



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

*Distributed
Computing*



Smart Recipes

Bachelor Thesis

Alessio Bähler

`abaehler@student.ethz.ch`

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:

Laura Peer

Prof. Dr. Roger Wattenhofer

December 22, 2015

Abstract

With the increasing popularity of recipe sharing websites over the last years the Internet has become an accessible source for large quantities of recipe information. A major problem that we encounter, is that the recipe format varies greatly among the different sources. We therefore collect a large amount of recipes from different websites and we parse the content in order to obtain a uniform structure. Afterwards, we implement a classification schema to determine the difficulty level of a recipe based on the collected data. For this purpose we consider ingredients, timings and instructions and we use *Gaussian Naive Bayes* and *Decision Tree* classification techniques to assess the quality of our methods. Furthermore, we consider the space determined by the most frequent ingredients found among a subset of recipes to discover similarities and differences between recipes belonging to different world cuisines and meal categories, for example main dishes, desserts or appetizers.

Contents

Abstract	i
1 Introduction	1
1.1 Motivation	1
1.2 Related Work	2
1.3 Overview	2
2 Background	3
2.1 Classification	3
2.1.1 Gaussian Naive Bayes	3
2.1.2 Classification Trees	4
2.2 K-Means Clustering	5
3 Design	7
3.1 Data Acquisition	7
3.1.1 Collecting Recipes	7
3.1.2 Structuring Collected Data	8
3.2 Recipe Difficulty Classification	10
3.2.1 Developing an Objective Method	10
3.2.2 Assessing the validity of our methods	11
3.3 Discovering Recipe Similarities	11
3.3.1 Recipe Similarity using K-Means Clustering	11
3.3.2 Recipe Similarity using Classification Trees	12
4 Implementation	13
5 Results	14
5.1 Difficulty Classification Results	14
5.2 Category Similarities	16

CONTENTS	iii
5.2.1 K-Means Clustering	16
5.2.2 Tree Classification	21
5.3 Cuisine similarities	26
5.3.1 K-Means Clustering	26
5.3.2 Tree Classification	30
6 Outlook and Summary	34
Bibliography	35
7 Appendix Chapter	37

Introduction

1.1 Motivation

Since ancient times, recording and sharing cooking recipes has been a traditional human activity. The recipe description was usually printed on a sheet of paper. The increasing ubiquity and popularity of the Internet opened the doors for large-scale collaborative sharing of recipes like it has never been possible before. Currently, there are millions of recipes from all over the world, published for free by people of different cultures and with varying cooking skills that are accessible to anyone. This trend is advantageous since it enables the exchange of recipes between mutually remote geographical locations and cultures, but it also has a major drawback: it is difficult to find specific recipes between millions of others. We are therefore interested in finding ways to improve recipe search by different parameters like ingredients, cuisine, category, nutritional values, cost and so on. But we could do a lot more. We could find ways to combine ingredients in order to create new recipes, replace ingredients with similar ones or learn more about the eating habits of the people and provide useful hints on how to improve the food quality. Since nowadays it is even possible to buy food on the Internet we could build a service that for example generates recipes for the user and it allows him to buy all the necessary ingredients with just one click. Afterwards, the products could be shipped directly to the user's home.

The first step we have to take is obviously getting the data. However there is neither a common structure nor a common language among the different recipe repositories, therefore the collected data needs to be processed and structured uniformly before any analysis can be done. In this work we make the first steps in collecting a large quantity of recipes from different recipe sharing websites and save each one in a uniform structure. Additionally, we try to develop a method to determine the difficulty level of a recipe by looking only at the ingredients that are needed, the timings and the instructions. To assess the accuracy we use some simple classification algorithms. We also use K-Means clustering to find relations between recipes belonging to different regional cuisines or meal categories in the space defined by the most frequent ingredients.

1.2 Related Work

The features offered by recipe sharing websites vary from a simple list of recipes to advanced search options and sometimes even include shopping list management. The website *allrecipes.com* [1] for example, groups recipes into cuisine and category, for example main dish and appetizer. Additionally, it enables the search of recipe names and ingredients, offers ingredient checklists and even allows the user to buy some products online. In contrast, *epicurious.com* [2] doesn't offer the last two options, but it has extended search capabilities, which allow the user to filter the results by meal, diet, ingredients, cuisine, dish type and preparation method.

Recipe retrieval and recommendation has also been a common research topic. Past work includes considering overlapping ingredients to find similar recipes based on cooking related web navigation history [3] or on users' past recipe ratings [4]. Teng et al. [5] developed two different ingredient networks that express relations between ingredients occurring frequently together as well as substitution candidates to predict user ratings of recipes. Another research branch focussed on discovering relations between ingredients and cuisines using generative probabilistic models [6] and classification techniques [7].

1.3 Overview

In Chapter 2 we introduce the different machine learning algorithms used throughout this work. In Chapter 3 we show the collection and structuring process (Section 3.1), how we developed our difficulty classification model (Section 3.2) and finally how the ingredient space clustering works (Section 3.3). Chapter 4 contains a brief overview on the tools we used to implement the scripts and the databases, while the analysis' results are explained in Chapter 5. Chapter 6 serves as summary of our work and proposes some possible extensions.

Background

This chapter is intended to give a quick introduction into the different machine learning algorithms used during this work. We firstly explain *Gaussian Naive Bayes* and *Decision Trees* classification approaches and secondly the *K-Means* clustering approach.

2.1 Classification

Machine learning algorithms are divided into different categories, one of which is supervised learning. In supervised learning, the task is to infer a function from a given "training" labelled dataset. The dataset contains examples in form of tuples, where the first element represents the input and the second represents the desired output, which are fed to the algorithm. The algorithm uses this data to infer a function that can afterwards be used to predict the output of values not included in the training dataset, i.e. whose output is not known. Classification is the subset of supervised learning techniques that considers distinct categories as target output.

2.1.1 Gaussian Naive Bayes

Naive Bayes classification refers to a set of classification algorithms based on the application of Bayes' theorem with the "naive" assumption that all variables are pairwise independent. Using different distributions it's then possible to obtain different algorithms.

Given a category variable y and a number of features x_1, \dots, x_n , Bayes' theorem states that the conditional probability of y being a certain value Y , knowing the value of all x_1, \dots, x_n , can be decomposed as

$$P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)},$$

With the naive assumption that all features x_1, \dots, x_n are pairwise independent we can repeatedly apply the independence rule $P(x_i, x_j|y) = P(x_i|y) * P(x_j|y)$

on $P(x_1, \dots, x_n)$ and obtain

$$P(y|x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i|y)}{P(x_1, \dots, x_n)}.$$

Since $P(x_1, \dots, x_n)$ doesn't depend on y , but only on the input values x_1, \dots, x_n , to find the class that fits with the highest probability to the input data, we can use

$$y = \arg \max_y P(y) \prod_{i=1}^n P(x_i|y)$$

as classification rule.

In Gaussian Naive Bayes the probability distribution function P is

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma_y^2}\right).$$

2.1.2 Classification Trees

Tree-based machine learning methods recursively partition the feature space into a set of spaces, and then fit a simple model in each one. The partitioning is represented using a tree structure composed of internal decision nodes and terminal leaves. Each decision node implements a test function with discrete outcomes labelling the branches. To predict the outcome of an input the tree is traversed starting from the root until a leaf is reached, at which point the category assigned the leaf constitutes the output. During this process a test is applied at each node and depending on the outcome one of the branches is taken. Figure 2.1 shows an example of how the algorithm works in two dimensions. The features x_1 and x_2 should be used to distinguish objects of category C_1 from the objects of category C_2 . Firstly the space is split in two regions by a vertical line corresponding to the value w_{10} of feature x_1 and secondly the right region is again divided in two, but using an horizontal line corresponding to the value w_{20} of feature x_2 . The resulting regions each contain only one type or class of object and we can therefore build a decision tree that maps objects to a category by comparing their (x_1, x_2) coordinate with w_{10} and w_{20} . For example an object with x_1 coordinate smaller than w_{10} will fall to the left of the vertical line, therefore it will be assigned to C_1 because the condition in the root node is not satisfied.

For classification purposes, an impurity measure is used to quantify the goodness of a region split, thus determining the form of the tree. A split is said to be pure if after the split, for all branches, all the instances choosing a branch belong to the same class. In such a case there is no need to split any further and a leaf node with the corresponding label can be added to the tree. If a node is not pure then the instances should be split to decrease impurity until a stop criterion is reached.

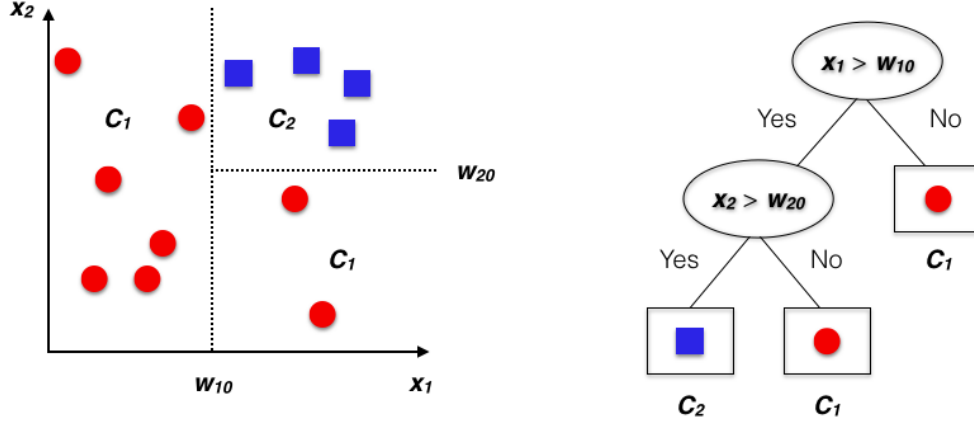


Figure 2.1: Example of Classification Tree construction. To the left we see the space separated into three different regions. To the right we see the corresponding binary tree.

2.2 K-Means Clustering

The goal of clustering algorithms is to partition a set of observed data points into groups, that are called *clusters*, such that the pairwise dissimilarities between those assigned to the same cluster tend to be smaller than those to data points located in different clusters [8]. K-Means clustering is one such algorithm. The number of clusters must be known a priori and the similarity between data points is expressed using some form of distance metric. One such example is the squared Euclidean distance. We denote the distance between points a and b with $d(a, b)$. Each cluster C is then represented by the mean μ_C , which is defined by

$$\mu_C = \frac{1}{|C|} \sum_{x \in C} x, \quad (2.1)$$

where $|C|$ is the number of points within cluster C . Because the mean represents the "middle" point of the cluster, it is also called centroid. The objective is to find the centroids that minimize the sum of all squared distances from all observations within each cluster. That is, for each cluster we want to find a mean μ such that

$$\arg \min_{\mu} \sum_{x \in C} d(x, \mu). \quad (2.2)$$

As seen in Algorithm 1, the first step consists of randomly selecting a centroid for each cluster. The next step assigns each data point to the nearest centroid. Then, all centroids are updated by computing the mean of all observed data points that are assigned to them using Formula 2.1. The difference between the

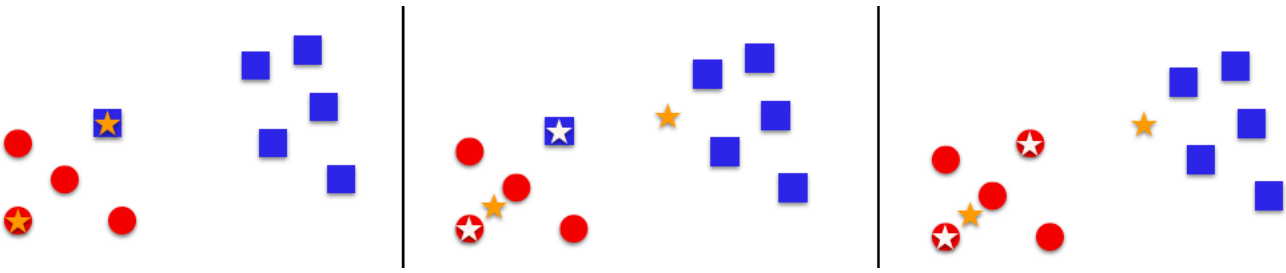


Figure 2.2: Example of how the K-Means Clustering algorithm works using two clusters on a set of two-dimensional data points. Firstly two points are randomly chosen as initial centroids (yellow stars) and the points are assigned to the nearest centroid. Secondly the within cluster mean is computed and used to update the centroids. Lastly the points are reassigned to the new nearest centroid.

new and the old values of the centroids are then computed and compared with a threshold to determine whether the algorithm should continue or it can be stopped because the centroids didn't move significantly. If this threshold is not met, we again assign data points to the newly set cluster centroids.

Algorithm 1: K-means Clustering

1. The centroids are initialized with random points from the data set.
 2. Every other point in the dataset is assigned to its nearest centroid according to the chosen distance measure.
 3. For each cluster the mean of all assigned points is computed and it becomes the new centroid.
 4. If the iteration count exceeded the maximum loop number or the a given convergence criterion is satisfied, terminate the algorithm, otherwise execute step 2.
-

Design

In this chapter we look at the two phases that can be distinguished in this work: firstly we describe methods for gathering structured recipe data and secondly we discuss our analysis schemes.

3.1 Data Acquisition

3.1.1 Collecting Recipes

A great number of websites provide cooking recipes of any kind. Sometimes they are published by professional cooks, sometimes by users with varying degree of expertise. We crawled the most popular websites in three different languages: English, German and Italian. This choice was taken because different cultures express different cuisine styles and, although thanks to the Internet spatial barriers are nowadays easy to break, the language barrier is still there. Also the exchange of recipes between cuisines is not free from modifications, for example a Lasagna dish will probably be cooked differently in the US compared to a traditional Italian Lasagna.


In total about 1M recipes were collected from the following websites:

English	German	Italian
www.allrecipes.com	www.chefkoch.de	www.giallozafferano.it
www.bbc.com	www.lecker.de	www.repubblica.it
www.chow.com		www.williamssonoma.com
www.epicurious.com		www.ricetedellanonna.it
www.food.com		www.saleepepe.it
www.foodnetwork.com		
www.tablespoon.com		
608136	327806	14750

3.1.2 Structuring Collected Data

Since most websites display the recipes as unstructured text, it is necessary to parse the webpages in order to extract any possible information that could afterwards be needed for our analysis. Straightforward examples are ingredients, directions and timings. Because we didn't know in advance all information that we would need from each recipe, we firstly downloaded the whole HTML code of each webpage, and only afterwards we parsed the code and extracted the sought information. In this process, the structure of each website was taken into consideration and the DOM tree of each webpage was traversed accordingly.

Figure 3.1 shows an example of how a webpage looks like, while Figure 3.2 shows website's HTML structure.



Bolognese Lasagna

Prep Time: 30 minutes Cook Time: 60 minutes Servings: 12

Make a big batch of Bolognese sauce and use it to prepare this hearty lasagna.

[Add to Recipe Box](#) [Email](#) [Print](#)

[G+1](#) | [67](#) [Tweet](#) [Pin it](#)

Ingredients:

- 16 lasagna sheets
- 4 cups ricotta cheese
- 2 eggs
- 1/4 cup finely chopped fresh flat-leaf parsley
- Salt and freshly ground pepper, to taste
- 6 cups [Bolognese sauce](#)
- 1 1/2 lb. mozzarella cheese, thinly sliced
- 2 cups grated Parmigiano-Reggiano cheese
- 3 Tbs. sliced fresh basil

Related Recipes

[Bolognese Sauce](#) ▶

Directions:

Preheat an oven to 375°F. Grease a 9-by-13-inch lasagna pan or baking dish with olive oil.

Line a baking sheet with paper towels. Bring a large pot of salted water to a boil over high heat. Add the lasagna sheets and cook according to the package instructions. Drain, rinse with cold water and drain again. Transfer to the prepared baking sheet and pat dry.

In a bowl, stir together the ricotta, eggs, parsley, salt and pepper.

Spread 1/4 cup of the Bolognese sauce in the bottom of the prepared pan. Lay 4 lasagna sheets, slightly overlapping, on the sauce. Spread 1 cup of the ricotta mixture evenly over the pasta. Spread one-fourth of the remaining sauce over the ricotta mixture. Arrange one-fourth of the mozzarella slices on the sauce, then sprinkle with 1/2 cup of the Parmigiano-Reggiano. Lay 4 more lasagna sheets, slightly overlapping, on top, and layer with another portion of the ricotta mixture, sauce, mozzarella and Parmigiano-Reggiano. Repeat the layering 2 more times, for a total of 4 layers. Sprinkle the basil evenly on top.

Cover the pan with aluminum foil and bake for 25 minutes. Uncover the pan and continue baking until the sauce is bubbling, 25 to 30 minutes more. Remove the pan from the oven and let rest, uncovered, for 15 minutes before serving. Serves 10 to 12.

Figure 3.1: Example of recipe webpage from *williams-sonoma.com*

Figure 3.4 shows some examples of most frequent ingredients found. A descrip-

Field	Content
Description	Recipe description
Name	Recipe's name
Author	Name of user who published the recipe
Ingredients	List of ingredients with their quantity
Prep_time	Preparation time
Cook_time	Cooking time
Tot_time	Total time
Directions	List of steps with instructions
Related_recipes	Urls to other recipes showed in the same webpage
Notes	Hints or author's comments
Reviews	Comments of other users

Figure 3.2: Field name and content description of a structured recipe.

tion of the same ingredient could in fact differ in many different ways, but most commonly we find the presence of adjectives or brand names and abbreviations for well-known ingredients. To tackle the problem we perform an N-gram analysis on all ingredient names. An N-gram is a sequence of N words within a sentence. Figure 3.3 shows all grams for "Condensed cream of mushroom soup". We compute the frequencies of all 1-, 2-, 3- and 4-grams found in all recipe ingredients by splitting all ingredient strings into separate words. We thus collect a big number of these N-grams which we store into our database along with their frequency of occurrence. Obviously not all the computed grams are ingredient names, therefore we removed adjectives and brand names from the ingredients before computing the grams to reduce the search space. Afterwards we filtered the database by removing non meaningful ingredient names and merging different grams that referred to the same ingredient, like plural and singular names.

"Condensed cream of mushroom soup"	
1-grams	"Condensed", "cream", "of", "mushroom", "soup"
2-grams	"Condensed cream", "cream of", "of mushroom", "mushroom soup"
3-grams	"Condensed cream of", "cream of mushroom", "of mushroom soup"
4-grams	"Condensed cream of mushroom", "cream of mushroom soup"
5-gram	"Condensed cream of mushroom soup"
6-gram	{}

Figure 3.3: N-grams of the string "Condensed cream of mushroom soup"

Count	Ingredient	Different descriptions
20142	salt	salt, sea salt, Hawaiian sea salt, ground sea salt
18674	pepper	pepper, lemon pepper, white pepper, seasoned pepper
18015	sugar	sugar, confectioners' sugar, granulated sugar, superfine sugar
13204	butter	butter, butter, softened, unsalted butter, frozen
12485	oil	oil, oil for frying, vegetable oil, olive oil, dark sesame oil
11345	cheese	mozzarella, mozzarella cheese, cubed Cheddar cheese
10864	flour	flour, all-purpose flour, rice flour, pastry flour
10566	garlic	garlic, minced garlic, McCormick® Garlic Powder
10202	onion	onion, finely diced onion, white onion, chopped Spanish onion
8681	water	warm water (110 degrees F), carbonated water

Figure 3.4: Number of occurrences, name and some different descriptions for the ten most frequent ingredients found in recipes from *allrecipes.com*.

3.2 Recipe Difficulty Classification

Cooking is a process that involves many different tasks and skills. Since a recipe describes such process, it can be seen as an algorithm to produce some elaborate food. Humans can easily distinguish between different recipe complexity levels. Unfortunately, computers don't have that advantage. Consider for example preparing some "Cucumber Sandwiches" against cooking a "Rosemary Turkey Roast". Already from the recipe name and without even looking at the recipe description, a lot of people would say that the roast is more difficult than the sandwiches because of their past experiences.

3.2.1 Developing an Objective Method

Our goal is to find an objective measure to express the difficulty of a recipe, which doesn't depend on our personal perception, but only from the recipe itself. But which recipe data should we consider? Well, a recipe requiring many different tasks in its preparation might be more complex than another one requiring just a few steps, thus we choose to consider the directions in determining the recipe difficulty. The time needed to prepare the recipe might also provide us with some useful information about the difficulty. Thus we consider preparation and total time as well. Lastly, we take into consideration the number of ingredients that are required for the recipe, because the more ingredients are needed, the bigger the effort to collect them and to organize the working environment.

We assume the difficulty to be linearly dependant to the chosen parameters and we consider three combinations of parameters shown in Figure 3.1 as features for our classification.

	Number of conjunctors	Number of dots	Preparation time	Total time	Number of ingredients
Method 1	x		x		x
Method 2	x			x	x
Method 3		x		x	x

Table 3.1: Parameters considered in the different methods. Conjunctors refer to elements from the set $\{".", ",", ":", ";", "and", "or"\}$. Both number of conjunctors and of dots are counted in the recipe description. Time is measured in minutes.

3.2.2 Assessing the validity of our methods

As ground truth we took *chefkoch.de* recipes which contain user-assigned difficulty levels and we used *Naive Bayes* and *Decision Trees* classification methods to determine how well these features label the difficulty level. For both methods the training sets were generated randomly by taking 90% of the considered recipes, while the remaining 10% were used as the test set. Furthermore different seeds for the random generator were used to build different training sets.

3.3 Discovering Recipe Similarities

For centuries, different cultures have developed different cuisines. With increased globalisation the exchange between different cuisines has grown rapidly and consequently we have the possibility to eat food and buy ingredients that were previously completely unknown to us. But does what we believe to know really correspond to the reality? And are there similarities or differences which we don't know and maybe won't ever imagine to be possible? In this section we aim to investigate differences and similarities between different kinds of recipes. We therefore apply two different machine learning algorithms to recipe data retrieved from the *allrecipes.com* website.

3.3.1 Recipe Similarity using K-Means Clustering

Similarly to what was done in [6], we define each recipe as a point in k-dimensional Euclidean space, each dimension representing an ingredient. We consider only the most frequent ingredients along our dataset instead of all ingredients and we extended the representation to consider also the ingredient quantity (for example ml or grams) per serving, instead of considering only the presence or absence of the ingredient in the recipe. Each recipe is then represented as a vector, where the value at each index to an ingredient and it is set to 0 if the ingredient is not present.

In order to better visualise the data we use a dimensionality reduction scheme, which is particularly well suited for embedding high-dimensional data into two or three dimensions. To visually assess the quality of the clustering we color the points according to the cuisine or the category they belong to. The cuisine and category information is taken from *allrecipes.com* and is shown in Figure 3.5.

Category	Macro-cuisine	Cuisines
Appetizer	African	North African / South African
Breakfast	Asian	Chinese / Filipino / Indian
Dessert		Japanese / Korean / Thai / Vietnamese
Salad	Canadian	
Bread	European	East European / French / German / Greek / Italian
Sauce and Condiment		Portuguese / Scandinavian / Spanish / English and Irish
Side dish	Latin American	Caribbean / Mexican / South American
Soup	Middle East	
Drink	US	Amish / Cajun / Jewish / New England / Soul Food
Chicken		Southern
Main dish	Australian and New Zealand	

Figure 3.5: List of different category, macro-cuisine and cuisine types.

3.3.2 Recipe Similarity using Classification Trees

With this approach we aim to find the best tree that shows which ingredients are most relevant when deciding if a recipe is in a specific category or cuisine. Therefore we consider again the space defined by the most frequent ingredients and we build a classification tree using all recipes. The internal nodes of such tree will then embed a binary condition on a single ingredient, for example $beef \leq 0.5$, meaning that if the condition is satisfied we take one branch and otherwise we take the other. This reflects whether an ingredient is present or not for the unweighted case and also in what quantity for the weighted case. Each leaf includes all recipes that satisfy all conditions along the path from the root down to the leaf itself. By looking at the tree's final structure we want to see which ingredients are most typically present in a subset of recipes but not in others, and can therefore be used to distinguish such recipes.

Implementation

In this chapter we intend to explain the tools used throughout this project as well as to give an overview of the scripts implemented for the analysis.

Every script was implemented using Python. To crawl the different websites we used the modules *requests* [9] for simple HTTP requests and *selenium* [10] when we had to programatically interact with the web browser because some JavaScript code had to be executed on the target webpage. To parse and search through the downloaded HTML code, we used *Beautiful Soup 4*, which provides idiomatic ways of navigating, searching, and modifying the parse tree [11]. As Python 2.7 still has ASCII default encoding and Unicode is the most frequent encoding used over the Internet we incurred in some decoding errors while parsing the downloaded HTML code. To "repair" the broken code we used some functionalities provided by *ftfy* [12].

To extract information from string we widely use regular expression functionalities provided by the *re* module.

For the tree classification, the recipe clustering and the difficulty classification we used algorithms provided by the *scikit-learn* [13] module, while *numpy* [] was used for building and handling matrices.

For the database we preferred *MongoDb* [14] over *MySQL* [15] because it allowed us to keep a really flexible database schema.

Results

In this chapter we list our results of the performed recipe difficulty classification, and in the end, qualitative observations on category and cuisine similarities are presented.

5.1 Difficulty Classification Results

Figure 5.1 shows the average values measured during the process explained in Section 3.2 with 4 different random generator seeds. Overall both average preci-

	Precision		Recall	
	Tree	Bayes	Tree	Bayes
method 1	0.66	0.66	0.67	0.65
method 2	0.65	0.64	0.65	0.63
method 3	0.60	0.65	0.60	0.63

Figure 5.1: Average precision and recall values for the different methods and algorithms.

sion and average recall are above 60%, which is not really bad for a first attempt with simple algorithms. We observe that the difference between using *Tree* classification and *Gaussian Naive Bayes* is minimal, being at most 5%. With at most 2% of difference, the same holds for the first two methods, meaning that preparation time and total time have a similar influence on the recipe difficulty. The third method however performs a little worse than the others, thus we can conclude that by more precisely analysing the directions we can improve the accuracy a bit.

However, if we look at the single values shown in Figures 5.2 and 5.3, we have a different picture: it is clearly visible that the first difficulty level has the best score and that the percentage decreases as the difficulty increases. Also interesting is that underestimation is much more frequent than overestimation and that, for the most part, all wrong matches are mislabelled into the difficulty level just

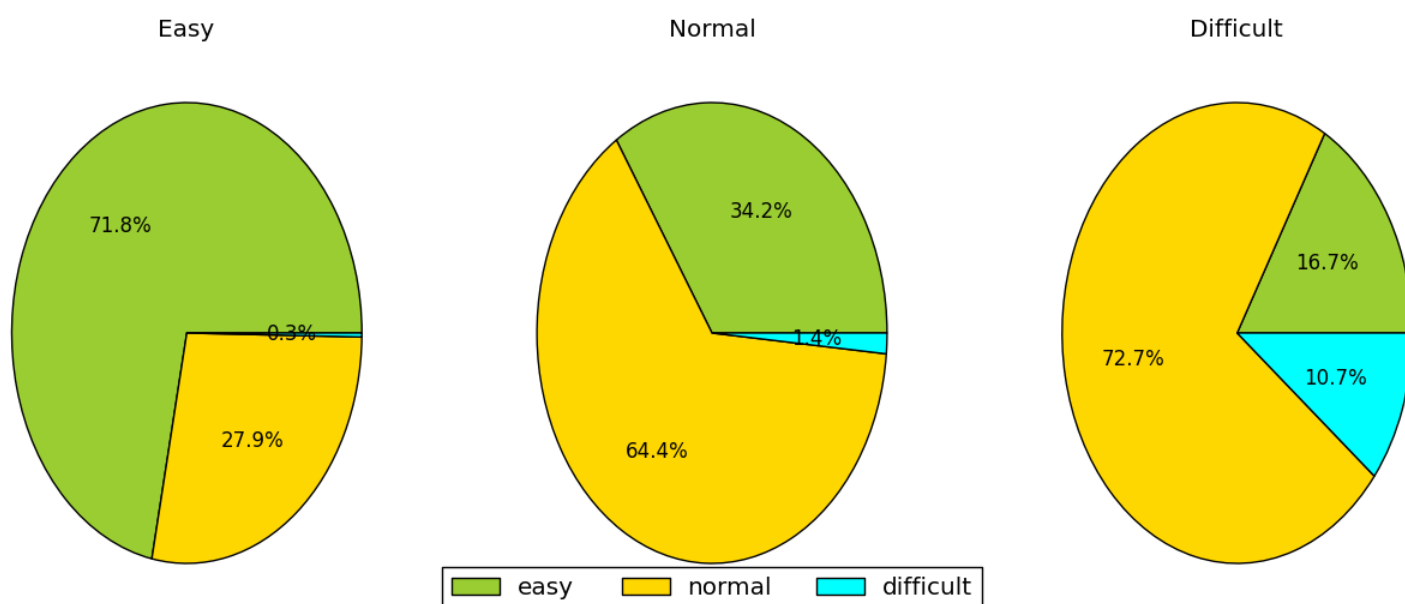


Figure 5.2: Pie plots with recall values for each difficulty level of one classification run using Tree Classification with method 1.

one below the expected one. Furthermore there is a more significant difference between the two algorithms than in the average case, concerning the accuracy for the first two levels: Tree classification has lower recall for easy recipes than Naive Bayes, but it kind of compensates with a higher percentage for the normal recipes.

There are many factors that could have influenced this results and that could be taken into consideration to improve the classification. First of all the difficulty labels are given from the user who posted the recipe, which means that they are subjective and perhaps skewed. Considering that it's more likely that we share recipes we like or we are familiar with, and that people tend to underestimate the difficulty of a task or to overestimate their skills (Dunning-Kruger Effect [16]), it is necessary to assume that the dataset is biased. This could be an explanation to the curious one-level underestimation problem we noticed before and also to the low number of recipes labelled as difficult.

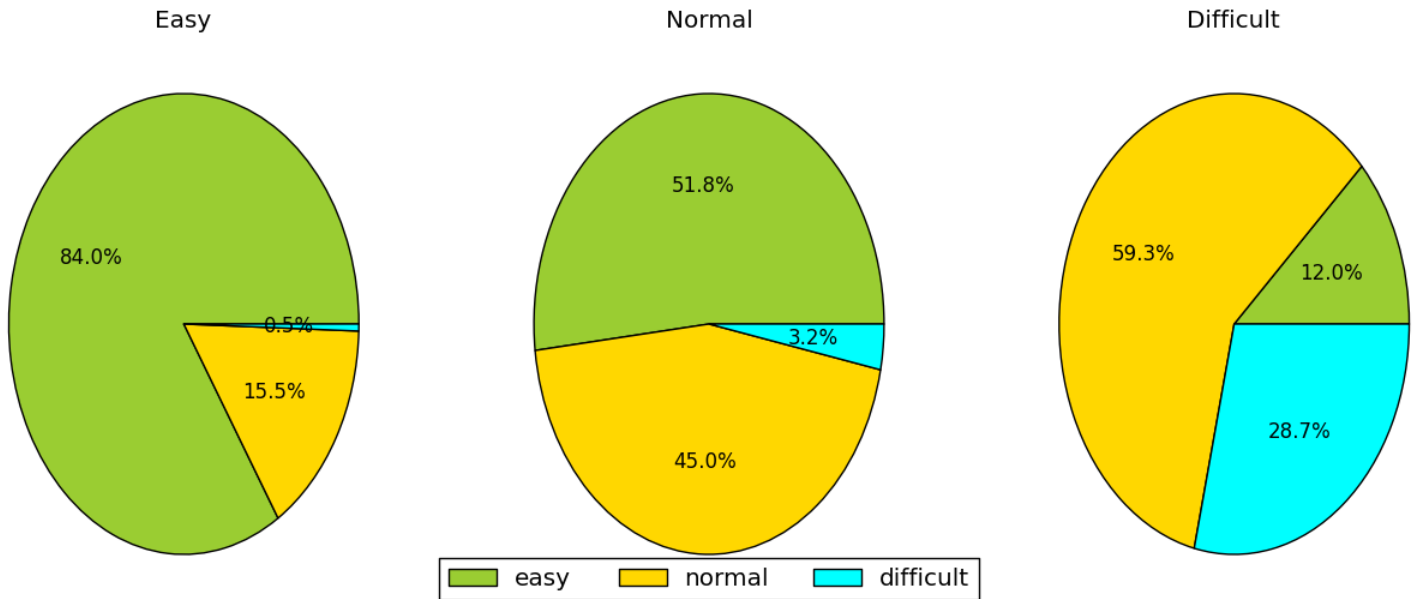


Figure 5.3: Pie plots with recall values for each difficulty level of one classification run using Gaussian Naive Bayes Classification with method 1.

5.2 Category Similarities

It is a simple task for humans to separate food into different categories. For example appetizers, main dishes, side dishes, etc. are eaten in different contexts and at different times of the day, therefore they may have a different structure. Our goal is to find similarities and differences between recipes of different categories by looking only at their ingredients.

5.2.1 K-Means Clustering

To begin we consider the following 4 categories:

- appetizer
- side dish
- main dish
- dessert

We would expect to see big differences between desserts and the other three categories. We would also consider appetizers to stand out a little from side and main dishes, while we expect side and main dishes to share most characteristics. Figures 5.4 and 5.5 show the clustering of all the recipes belonging to these four categories, where each category is represented using a different color. In the first figure we used a scatter plot and in the second a pie plot for each cluster. Figure 5.6 shows the three most frequent ingredients of each cluster and the corresponding pie plot.

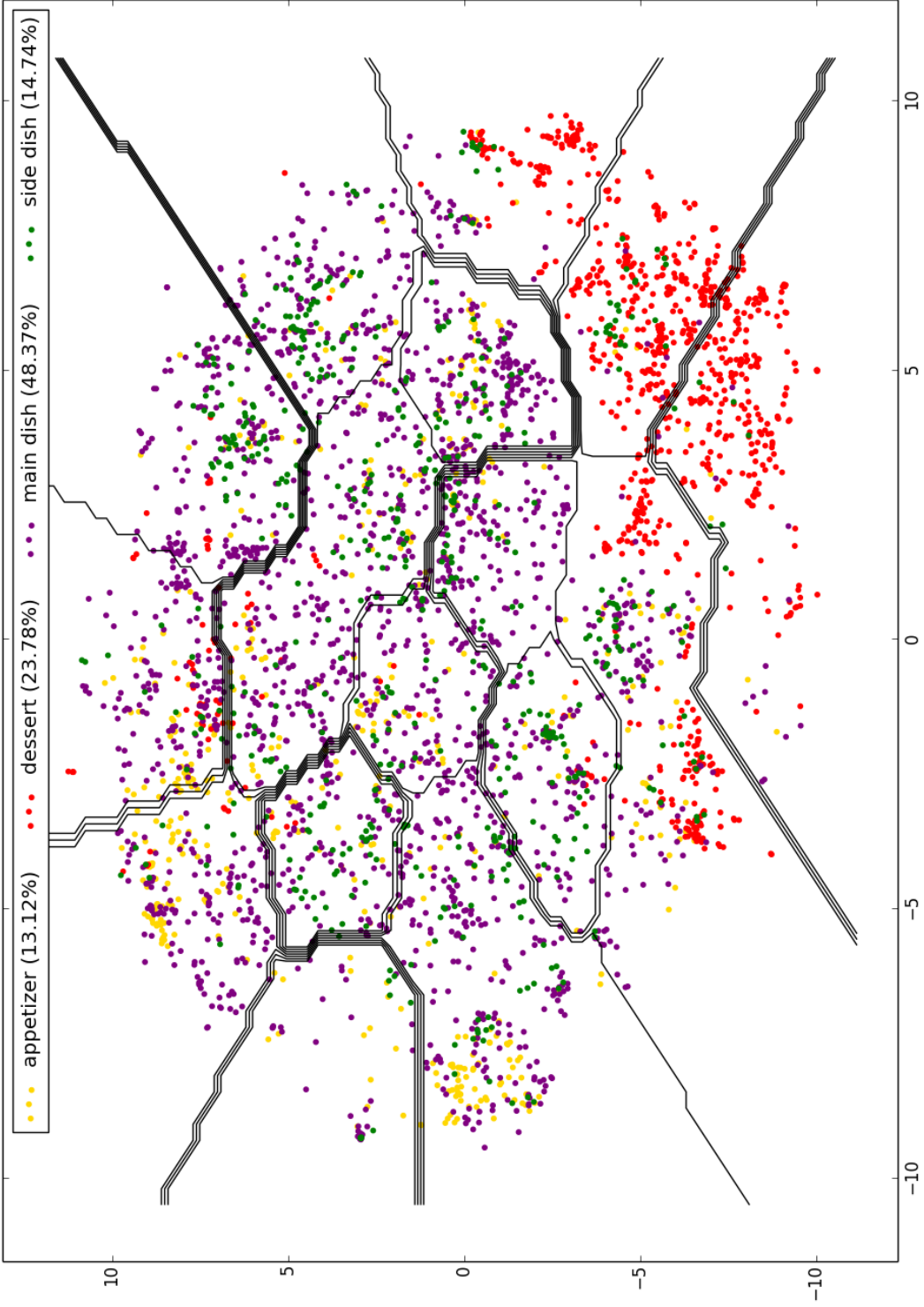


Figure 5.4: Scatter plot representing distribution of appetizer, side dish, main dish and dessert recipes.

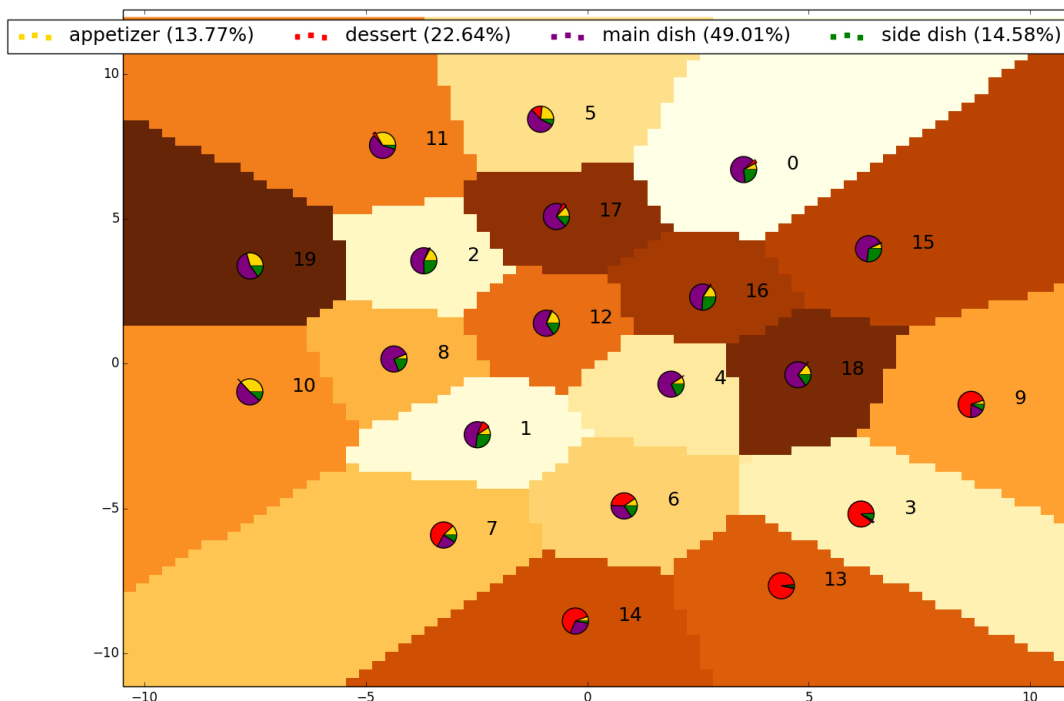


Figure 5.5: K-Means clustering of appetizer, dessert, main dish and side dish categories with 20 clusters. Each cluster is numbered and contains a pie plot with the distribution of contained recipes among the four categories.

The first observation we can make is that desserts are actually well separated from the other dishes because they mostly cover a very distinct area of the space. Main and side dishes are more distributed and cover a wider area without big concentrations in specific areas. Appetizers are mixed with main and side dishes, but are more concentrated in two areas which lie further away from desserts. Finally there is no visible separation between main and side dishes. These observations are even more clear by looking at the pie representation, where we can also see that the most frequent ingredients in desserts are white sugar, butter and flour. In contrast for the other dishes we find more often spices (like black pepper and salt), garlic, onion, cheese and oil.

Interestingly, we have a pretty different picture if we consider also the quantity of the ingredients. As it can be seen in Figure 5.7 the clear distinction between desserts and the other categories is no more visible, although we can see from Figure 5.8 that the most frequent ingredients for desserts are mostly the same. This is not true for the rest of the recipes, where we can notice new most frequent

Cluster	Pie plot	Most frequent ingredients
0		butter (68%) / black pepper (35%) / parmesan cheese (34%)
1		water (97%) / salt (54%) / onion (33%)
2		vegetable oil (73%) / onion (57%) / black pepper (44%)
3		white sugar (94%) / butter (88%) / flour (61%)
4		garlic (94%) / olive oil (87%) / salt (77%)
5		chicken (22%) / cheese (15%) / cream (15%)
6		salt (69%) / flour (65%) / egg (49%)
7		white sugar (90%) / water (60%) / salt (41%)
8		garlic (91%) / salt (82%) / onion (66%)
9		butter (87%) / flour (61%) / salt (53%)
10		cilantro (91%) / salt (72%) / garlic (62%)
11		cheddar cheese (58%) / sour cream (48%) / onion (32%)
12		salt (84%) / onion (48%) / black pepper (41%)
13		flour (96%) / white sugar (89%) / salt (77%)
14		salt (94%) / flour (83%) / water (83%)
15		butter (97%) / salt (90%) / onion (48%)
16		olive oil (85%) / black pepper (66%) / salt (52%)
17		garlic (51%) / onion (38%) / soy sauce (32%)
18		olive oil (92%) / garlic (90%) / basil (52%)
19		salt (69%) / black pepper (54%) / garlic powder (52%)

Figure 5.6: Three most frequent ingredients and pie plot of all clusters from Figure 5.5.

ingredients like beef and shrimp. Also in many clusters we now have lower percentages for the most frequent ingredients, which indicates that the clustering is worse.

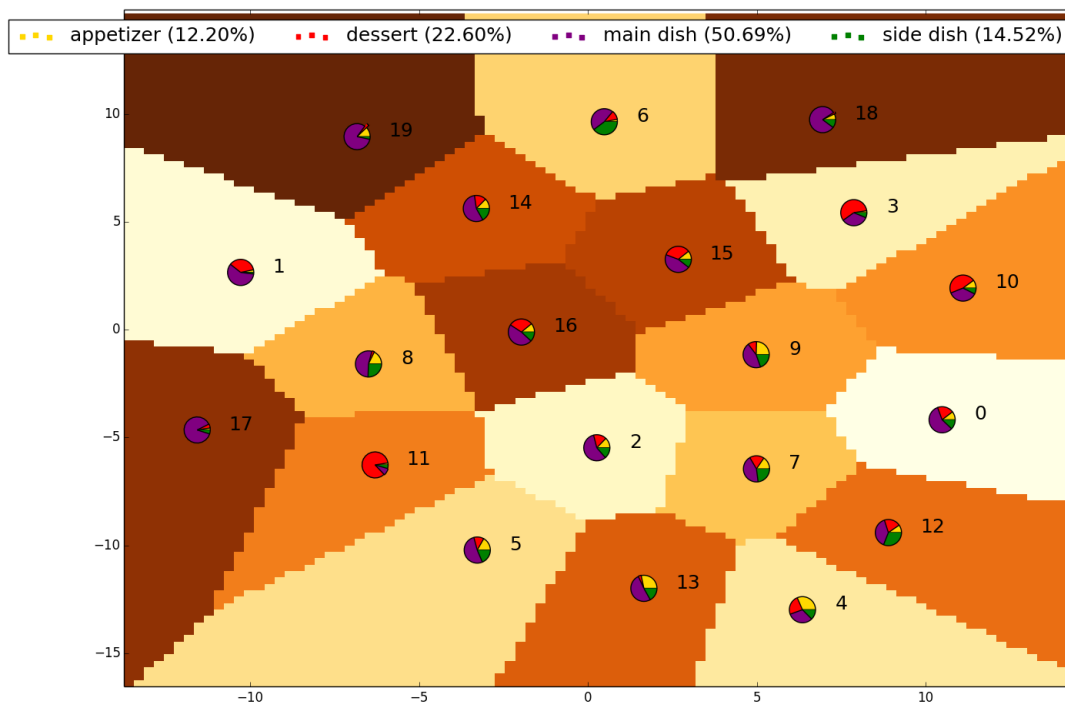


Figure 5.7: K-Means clustering of appetizer, dessert, main dish and side dish categories with 20 clusters. The weight of the ingredients is also considered. Each cluster is numbered and contains a pie plot with the distribution of contained recipes among the four categories.

5.2.2 Tree Classification

We again consider the categories of Subsection 5.2.1, but this time we use classification trees. Figure 5.9 shows a tree of depth 4 of recipes from only the main dish and dessert categories from which it can be clearly be seen that, as we would expect, if a recipe contains sugar it is most likely a dessert. On the other hand by looking at the two internal nodes that test for the presence of garlic we also clearly see that both generated leaves representing recipes containing garlic are pure and contain only main dishes.

Cluster	Pie plot	Most frequent ingredients
0		water (35%) / pork (33%) / garlic (30%)
1		beef (47%) / flour (47%) / white sugar (35%)
2		cream (31%) / butter (31%) / onion (30%)
3		flour (70%) / butter (50%) / white sugar (43%)
4		white sugar (22%) / butter (17%) / garlic (15%)
5		vegetable oil (40%) / onion (34%) / flour (28%)
6		water (97%) / onion (33%) / garlic (30%)
7		garlic (41%) / olive oil (41%) / butter (23%)
8		onion (69%) / garlic (37%) / olive oil (66%)
9		garlic (36%) / butter (39%) / onion (29%)
10		flour (60%) / butter (35%) / white sugar (34%)
11		flour (76%) / white sugar (74%) / butter (64%)
12		butter (55%) / brown sugar (22%) / olive oil (19%)
13		garlic (30%) / vegetable oil (21%) / olive oil (21%)
14		garlic (29%) / water (29%) / butter (27%)
15		butter (41%) / milk (30%) / white sugar (29%)
16		butter (33%) / flour (28%) / white sugar (27%)
17		pasta (59%) / garlic (56%) / olive oil (43%)
18		shrimp (95%) / garlic (52%) / butter (38%)
19		beef (92%) / onion (42%) / water (34%)

Figure 5.8: Three most frequent ingredients and pie plot of all clusters from Figure 5.7.

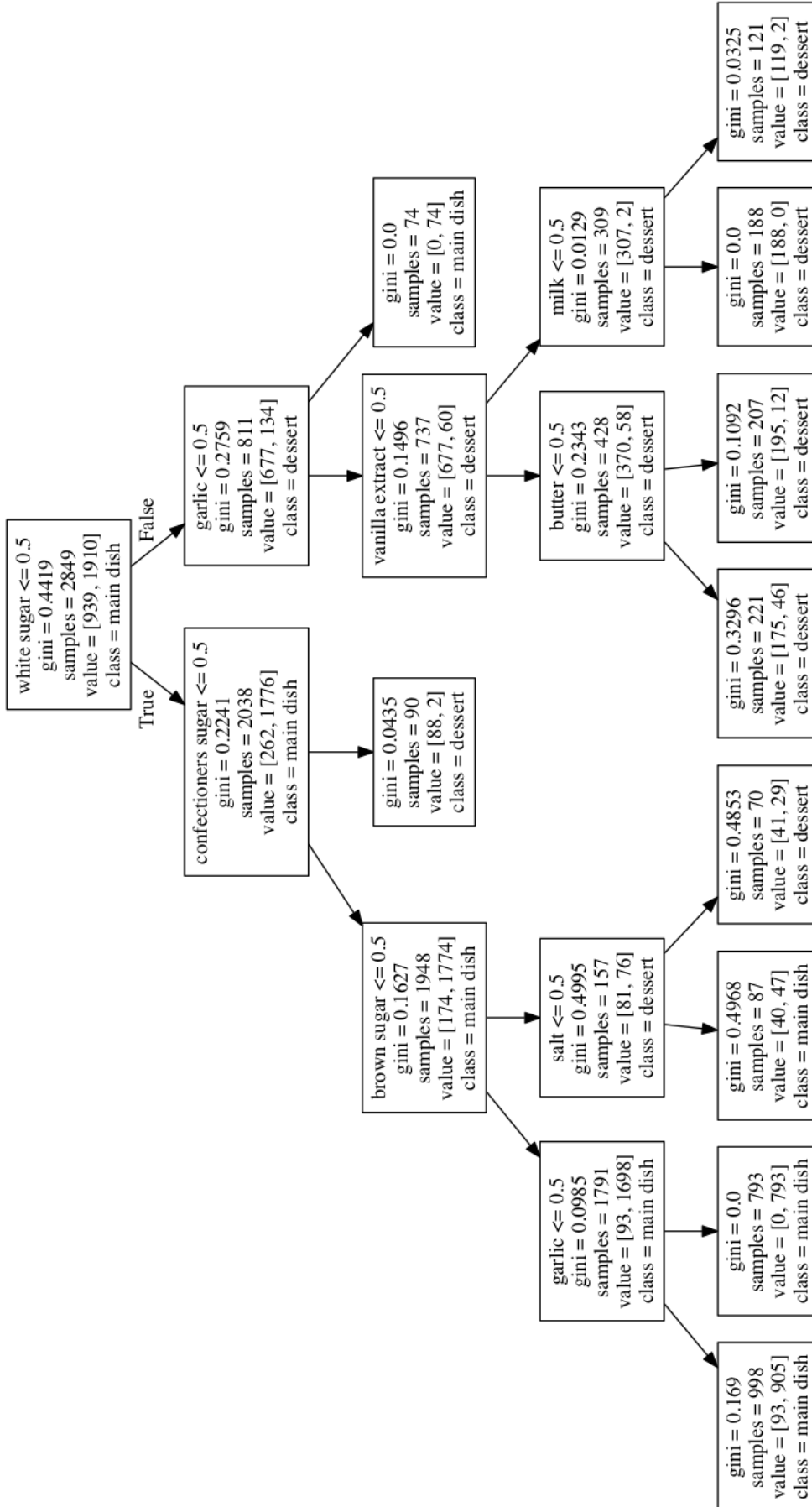


Figure 5.9: Classification tree with depth 4 of recipes from the main dish and dessert categories.

In Figure 5.10 we have a tree of depth 5 build using recipes from appetizer, main and side dishes categories. Although the numbers are quite skewed we can still see that the presence of meat or pasta is an indicator for main dishes, while cream cheese is most frequently found in appetizers, but apart from this we can't really see any ingredient that distinguishes one category from the others.

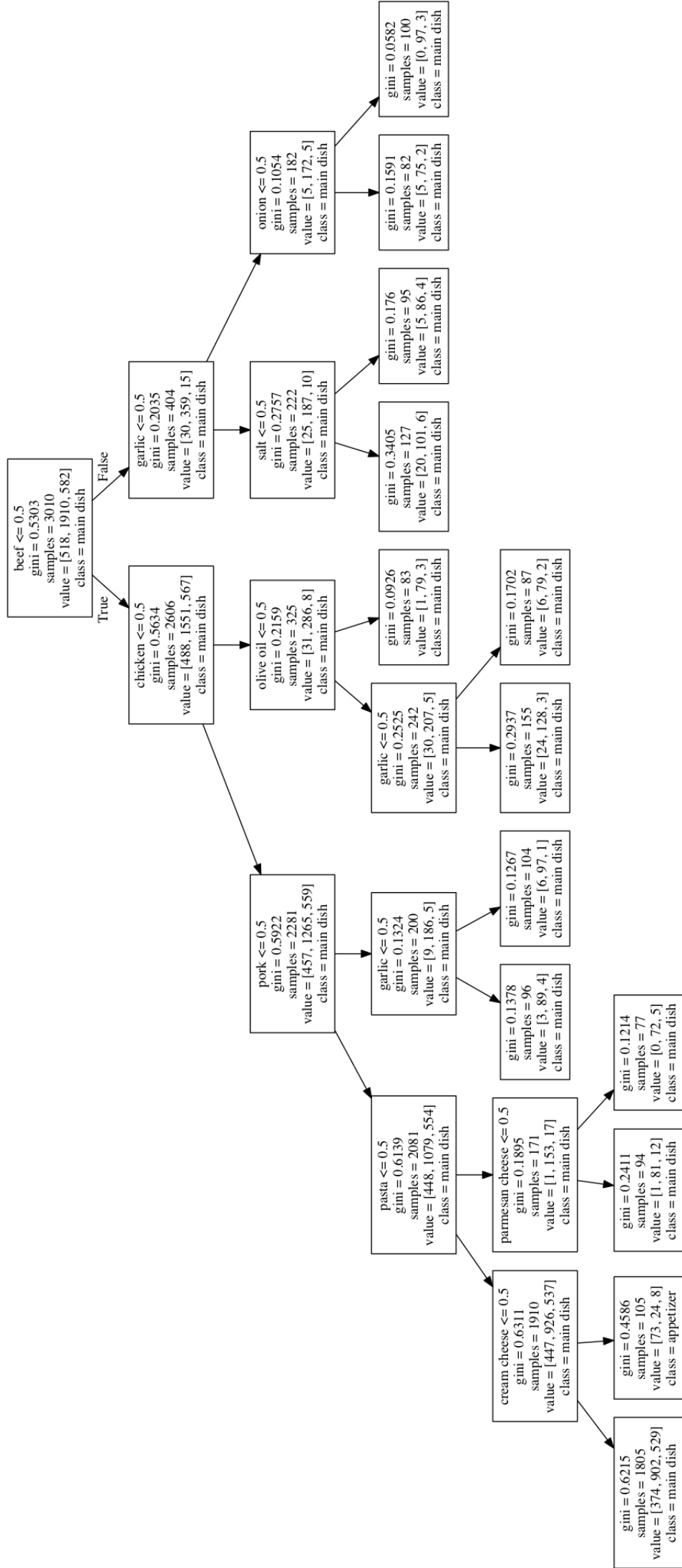


Figure 5.10: Classification tree with depth 5 of recipes from the appetizer, side dish and main dish categories.

5.3 Cuisine similarities

People somewhat know what types of dishes they can expect when they go to a Chinese restaurant as opposed to an Italian restaurant. We now use the same approach with world cuisines, to see for example if it is true that the Asian cuisine is really different from the European cuisine.

5.3.1 K-Means Clustering

By looking at Figures 5.11 and ??, which show the distribution of different cuisines in the ingredient space, we can already make the following observations:

- European and Latin American cuisines are mostly separated;
- US, Australian and Canadian cuisines cover the whole space;
- the Asian cuisine is scattered between Latin American and European cuisines.

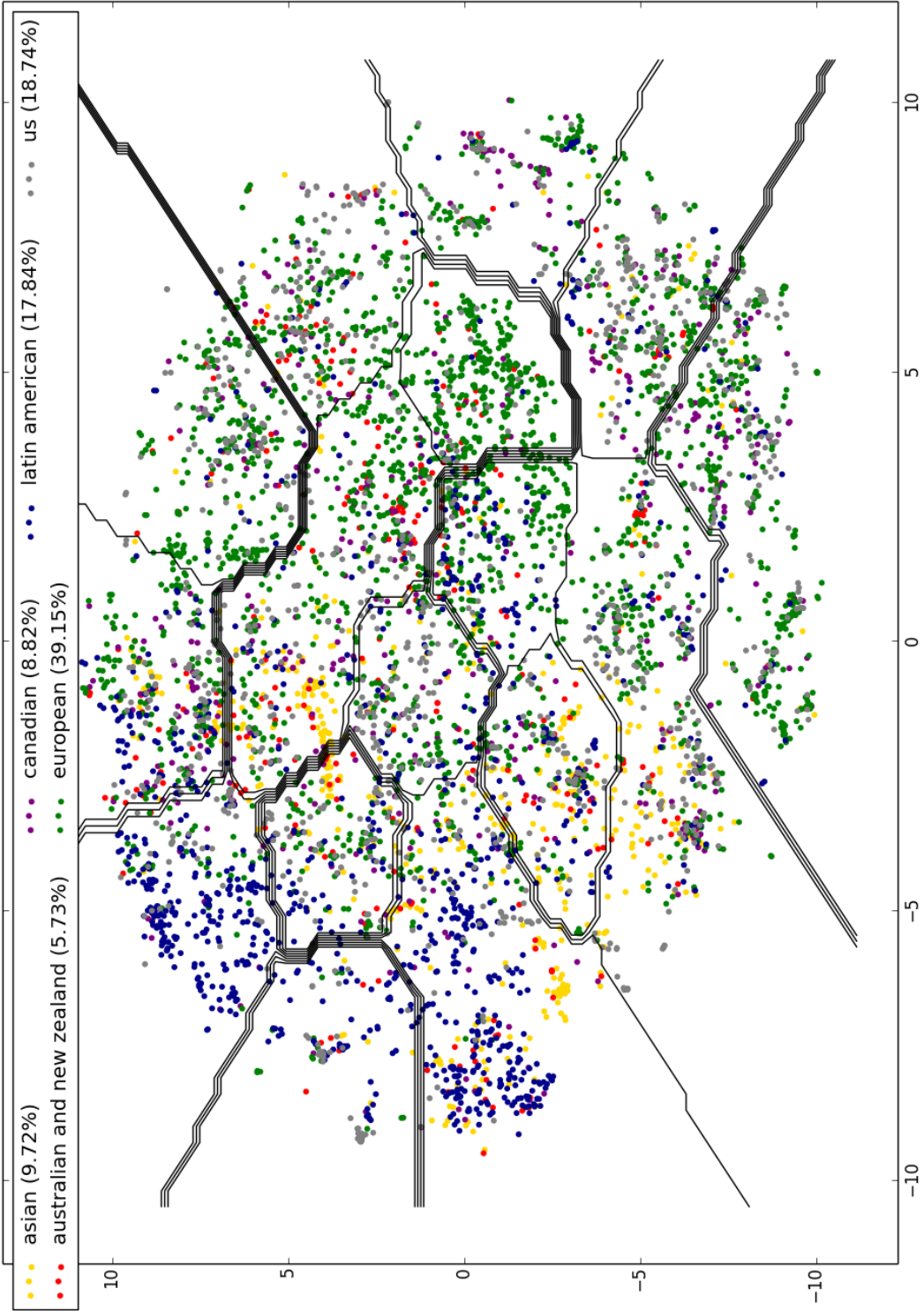


Figure 5.11: Scatter plot representing distribution of recipes from macro-cuisines but African and Middle East.

The separation of Latin American and European is interesting a little surprising, because for historical reasons we would expect more connections, since Europeans spread their culture in those regions and also brought back new ingredients to Europe. This phenomenon could nevertheless be an explanation for the many similarities between US and European cuisines.

Since the European macro-cuisine not only covers a big area in the ingredient space, but also includes a lot of different cuisines, like Italian, French and so on, we want to get a closer look. We consider the well-known Mediterranean cuisine, which includes Spanish, Portuguese, Italian and Greek cuisines. Figures 5.12 and 5.13 show their distribution in the ingredient space using scatter and pie plots.

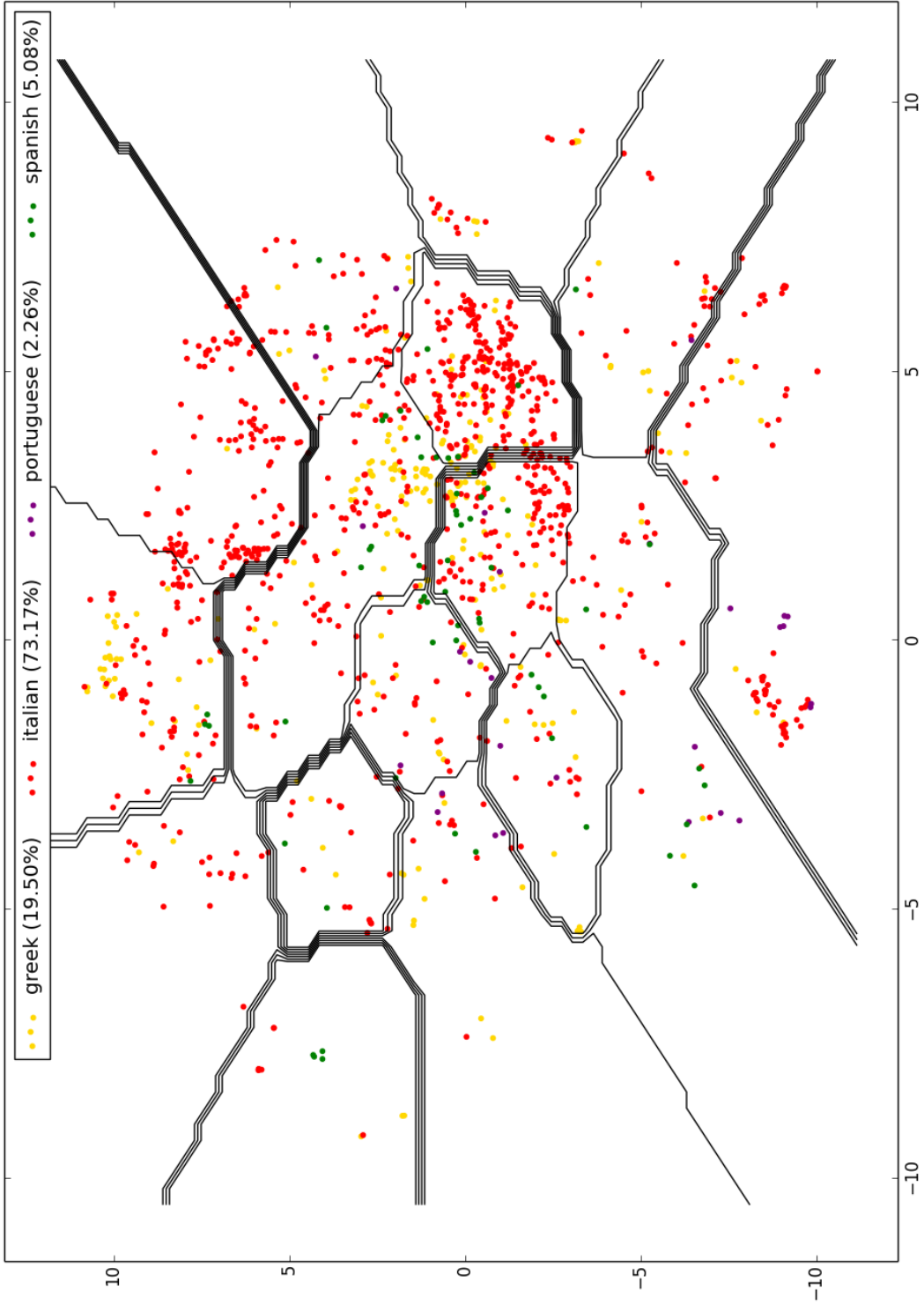


Figure 5.12: Scatter plot representing distribution of recipes from Mediterranean cuisines.

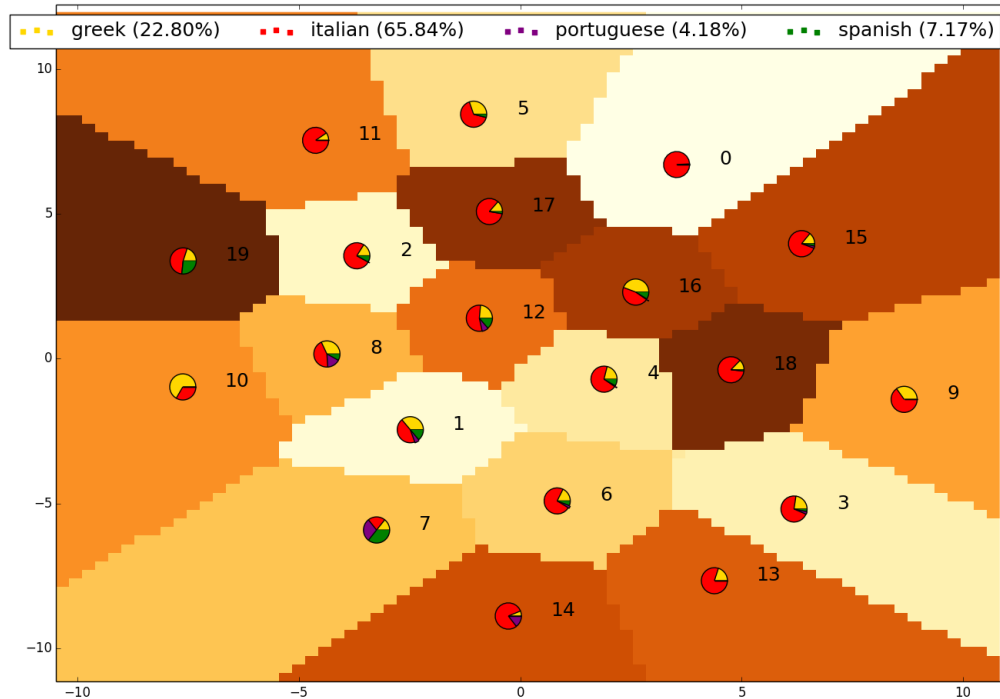


Figure 5.13: mediterranean pie

As we would expect, it is possible to notice that there are many overlaps, in particular between Greek and Italian cuisines. Because they occur less frequently, it is difficult to tell whether the same holds also for Spanish and Portuguese cuisines, but we can see that they are not concentrated in a single region. By looking at Figure 5.14 it is also interesting to see that cluster 0, which has Parmesan cheese, mozzarella and butter as most frequent ingredients, contains for the vast majority Italian recipes. Just nearby there is the region (clusters 4, 16, 18) with the highest concentration of recipes from all but the Portuguese cuisine. It is clearly visible from Figure 5.14 that most recipes mapped there contain olive oil, which is one of the characteristic ingredients of the Mediterranean diet.

5.3.2 Tree Classification

Here we want to investigate the Mediterranean cuisine a little further to see if there are ingredients that aren't shared and can be considered as typical for one

Cluster	Pie plot	Most frequent ingredients
0		parmesan cheese (63%) / mozzarella (50%) / butter (47%)
1		water (86%) / salt (72%) / garlic (33%)
2		onion (72%) / black pepper (60%) / vegetable oil (56%)
3		white sugar (97%) / butter (74%) / flour (68%)
4		garlic (92%) / olive oil (78%) / salt (71%)
5		salad dressig (31%) / chicken (30%) / mozzarella (28%)
6		egg (59%) / flour (56%) / salt (50%)
7		white sugar (100%) / water (50%) / lemon (36%)
8		slat (92%) / garlic (84%) / onion (56%)
9		flour (96%) / butter (91%) / salt (48%)
10		garlic (100%) / salt (100%) / cilantro (100%)
11		cheddar cheese (61%) / beef (48%) / onion (38%)
12		salt (75%) / onion (55%) / garlic (37%)
13		white sugar (95%) / flour (93%) / baking powder (65%)
14		salt (96%) / yeast (96%) / water (93%)
15		butter (93%) / salt (86%) / black pepper (52%)
16		olive oil (83%) / black pepper (78%) / salt (51%)
17		garlic (59%) / beef (43%) / onion (38%)
18		olive oil (92%) / garlic (86%) / basil (60%)
19		lemon (47%) / parsley (40%) / basil (40%)

Figure 5.14: Three most frequent ingredients and pie plot of all clusters from Figure 5.13.

or more of the cuisines. By looking at Figure 5.15 we see that feta is an ingredient present mostly in Greek recipes, as we would expect since it is originally from Greece. But there is also a certain number of Italian recipes that contain it, which could mean that either those recipes are wrongly labelled as Italian recipes despite being Greek recipes, or that there was at one point an exchange between the cuisines. We also see that parmesan cheese plays a major role in Italian recipes. Other ingredients that are clearly typical only for the Italian cuisine are basil, mozzarella and cheese (which includes many less known qualities).

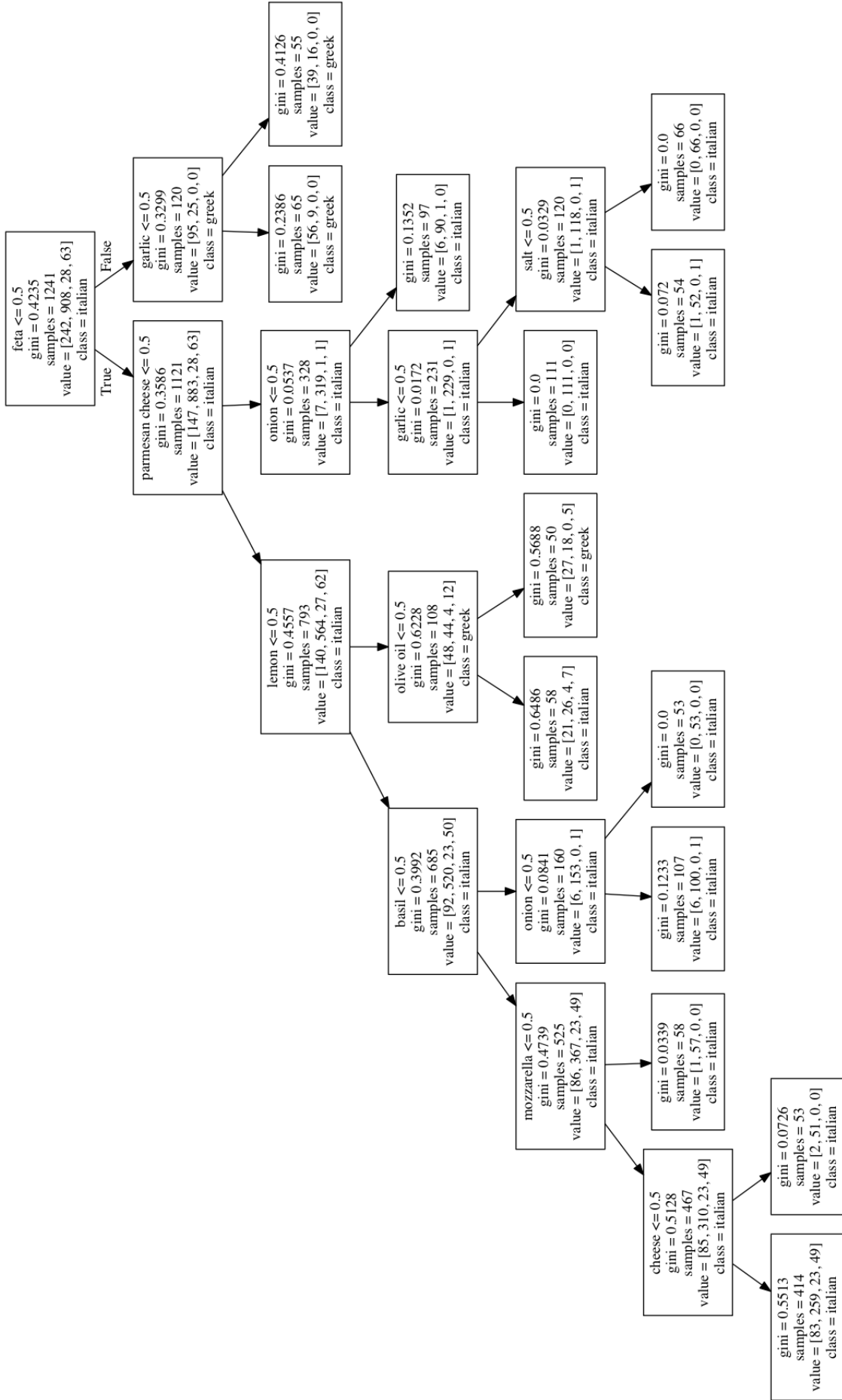


Figure 5.15: Classification tree with depth 6 of recipes from Mediterranean cuisines.

Outlook and Summary

In this thesis we collected a great number of recipes and developed a large database of structured recipes. Additionally, we proposed some possible models for determining the difficulty of a recipe from its description and we used *Gaussian Naive Bayes* and *Decision Tree* classification algorithms in order to classify recipes into different difficulty categories. Furthermore we used *K-Means Clustering* and *Classification Trees* to search for similarities and differences in the ingredient space between recipes of different categories and world cuisines. A possible future work could be to improve the recipe database to obtain a uniform structure independently from source and language of the data, as well as including all ingredients into the analysis and not only the most frequent ones. Another possible future work could be to collect more information, like cost and nutritional value, for separate ingredients, which would allow us to develop a healthy recipe suggestion app for users who want assistance in eating more healthily. To improve the difficulty analysis, we could consider more parameters like the presence of specific ingredients or the separate words in the cooking direction steps. This could be done with a grammatical analysis to count the number of actions needed and give a higher weight to more involving actions. This work has laid the foundation for further recipe studies by generating a large collection of structured recipe data.

Bibliography

- [1] : Allrecipes: homepage. <http://allrecipes.com/> [Accessed: 2015-12-17].
- [2] : Epicurious: homepage. <http://www.epicurious.com/> [Accessed: 2015-12-17].
- [3] Svensson, M., Höök, K., Cöster, R.: Designing and evaluating kalas: A social navigation system for food recipes. *ACM Transactions on Computer-Human Interaction (TOCHI)* **12**(3) (2005) 374–400
- [4] Ueda, M., Takahata, M., Nakajima, S.: User’s food preference extraction for personalized cooking recipe recommendation. *Semantic Personalized Information Management: Retrieval and Recommendation SPIM 2011* (2011) 98
- [5] Teng, C.Y., Lin, Y.R., Adamic, L.A.: Recipe recommendation using ingredient networks. In: *Proceedings of the 4th Annual ACM Web Science Conference*, ACM (2012) 298–307
- [6] Nedovic, V.: Learning recipe ingredient space using generative probabilistic models. In: *Proceedings of Cooking with Computers Workshop (CwC)*. Volume 1. (2013) 13–18
- [7] Su, H., Shan, M.K., Lin, T.W., Chang, J., Li, C.T.: Automatic recipe cuisine classification by ingredients. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ACM (2014) 565–570
- [8] Friedman, J., Hastie, T., Tibshirani, R.: *The elements of statistical learning*. Volume 1. Springer series in statistics Springer, Berlin (2001)
- [9] : Requests: homepage. <http://docs.python-requests.org/en/latest/> [Accessed: 2015-12-17].
- [10] : Selenium: homepage. <http://www.seleniumhq.org/> [Accessed: 2015-12-17].
- [11] : BeautifulSoup: homepage. <http://www.crummy.com/software/BeautifulSoup/> [Accessed: 2015-12-17].
- [12] : Ftfy: Github repository. <https://github.com/LuminosoInsight/python-ftfy> [Accessed: 2015-12-17].

- [13] : Scikit-learn: homepage. <http://scikit-learn.org/stable/> [Accessed: 2015-12-17].
- [14] : MongoDB: homepage. <https://www.mongodb.org/> [Accessed: 2015-12-17].
- [15] : MySQL: homepage. <https://www.mysql.com/> [Accessed: 2015-12-17].
- [16] Dunning, D., Johnson, K., Ehrlinger, J., Kruger, J.: Why people fail to recognize their own incompetence. *Current Directions in Psychological Science* **12**(3) (2003) 83–87
- [17] : Selfnutritiondata: homepage. <http://nutritiondata.self.com/> [Accessed: 2015-12-17].

Appendix Chapter

Cooking measure	Weight (grams)
cup	125
teaspoon	4
tablespoon	12
clove	8
pinch	2
dash	5
cube	15
slice	12
bunch	3

Figure 7.1: Conversion table for different non-standard measures found in ingredient descriptions.

ingredient	Weight (grams)
egg	53
(red) onion	110
zucchini	196
(red) potato	213
lime	67
tomato	123
{green, red, yellow} bell pepper	119
carrot	61
avocado	201
mushroom	18
apple	110
bun	140
peach	150
egg white	29
egg yolk	13
orange	159
banana	118
leek	89
lemon	108
coconut	397
pineapple	905
red pepper	45
sausage	93
mozzarella	30
pumpkin	1500
broccoli	500
cucumber	301
cabbage	35
lettuce	260
green bean	6
salmon	159
asparagus	16
mango	207
tuna	180
cauliflower	216
artichoke	128
cherry tomato	16
artichoke heart	100
fish	57
pickle	37
pear	178
chicken thigh	140
apricot	35
eggplant	458
sun-dried tomato	3

Figure 7.2: Conversion table for the density of different ingredients (considering average medium size from [17]).