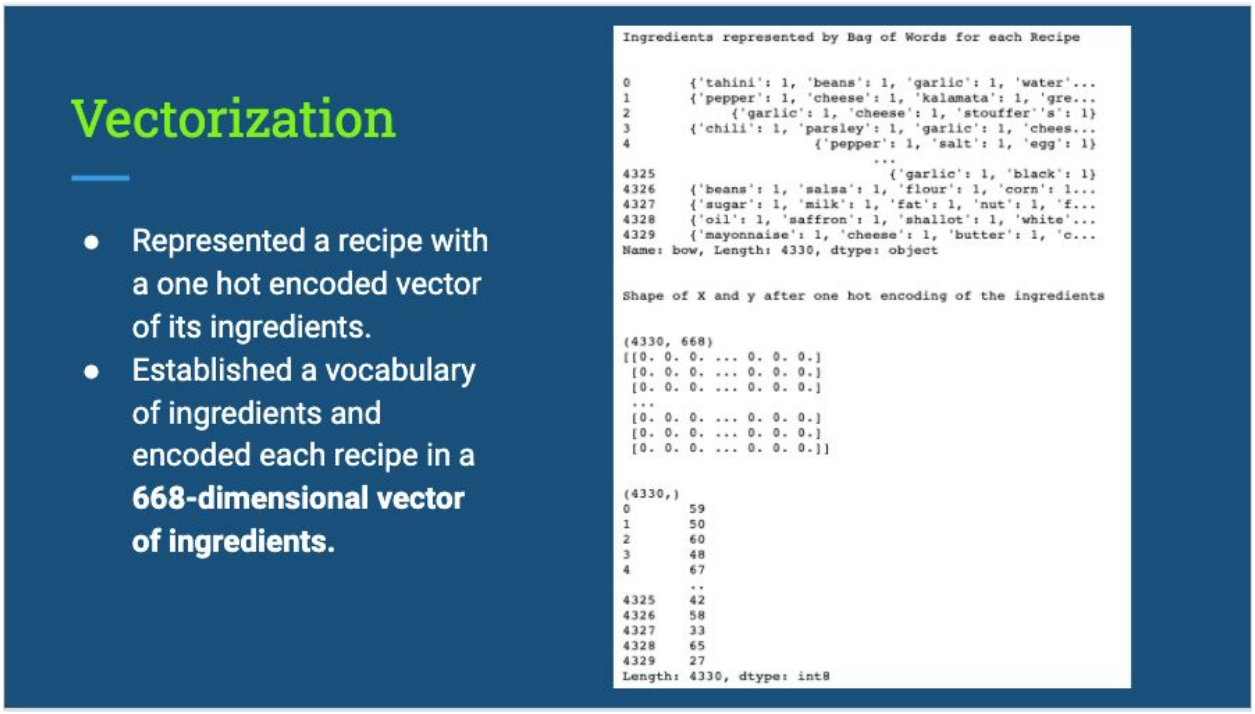
Market Basket Analysis backed up our network analysis results. We used the **Apriori** algorithm to find frequent itemsets (here, each item is an ingredient) of size 2 in order to find the most common co-occurring pairs of ingredients used in the preparation of dishes. The **minimum support is set at 0.1** i.e. any itemsets which occur in at least 10% of the dishes are frequent enough.



Association rules were mined from the data using the **Apriori** algorithm to find a few ingredients that occur frequently individually but do not occur together as much as we expect them to. This is backed up by the **low lift values (<1)** for these pairs.

A few such pairs are sugar and salt, egg and oil and salt and cheese. The rule, {salt} => {sugar} has a lift of 0.99 which tells us that having salt in our recipe does not increase the chance of occurrence of sugar in spite of the rule showing a high confidence value.

Minimum support of 0.1 and **minimum confidence of 0.1** were chosen since we didn't want to mine rules that aren't frequent enough.



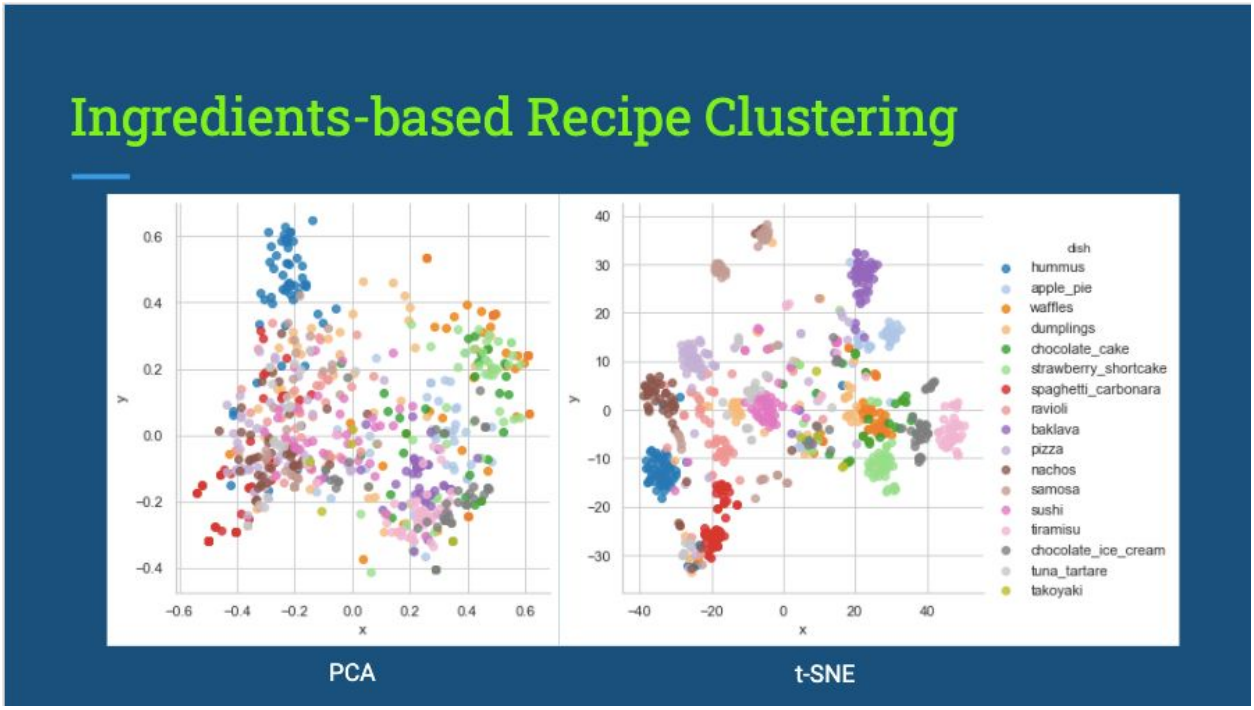
As we were interested in finding similarities between dishes, performing network analysis and clustering recipes using their ingredients would help visualize those similarities and differences.

We represented a recipe by a one hot encoded vector of its ingredients. To do so, we used DictVectorizer provided by scikit learn which established a vocabulary of ingredients and encoded each recipe in a 668-dimensional vector of ingredients.

Once we had the one hot encoding of all recipes, we needed a way to project the high dimensional one hot encoding vector to a 2 dimensional space. We achieved this using the following methods:

- **PCA:** finds linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. (In our case we used it with the cosine kernel as distance metric)

- **t-SNE:** is based on probability distributions with random walk on neighborhood graphs to find the structure within the data.



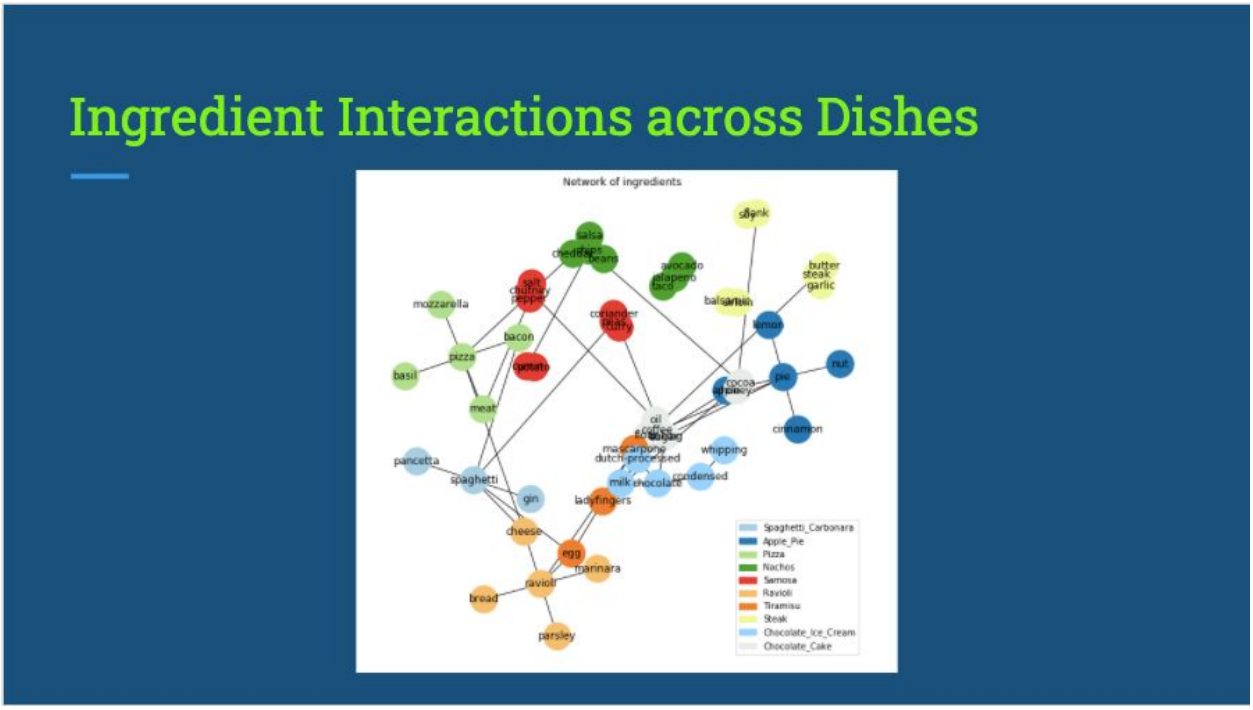
PCA's limitation is that it can not find non-linear relationships between data points. That's one reason why we chose t-SNE as an alternative technique to reduce dimensionsionality of our data.

Besides, we used cosine-similarity as a metric, since for our use-case, we wanted to abstract out the magnitude of every vector and look at the “directional similarity” between two vectors.

Following which, we filtered the dataset for our convenience, and plotted the clusters for a **subset of dishes** only. The latter were randomly chosen within 101 dishes. Finally, we computed the square form of cosine similarity of the one hot encoded vectors in our dataset.

From the PCA plot we can observe the presence of some clusters as tiramisu and chocolate ice-cream clusters overlap (desserts), and spaghetti and pizza clusters overlap (Italian). PCA with cosine kernel does fairly well for the subset of dishes but as the number of dishes increase, it will be impossible to distinguish the clusters.

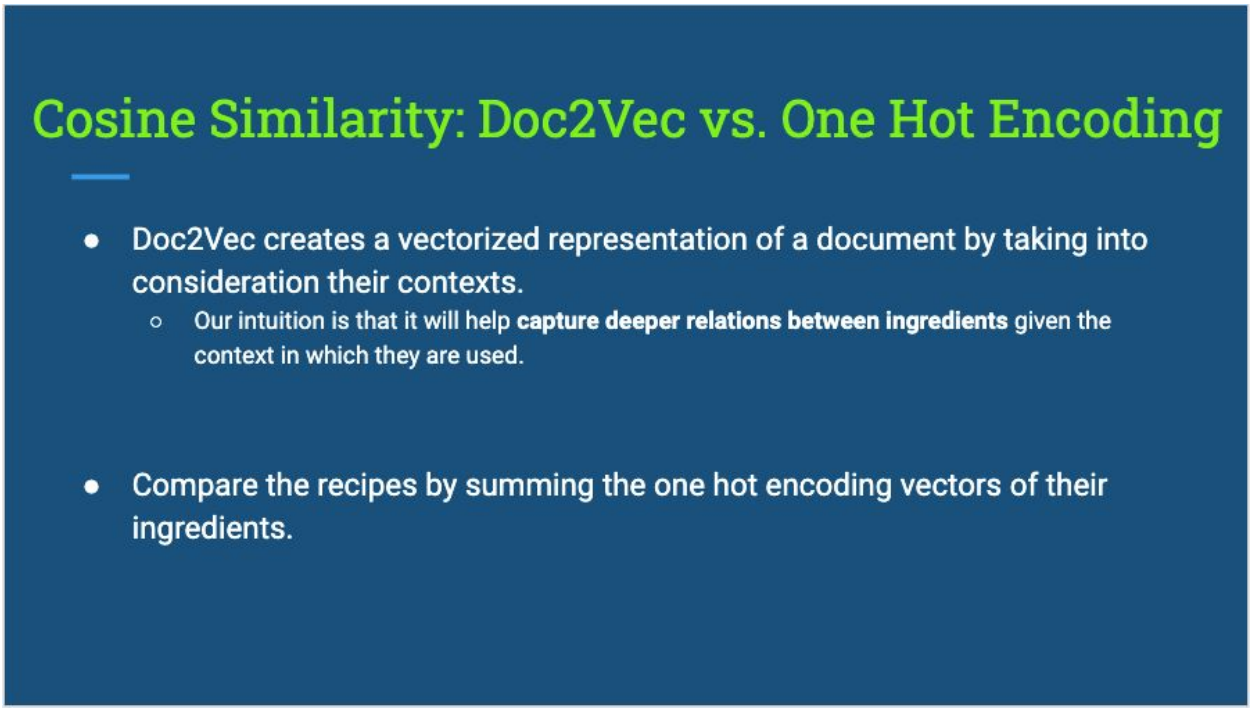
At first observation, t-SNE projection is much clearer than PCA: clusters are more visible and we have far better insights in comparison to KPCA (e.g. Baklava, Tiramisu, Chocolate Cake and other desserts (dishes) are relatively close in both graphs, however more obvious with t-SNE).



In this section, we analyzed combinations of ingredients in dish recipes for a better description of a dish. Our approach was to extract the frequently used combination of ingredients that are typical to each dish.

First, we compute the pairwise appearance frequency between all ingredients in our dataset which will generate a N by N matrix of frequencies (N is the number of ingredients). We then repeated the same operation for every dish present in the filtered subset.

From the network of ingredients, we can observe how several dishes share ingredients. The dishes which can be categorized as desserts (i.e. apple pie, tiramisu, chocolate cake and chocolate ice-cream) share ingredients like ‘milk’, ‘cocoa’, ‘oil’ etc. Also, Italian dishes like pizza and spaghetti have several ingredients in common too.



After the previous analysis, we saw how dishes were similar to each other in different aspects. Therefore, we decided to build a recommendation system for recipes.

For a comparison purpose, we approached this task using two different vectorization techniques to estimate the performance of the recommendation model since there was no clear metric associated with the task. The following two techniques were used:

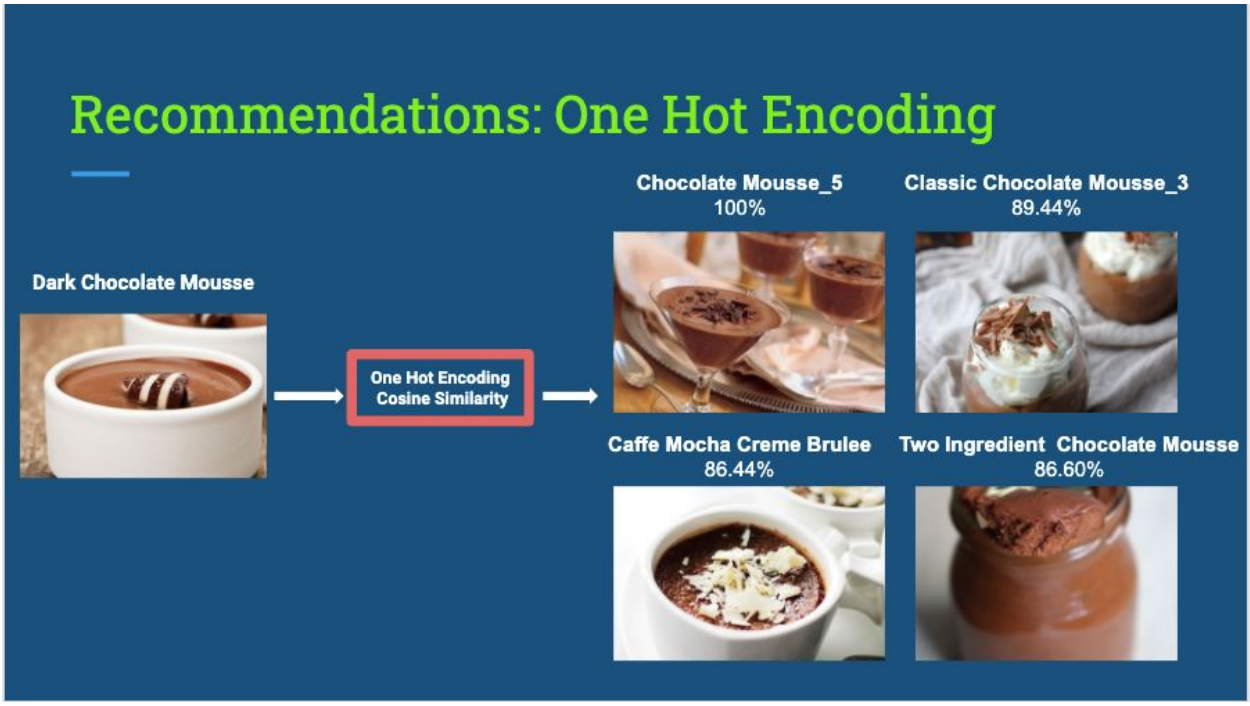
- **Doc2Vec:** creates a vectorized representation of a document by taking into consideration their contexts. Our intuition is that it will help capture deeper relations between ingredients given the context in which they are used.

- **One-hot encoding:** compare the recipes by summing the one-hot encoding vectors of their ingredients and computing the cosine similarity between them. Hence, with this method, the problem of answering the question, “How two recipes are close to each other?” reduces to evaluating the cosine similarity of their vector representation. A value close to 1 indicate a high similarity between the two recipes, whereas a value close to 0 indicate no similarity between the two recipes.

The Doc2Vec algorithm builds a neural network to predict a word based on its context. The size of the context is defined beforehand and is called the window size. For example, if we choose a window size of 2, every word's context will be defined based on the 2 words that neighbor it.

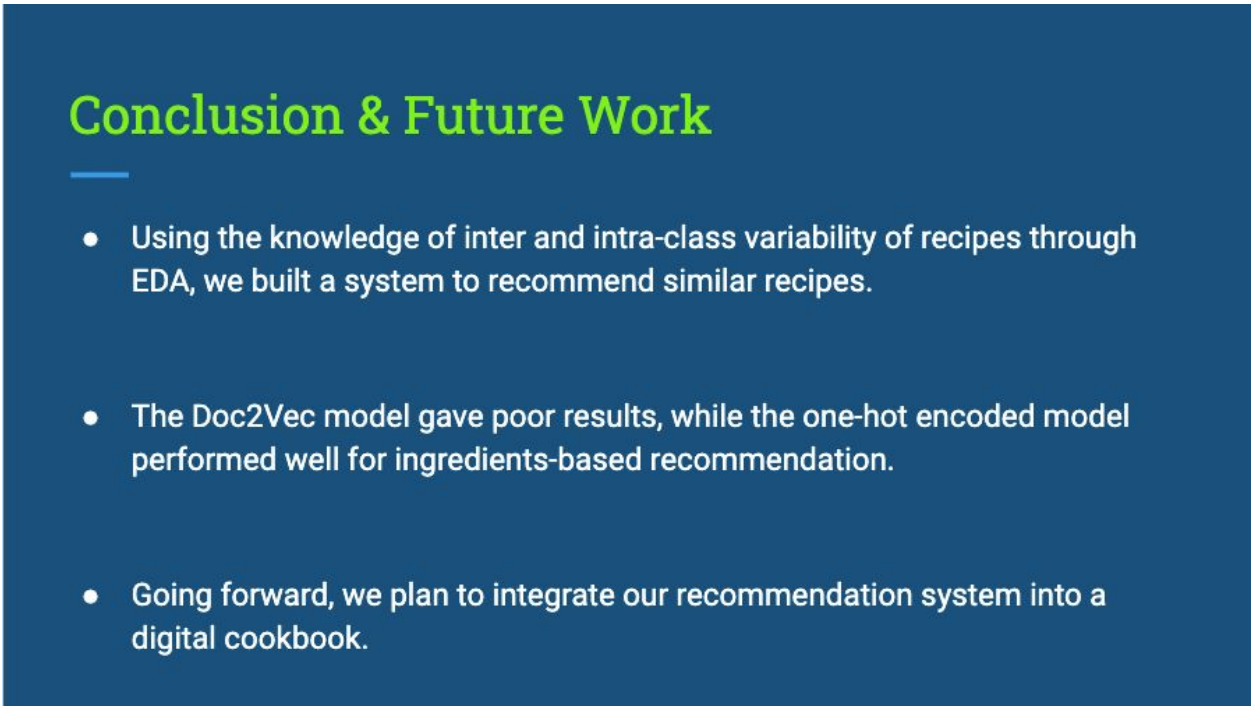
As we can see from the recommendations provided for Dark Chocolate Mousse using the Doc2Vec algorithm from Gensim, the approach does not perform well. The reason behind it is that Doc2Vec uses the misleading representation of recipe as a document where the order of the words (in this case, the ingredient names) matter which is not the case. Thus, if the size of the aforementioned window is not big enough, Doc2Vec would be limited in inferring the context of ingredients which may lead to poor results.

Even though the confidence for all the recommendations is 100%, not all the recommended recipes are quite similar to the input recipe solely based on the recipe, they might however share common ingredients.



From the recommendations obtained from cosine similarity with one-hot encoded vectors, we can state that this technique would be more suited for the task of ingredients-based recipe recommendation, since it computes the dot product between ingredients which preserves the relationship between them.

For instance, if we are looking for similar recipes for Dark Chocolate Mousse with ‘sugar’, ‘chocolate’ and ‘eggs’, the one-hot encoding approach will associate higher weights to recipes having all these ingredients, ignoring the context in which the ingredients actually occur, which is the case with the Doc2Vec approach.



In this project, we answered various questions by analysing our dataset. We established similarities between dishes, their mutual influences and finally characterized each with a set of distinctive components. To do so, we needed to carefully explore and clean the data during which we faced challenges such as correcting the ingredients names and dealing with duplicate values. Besides, we explored various techniques, some of which revealed good results and led to meaningful insights.

Going forward, we want to collect more data i.e. include more dishes and recipes in our dataset. We will group the dishes into appetizers, main course and desserts to get better recommendations. Furthermore, we plan to integrate our recommendation system into a digital cookbook.