Food Swiping

DBM180 Report

Group 4
Ananya Mehrotra
Thomas Pilaet
Rick van Schie
Yunjia You
Yanyu Zheng

Table of Content

Introduction

Nowadays, the popularity of the Internet has shortened the distance between people, followed by a consequence that it is becoming easier to approach a diversity of information, including food. Beyond meeting the basic physiological need to feel satiety, the role of diet has been further explored and linked to other aspects of life. The study of O'Neil et al. (2014) highlighted that the dietary patterns and quality could not only affect the physical conditions, but also be related to mental health early in the life span. Moreover, adherence to one specific diet, namely Mediterranean diet, has been found to be positively correlated with subjective happiness, especially in adolescents (Ferrer-Cascales et al., 2019). Therefore, how to eat well both physically and mentally has become a topic of more and more interest.

To develop an ideal personal diet, it is worthwhile to apply artificial intelligence (AI), as this tool could give users control to the most extent, as well as maintain high performance in predicting user's preference with a continually evolved model. By collecting the right data and training the appropriate model, there is a possibility that the most essential attribute on which users make their decisions based could be found. Since everyone is likely to have a unique appetite and the scenario where the user has a meal could be various, the model would be personalized and usage of the product is expected to be more adaptive.

It could be the case that sometimes users do not even know how they choose what to eat, but this time Food Swiping could help. Inspired by the characteristics of advanced AI, the idea of designing a cuisine recommendation application comes up. To be more specific, this AI-based app Food Swiping is aimed at offering dynamically personalized recipes to those who want to explore different cuisines. Different from searching for a preferred recipe aimlessly, users are able to pick from recipes that the algorithm generates from their personal behavioural patterns. Also this app is designed to be Plug&Play, in which way users could enjoy cooking new recipes without spending more time in learning to use a new app.

In the following report, the design process composed of conceptual design, data collection, data documentation, and data mining would be further explained, followed by the demonstration and discussion of the result found through the iterative prototype. (In the end, the promising prospect and unique value of AI would be seen via this project. It indeed provides more solutions in implementing domain-specific technologies to the domain of design.)

Conceptual Design

This section details the inspiration, goal, user interface and user experience of the app (Food Swiping), thus going deeper in depth regarding the aesthetical prototype. Further, it details the incentive for further work.

Inspiration: Benchmarking (tinder/tender)

Tender is geared toward young people who want to eat out less and cook more. The app offers up food porn from all over the Internet and lets you swipe right to save a recipe, and swipe left to throw it away (Figure 1). It is more like Tinder for cooking (Dulenko, 2019) (Figure 2). But "Tinder for Food" is a way catchier headline. You can also filter results to your liking (options include drinks, dessert, chicken, vegan, seafood, pork, beef, and vegetarian, with more to come), and you can save recipes to your "Cookbook" (Huen, 2015; Meilus, 2015;).

Based on the user reviews Tender has some shortcomings which we as a team can use as a strength to improve it further and incorporate it within the newly designed app. Firstly, while using Tender most users found glitches with the app functionality. Specifically for the filtering aspect. Secondly, Tender does not seem to refine its suggestions based on the users likings and rejections (Judkis, 2015).Therefore, with the introduction of our app the team aimed to create a reliable system that is dynamic to the user's preferences.

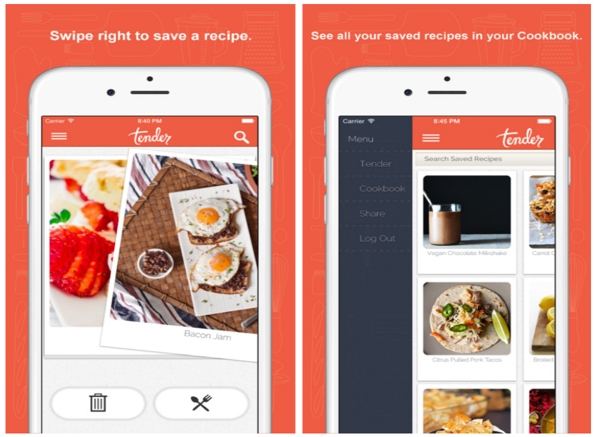

Swipe right to save a recipe.
See all your saved recipes in your Cookbook.
Tender
Menu
Tender
Cookbook
Share
Log Out
Search Saved Recipes
Bacon Jam

Figure 1: Design inspiration - Tender

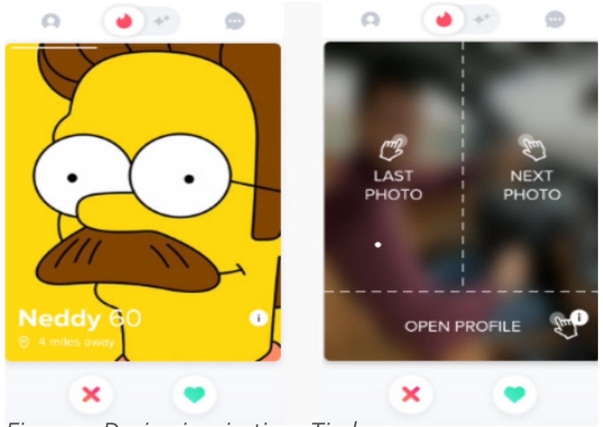


Figure 2: Design inspiration - Tinder

Goal of Food Swiping

Based on the inspirations, Food Swiping is geared towards all people who want to explore different cuisines whilst also keeping a healthy diet. The app offers recipes from all over the world and lets you save, discard or immediately start cooking a recipe. Thus, the main features of the app are seen in figure 3.

Behind the curtains a j48 algorithm provides personalized recipe suggestions based on the personal profile and the time when you use the app. From the user habits, it gains information about the user's specific preference, with respect to price, nutrition, dietary category, preparation time and the time of using; higher accuracy with feedback from users. Additionally, dishes that suit the prior preference more will appear with higher possibility.

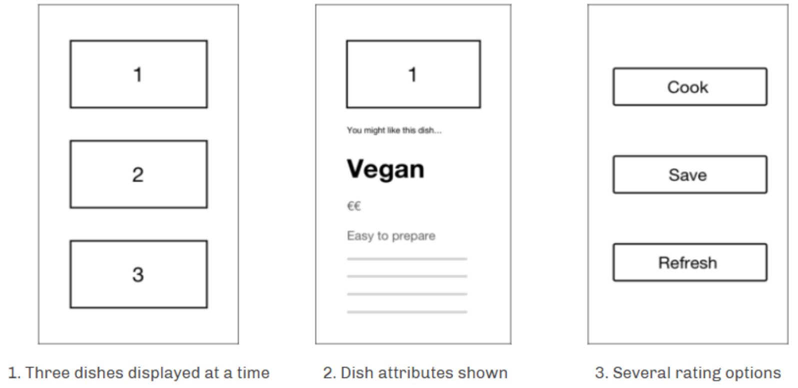

1
2
3
1
You might like this dish...
Vegan
€€
Easy to prepare
Cook
Save
Refresh
1. Three dishes displayed at a time
2. Dish attributes shown
3. Several rating options

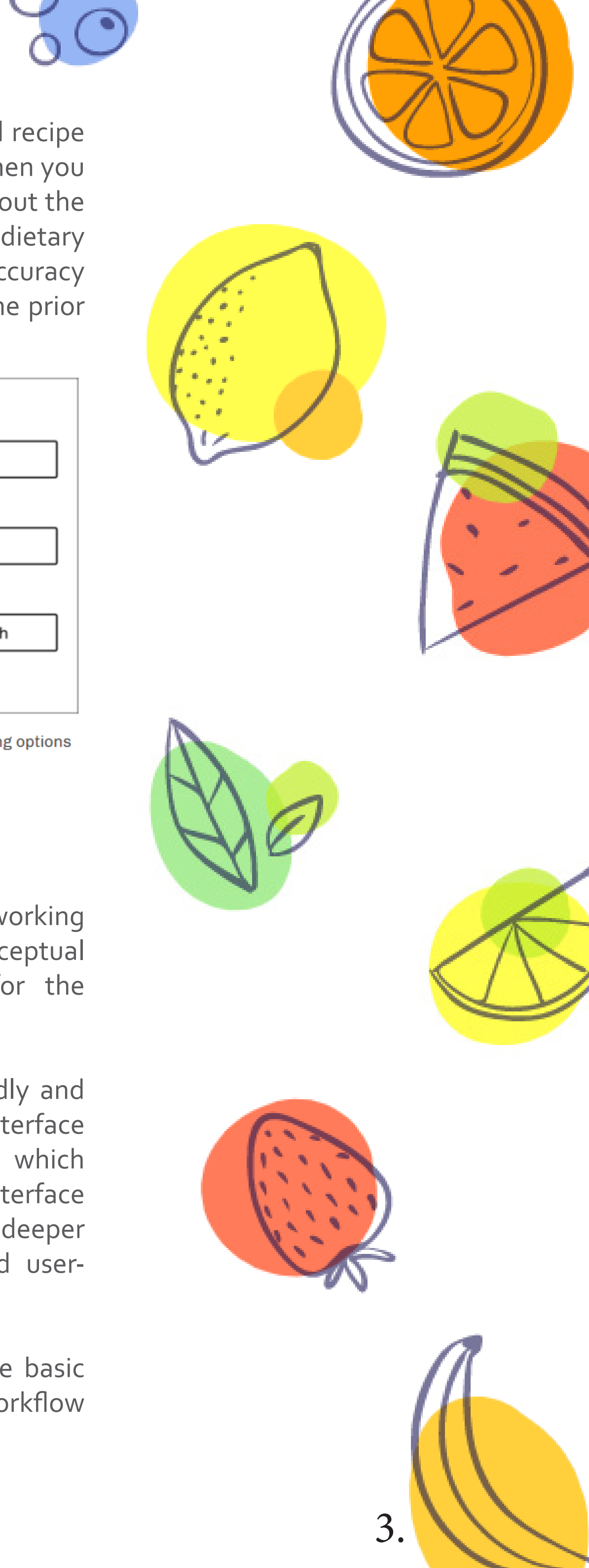
Figure 3: Features offered by Food Swiping

Adobe XD UI

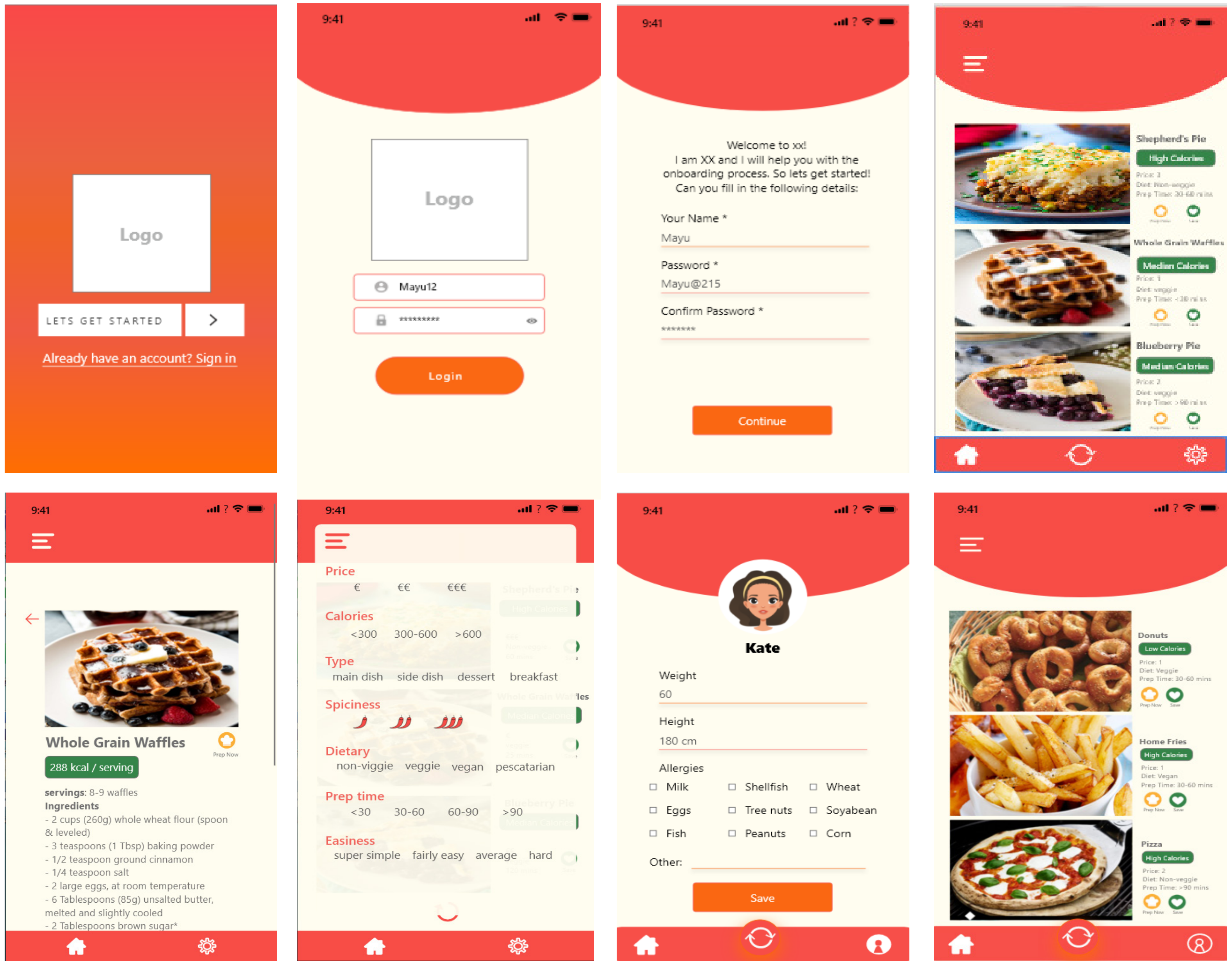
To have both the conceptual design as well as an entirely working app, the team decided to use Adobe XD for the conceptual interface (aesthetic prototype) whereas Processing for the working app (functional prototype).

Adobe XD facilitated the creation of a highly user friendly and intuitive interface. Thus, an interactive conceptual interface demonstrating the basic functionality and interactions which the app offers to the user was created. Initially the user interface of the existing food applications were referred to gain a deeper understanding on the aesthetics, layout, user-flow and user-friendliness while creating the conceptual design.

The initial version of the interface aimed at creating the basic layout of the app with a simple but a highly user friendly workflow incorporating all the functionalities as seen in figure 4.



This version represented the basic pen-paper idea and it included the desired information which is required to be communicated to the user through the app as seen in figure 4. Hence, the team aimed at gaining detailed feedback from the coaches and the fellow students whether sufficient information is shared with the users through the app or not. The user-flow of the conceptual design can be seen here: https://xd.adobe.com/view/fa4bb855-42d6-42ca-897f-744f4a930eb6-af18/



Welcome to xx!
I am XX and I will help you with the onboarding process. So lets get started!
Can you fill in the following details:

Your Name * Mayu
Password * Mayu@215
Confirm Password * *******

Whole Grain Waffles
288 kcal / serving
servings: 8-9 waffles
Ingredients
- 2 cups (260g) whole wheat flour (spoon & leveled)
- 3 teaspoons (1 Tbsp) baking powder
- 1/2 teaspoon ground cinnamon
- 1/4 teaspoon salt
- 2 large eggs, at room temperature
- 6 Tablespoons (85g) unsalted butter, melted and slightly cooled
- 2 Tablespoons brown sugar*

Weight 60
Height 180 cm

Figure 4: Initial version of the Concepual Interface

Based on the feedback from the coaches, the existing interface design was further improvised and converted into a more professional UI. Therefore, the new version of the conceptual interface highly focussed on enhancing the styling, aesthetics and user-friendliness of the app. Additionally, the user flow and the user experience was further enhanced during this version by integrating information in a highly visual manner of the app as seen in figure 5. The user-flow of the conceptual design can be seen here: https://xd.adobe.com/view/b3ab8e4f-5c8f-4e18-a5c6-3eefb45c153f-e6f8/

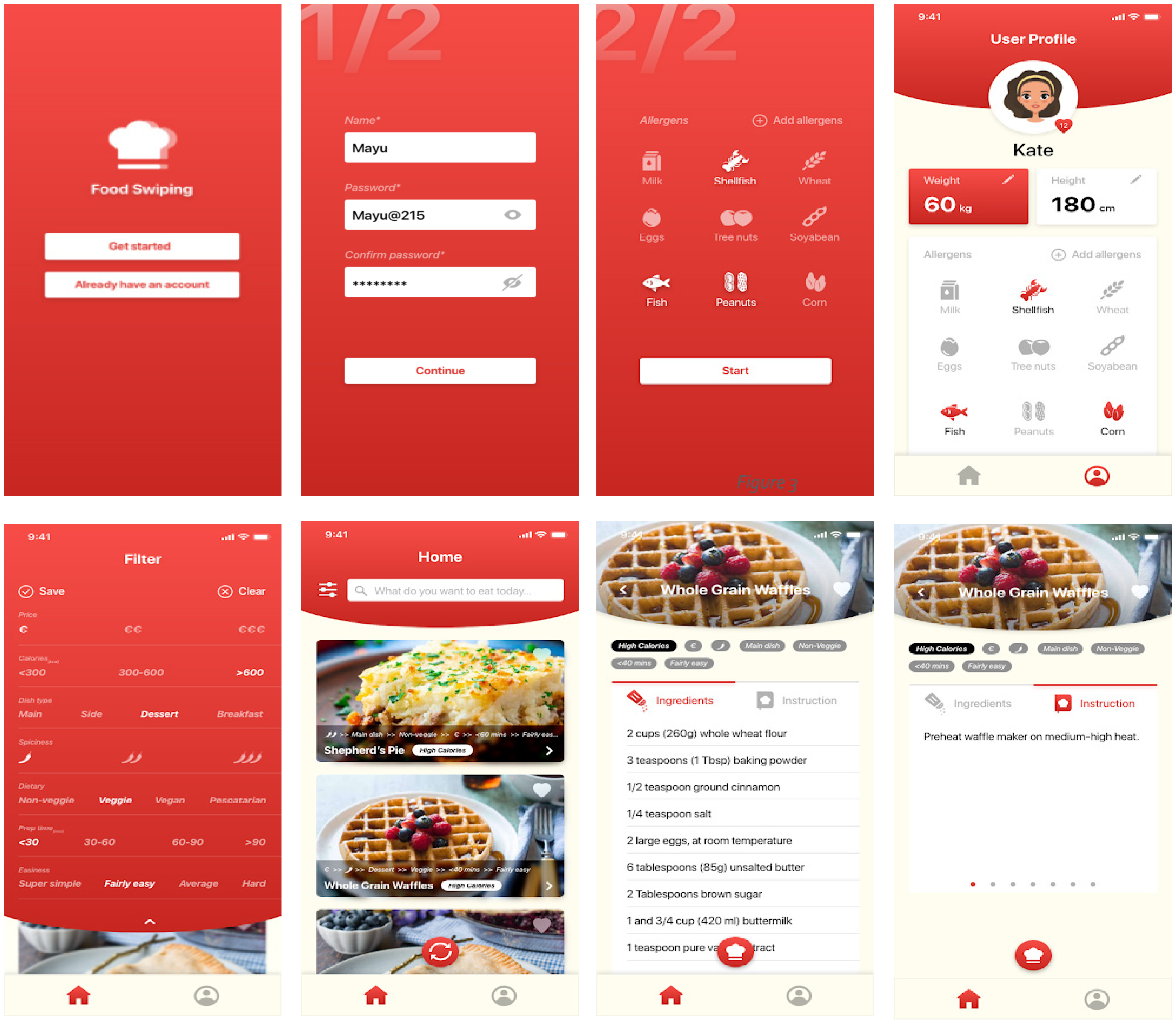


Figure 5: Final version of the Concepual Interface

User Experience

Goal of the Interface

The app aims to facilitate simple and basic user interactions in order to facilitate a basic user flow. Thus, with the newly improvised version of the conceptual design we aim to create a neat and clear user interface. The app is designed to be Plug&Play (or perhaps Click&Eat), therefore, no pre-training the model and no extensive account is required. Additionally, instead of comparing the recipes the app allows the users to choose their favourite recipes from the accepatables ones.

User-flow

Two different user scenarios were created in order to create a user friendly workflow for both first time users as well as for a standard user.

First time use

This user scenario elaborates on the step by step workflow which facilitates the first time downloading and installing procedure of a new user as seen in figure 6. The user initially needs to download Food Swippling from the app store, general dataset and the general model. After that the user opens the app and creates a personalised profile by adding in their name, gender, height, weight and allergies (if any). Further, the system extends the general dataset with personalized data, thus, asking the user to interact with the system inorder to train it. This helps in retraining the model to create a personalized model, therefore, this model is then applied in processing to create a personalized behaviour.

Standard use

This user scenario elaborates on the step by step workflow of a regular user when the user already has an account/profile created in the app as seen in figure 7. Each time the user opens the app, they have the ability to update their profile if neededby altering their name, gender, height, weight and allergies. Each time the user interacts with the app by going through piles of three recipes helps/contributes in training and adding new data in the app. Additionally, the user

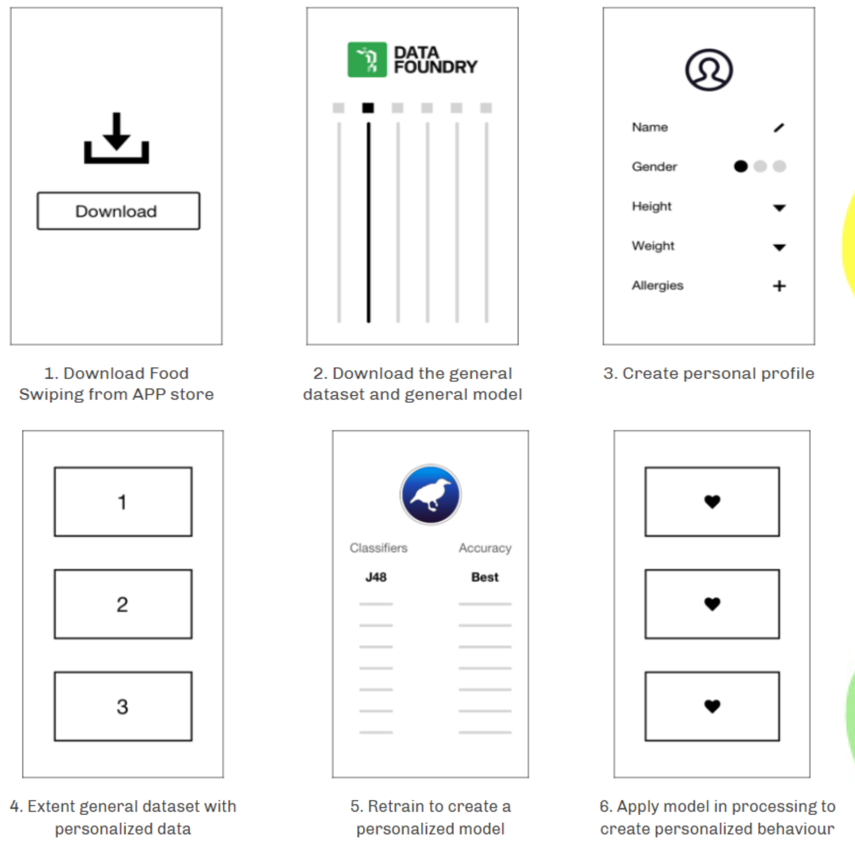

Download
DATA FOUNDRY
Name
Gender
Height
Weight
Allergies
1. Download Food Swiping from APP store
2. Download the general dataset and general model
3. Create personal profile
1
2
3
Classifiers
Accuracy
J48
Best
4. Extent general dataset with personalized data
5. Retrain to create a personalized model
6. Apply model in processing to create personalized behaviour

Figure 6: Final time use scenaio

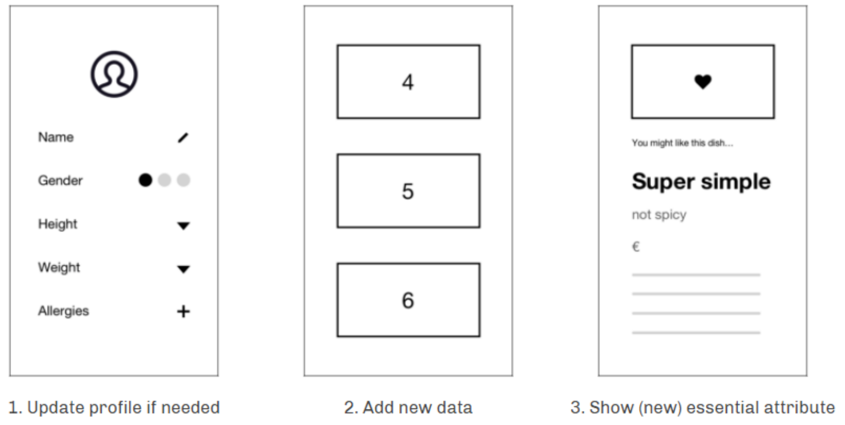

Name
Gender
Height
Weight
Allergies
4
5
6
You might like this dish...
Super simple
not spicy
€
1. Update profile if needed
2. Add new data
3. Show (new) essential attribute

Figure 7: Standard use scenaio

has the opportunity to alter and personalize their recipe options by using the filters anytime they wish to do so. The filters allow the user to narrow down the recipe options based on the price, calorie, easiness, dish type, dietary category, preparation time and spiciness. Thus, the system updates itself based on the new inputs and user behaviour, hence, would show the new essential attributes to the users.

Data gathering interface: Processing Interface Design

Later in the report, the Demonstrator section elaborates on the aesthetically basic, but otherwise completely functional Processing design. It includes all the features and functionalities that were envisioned for the app. Multiple Processing libraries were used, namely ControlP5, OOCSI and Weka. The Processing Interface design aims to collect and store data from different users for data mining purposes. The collection, documentation and mining process of this data will be described in the following sections.

Data Collection

Collected data

Two datasets were created to support the design, namely IoT dataset and entity dataset. The IoT dataset contains dynamic data related to dish characteristics (e.g. price, spiciness, etc.) and user preferences (e.g. choice) while the entity dataset contains static data related to the users (e.g. BMI, gender, etc.). The collected data is a mix of numerical and categorical data, saved in Data Foundry. How the data is sent and saved on Data Foundry will be elaborated in the Demonstrator section, and how it is used for training and predicting will be elaborated in the Data mining section.

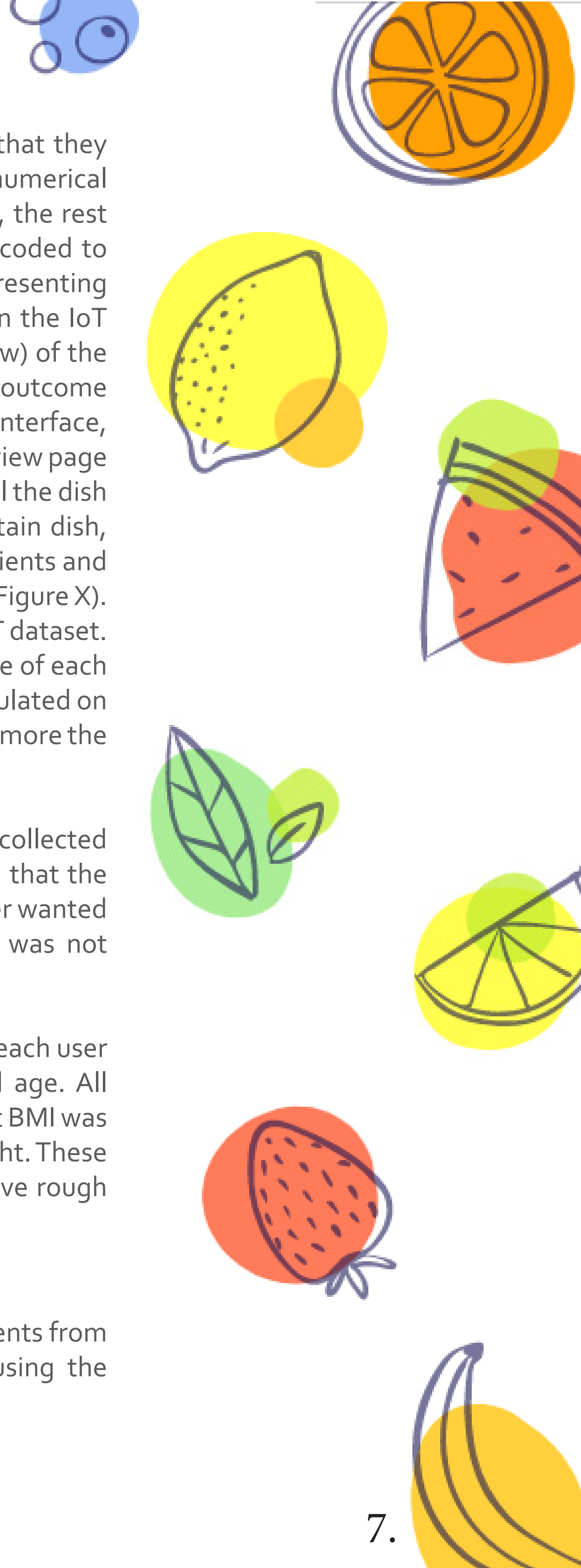
The dish characteristic data in the IoT dataset includes dish name, dish category (vegan, veggie, pescatarian, non-veggie), type of dish (main dish, side dish, dessert, breakfast), ease of preparation (super simple, fairly easy, average, hard), preparation time, nutrition value, price and spiciness. These data were included because of the assumption that users would make their decisions based on certain characteristics that they are interested in. Four out of these eight data are numerical (i.e. preparation time, nutrition value, price, spiciness), the rest four are categorical data. The price and spiciness are coded to numerical data ranging from 1 to 3, with higher score representing higher price or more spicy. The user preference data in the IoT dataset refers to the choice (i.e. refresh, save, cook now) of the users regarding each dish. This is the target prediction outcome in data mining. When interacting with the processing interface, the users click the "refresh" or "save" button on the list view page to indicate that they do not like the dish or that they feel the dish acceptable and want to cook it later. If they like a certain dish, they can go to the detailed view page to see the ingredients and procedures, and decide whether to cook it now or not (Figure X). These clicking actions are archived as "choice" in the IoT dataset. In fact, there was a numerical attribute storing the score of each dish, which was discarded in the final model. It was calculated on the basis of the user's choices. The higher the score, the more the user prefers that dish.

Contextual data such as day of the week and time were collected in the earlier stage. They were included for the reason that the time of using might influence which type of dish the user wanted to eat. However, they were discarded later as there was not much contribution.

As for the entity dataset, it stored the personal data of each user including height, weight, BMI, continent, gender and age. All these data were collected on the login page, except that BMI was automatically calculated from the user height and weight. These static data were used to train a general model that gave rough results for the first-time users.

Iterative collecting process

All the five students from the team and the fellow students from the same course contributed in data collection by using the processing interface.



In the first round of data collection, each team member spent nearly 10 minutes every day interacting with the processing interface, lasting for around one and a half weeks, so that there was enough data (at least 400 instances per person) for the exploration into different data mining methods. In this stage, the prior aim was to expand the dataset as much as possible, so some choices were quite random (excluded from the training set later). The choices, scores, as well as contextual data were collected via the interaction.

Next, in order to find out the most suitable data mining method, the data was collected in a more serious manner, with underlying patterns (around 700 instances). The choices were based on price or price and nutrition value. The score and contextual data were discarded to simplify the dataset and the predicting method, or to say, only recipe-related data (dish characteristics) and choices were collected and stored.

After having the model with the best performance (J48), more data were collected to confirm the decision and fine tune the parameters of the model. Two members interacted with the interface everyday for a week to collect more data, now with personal data included which were height, weight, BMI and gender. The choices were mainly based on dish category or nutrition value and preparation time. In this period they gathered around 800 instances in total. Later, continents of origin and age were added to the personal data. Hereby hopefully a general model can be trained to predict from personal data for first-time users.

At last, to make the general model work, 5 people from different backgrounds (our team members and fellow students) were invited to help with the data collection. Each of them were asked to spend around 10 minutes (around 100 instances per person). Another fellow student was invited to experience the interface, whose data were then used as the test set (153 instances).

Data Documentation

As mentioned in the Data Collection section, an integrated IoT dataset was created for the system. This was done by mergir an already existing dataset along with some added attribut required for developing an intelligent system.

Findability

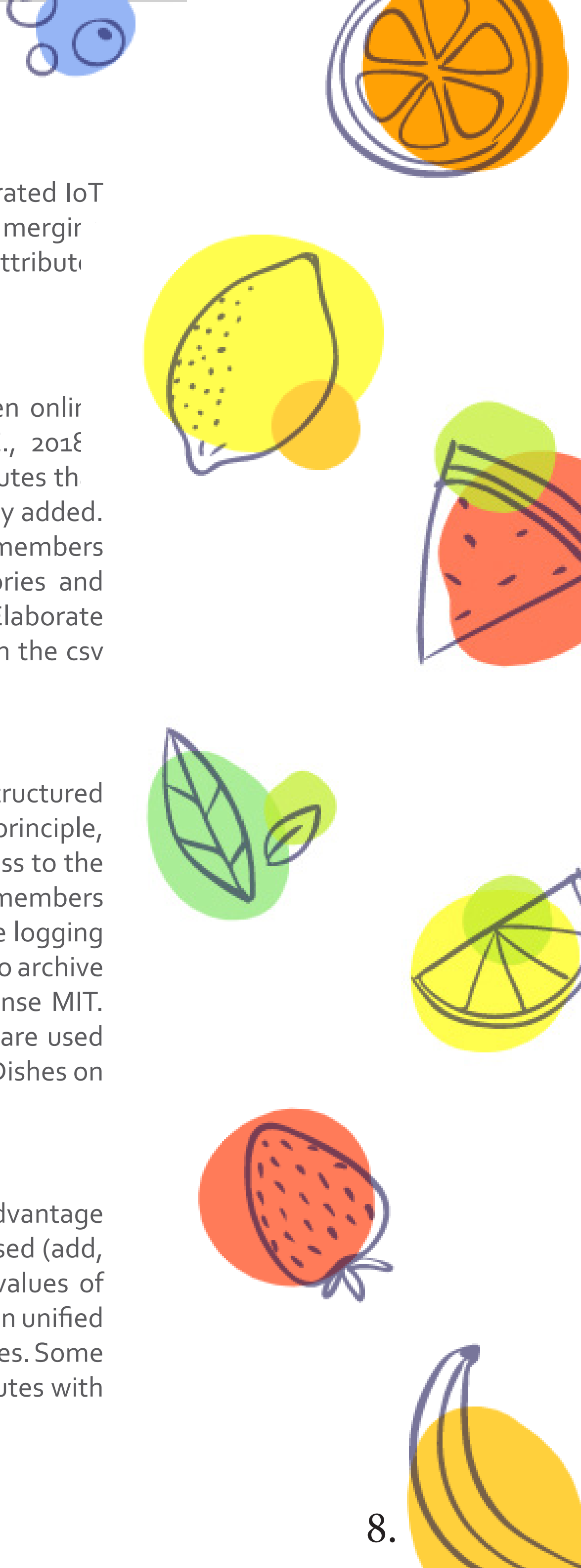
Most of the recipe attributes are gained from an open onlir recipe database of Airtable (Sayers, L., & Sayers, C., 201 Besides, missing information and other relevant attributes th the initial data repository does not contain are manually added. According to the required ingredients, the team members classified all the dishes into different dietary categories and subjectively evaluated the spiciness range of each dish. Elaborate information about the recipe dataset could be found in the csv file named RecipeDatabase.

Accessibility

To store, process and export data in an easy and structured way, the Data Foundry platform is introduced. In principle, Data Foundry only allows private or team internal access to the dataset. For this project, all data were collected from members of this team and some other teams in this course. Once logging in the data gathering interface, it would be automated to archive raw data in the Data Foundry platform under the license MIT. It is also possible to get open access to the data that are used for training via another csv file named Data_ALL_withDishes on Canvas.

Interoperability

Using the csv format to archive raw data has the advantage that data could be easily added, deleted, and reprocessed (add, subtract, multiply, and divide, etc.). All the numeric values of cells belonging to the same attributes are documented in unified units, thus allowing comparison between different recipes. Some continuous attributes are even transformed into attributes with number or symbol coding scheme.



For example, the attribute Price has been divided into three intervals using the euro symbol, from cheap to expensive. In this way, it would be easier to obtain an intuitive grasp of the meaning behind the data.

Reusability

The initial dataset contains both recipe attributes and user attributes to better describe what we are likely to use for training the model. Most attributes are built in the existing recipe dataset, while other additional attributes are well defined in the Wikipedia, meeting domain-relevant community standards.

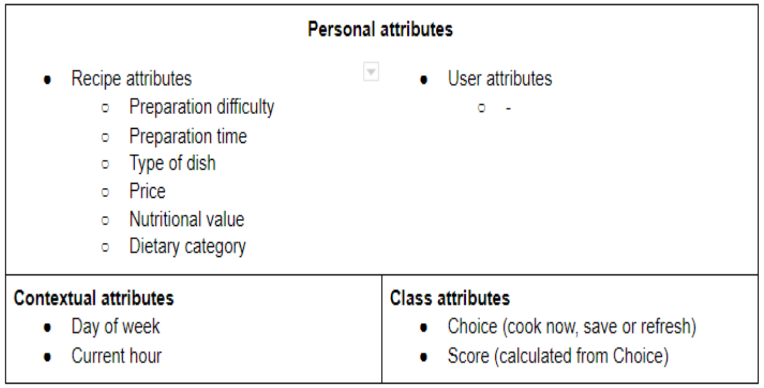
Data Mining

Attribute selection

In this subsection several iterations in our data mining process are described in terms of attribute selection. These were not hard separable iterations as described below, but instead overlapped in some cases. However, to improve readability they have been categorized in three different iterations.

First iteration

The initial idea was that through data collection of both personal and contextual attributes a model could be created fitted to each individual user. The personal attributes at that time only consisted of recipe attributes combined with an action (cook, save or refresh). As mentioned in the Data documentation section the recipe attributes in part came from Airtable (Sayers, L., & Sayers, C., 2018). However, more recipe attributes were also added manually. In the beginning no user attributes were collected. After data was collected from a user's interaction with the data gathering interface, two class attributes were saved. One being the choice of the user for each dish (cook, save or refresh) and the other being the score for each dish over time. The idea was that we could see what the score of each dish was over multiple interactions of the same user. An overview of all attributes used for data mining at this point can be seen in Table ??.

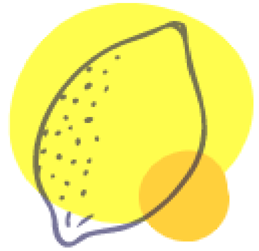


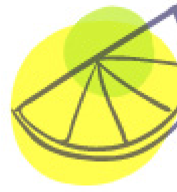
Personal attributes	
• Recipe attributes ◦ Preparation difficulty ◦ Preparation time ◦ Type of dish ◦ Price ◦ Nutritional value ◦ Dietary category	• User attributes ◦ -
Contextual attributes • Day of week • Current hour	**Class attributes** • Choice (cook now, save or refresh) • Score (calculated from Choice)

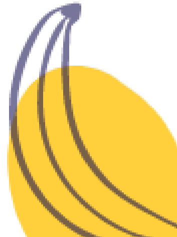
Second iteration

At first, the "score system" was adopted to represent the user's preference for each dish over time. Therefore, the algorithm's task was to predict the score of each dish. The idea was that all recipes would be placed in a 'stack of cards'. The higher the score, the higher the recipe would be in that stack. In terms of the interface a recipe would be shown more quickly if it had a high position in the stack. After a first round of data collection an attempt was made to explore what classifier(s) could best be used to find patterns in the data. The classifiers were trained with all the shown attributes in Table ??. Through ZeroR the baseline was set at an accuracy of around sixty percent. However, none of the classifiers that were tried, gave satisfying accuracy ratings. Weka classifiers such as OneR, NaïveBayes, Logistic Regression and J48 all gave lower or only slightly (a few percentage points) higher accuracies. For this reason it was not possible to conclude which classifier(s) could best be used for finding patterns.

In any case we saw that the contextual attributes did not give any insightful information. The reason for this was that data from a single person was collected in large amounts during only sporadic and irregular intervals. Initially it was planned that a user would use the data gathering interface as they would do in reality. For example, use it during each morning to select what to eat for breakfast. However, due to time constraints of all group members (the users) this was not possible. Instead, an individual user would use the data gathering interface a few







times per week and collect much data at once. With that the three choices connected to a recipe became in essence a rating system with 'Cook now' being the highest rating. Concluding, the data on the contextual attributes - for example the current hour and day of the week - did not give insightful patterns. For this reason it was decided to leave out the contextual data altogether for future data mining processes.

Moreover, it was found that the scoring system did not give insightful results either. Different scores for each individual dish were saved over time, but (maybe logically) did not lead to any patterns. Besides, the scoring system was also hard to implement adequately in the data gathering interface in Processing. For these two reasons it was decided to drop the scoring system and only use the individual choices each user gave to a recipe. With that the class attribute was only connected to an individual instance (recipe) instead of over multiple instances (recipes) over time.

Third iteration

After some rounds of data collection and mining it was also found that the recipe attributes did not indicate much about what the taste of the dish was. Several ideas emerged of how to indicate this. One idea was to add an attribute through which the most important ingredients were indicated. Implementing this would lead to a good indication of what the dish could taste like. However, it would also be very time-consuming and hard to implement, because a framework would need to be made with matching major ingredients for over 140 recipes. Another idea that emerged was to indicate which cuisine the dish belonged to. However, when this was looked into it was found that many possible cuisines existed and that some cuisines only had one recipe in our database that belonged to them. Instead of using the major ingredients or cuisine of a dish we chose to add a Spiciness attribute. This attribute was separated in three categories (Not Spicy, Mild and Hot). Adding this attribute was easy to do.

Next to that the taste indication was missing, we also noticed that we lacked user attributes in our data gathering. Actually none at all were gathered in the beginning. It was chosen to add such attributes as we imagined that for example the BMI of a us could preemptively indicate their preference for the nutrition value of a recipe. For this reason first the following user attribut were added: weight, height, gender and age. These attribut were used separately in the data mining process and weight ar height were also used to calculate the user's BMI. Which too wa added as an attribute in the data mining process. This seeming could lead to an overrepresentation of weight and height, bu this is not the case. Firstly, because the classifiers that we use do not use all attributes in their decision making necessarily (such as J48). Secondly, before a model was trained some user attributes were also filtered out, which will be explained in the 'Model analysis' section.

Later the attribute "Continent of origin" was also added. We imagined that this perhaps preemptively could indicate some taste preferences of the user. This was the final attribute that was added. With that the final overview of the used attributes can be seen in Table ??.

Model analysis

To find an indication what classifier could best be used for the data mining process personas were created. To do this two personas were created. One persona would only rate the recipes based on their price and the other persona only on their price and nutritional value. For the former, 266 instances were collected and for the latter 422 instances. The classifier accuracies for each persona can be found in figure 8. Ten fold cross-validation was performed for each classifier. The table shows that from this 'persona test' J48 was the most accurate classifier. However, many limitations are connected with finding a useful classifier through personas. Such personas are not valid 'end-users'. The next step was to find what classifier would be most accurate when trained on the data of actual users. Three different users were selected from the then existing database we had. A separate csv

Persona 1 (only price)		Persona 2 (only price and nutritional value)	
ZeroR	48,12%	ZeroR	42,65%
J48	100,00%	J48	99,76%
Logistic	100,00%	IBk	96,92%
OneR	100,00%	Logistic	95,02%
SMO	100,00%	OneR	91,00%
NaiveBayes	99,62%	NaiveBayes	91,00%
IBk	97,37%	SMO	88,15%

Figure 8: Table representing classifier accuracies for each persona

file was created for each which was then cleaned for training. This user data was trained with the same classifiers that were used for the personas. The user attributes were left out for this training as they had no added value, because they were the same for the individual users. The classifier accuracies for each of the three users can be found in figure 9. For two users all used classifiers were more accurate than the baseline with J48 again the most accurate. However, for 'User 3' no pattern could be found. Therefore none of the classifiers outperformed the baseline. The most likely cause for this was that this user filled in their recipe ratings randomly compared to the other two users who had rated the recipes in a more valid manner.

It now became clearer that J48 most likely was the classifier to go with to use in our model. For this reason a preliminary search was done on finding what J48 parameters to use in Weka. To do this, each parameter was changed separately to look at its effect on the accuracy of the model. From this it was found that none of these changes increased the accuracy. With the exception of when the subTreeRaising parameter was turned to false. This led to a very slight accuracy increase of around 0,2 percent point. The minNumObj parameter was also raised to higher values. This thus did not lead to higher accuracies, but neither led to much lower accuracy percentages immediately. For example the accuracy of the J48 model was still similar for values such as 2 and 50. For example for 'User 1' this led to accuracy percentages of 84,37% and 80,97% respectively. However, when the minNumObj parameter would be raised even higher it did lead to lower accuracies as the decision tree would be downscaled too much. In any case this showed that making the J48 model less complex would still lead to similar accuracy percentages and could therefore be advantageous. A advantage is for example that the model can better be upscale to a more complex one in the future.

User 1		User 2		User 3	
ZeroR	51,70%	ZeroR	61,59%	ZeroR	60,20%
J48	84,37%	J48	86,02%	NaiveBayes	62,50%
IBk	84,17%	IBk	83,01%	Logistic	60,53%
SMO	82,77%	Logistic	81,77%	SMO	60,53%
Logistic	82,16%	OneR	80,18%	J48	58,22%
NaiveBayes	75,95%	SMO	76,81%	OneR	55,92%
OneR	73,95%	NaiveBayes	75,22%	IBk	46,38%

Figure 9: Table representing classifier accuracies for each of the three users

The next step in the data mining process was to attempt to create a model that could encompass all user data at once. For this the same classifiers were tested again. Besides the 10-fold cross validation a test was also conducted with a test set. This test set was from a person from the course whose data has not been used in the training process. 153 instances were collected for this test set. However, the ratio between the choices of this person greatly differed from the ratios of the training set. The 'refresh' rate in the test set was over 76 percent of the total 153 instances. While in the training set the 'refresh' rate was only 61 percent. Therefore, a 'balanced' test set was made in which the refresh rate was similar to that of the training set. This was done to get a more balanced comparison between the training and test. The balanced test set was created by randomly deleting instances in which the user had selected refresh. From this, a test set was created of 94 instances and a refresh rate of just under 62 percent.

The accuracy of the classifiers was tested with all attributes and also with only the recipe attributes. The latter was done as well, because after some data mining it was noticed that

this increased the accuracy of the model when a test was performed with a test set. This indicates that by using the user data that we currently have the model is overfitted to those users. In figure 10 and in figure 11, the classifier accuracies are shown for both test sets.

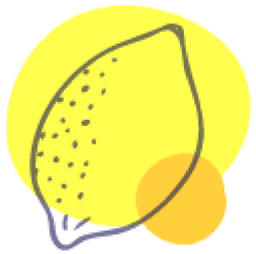
All users *(using all attributes)*			
Classifier	**10-fold CV**	**With test set**	**Balanced test set**
ZeroR	61,47%	76,47%	61,70%
J48	75,76%	50,33%	40,43%
IBk	74,36%	48,37%	39,36%
SMO	67,39%	76,47%	61,70%
Logistic	64,80%	52,29%	45,74%
OneR	62,91%	74,51%	62,77%
NaiveBayes	57,41%	74,51%	59,57%

Figure 10: Table representing classifier accuracies using all attributes

All users *(no user attributes)*			
Classifier	**10-fold CV**	**With test set**	**Balanced test set**
ZeroR	61,47%	76,47%	61,70%
J48	62,31%	77,78%	67,02%
IBk	62,51%	73,20%	63,83%
SMO	61,47%	76,47%	61,70%
Logistic	61,12%	75,16%	61,70%
OneR	62,91%	74,51%	62,77%
NaiveBayes	59,86%	73,20%	60,64%

Figure 11: Table representing classifier accuracies with no user attributes

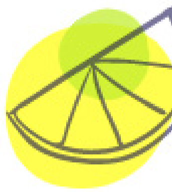
Looking at the accuracies of all classifiers for the model of all users we see that J48 still outperforms all other classifiers. With the exception of when the decision tree is overfitted through the usage of the user attributes. For this reason we stuck with using J48 as the classifier to use for creating our model. The next step was to see which user attributes could lead to more insights for creating the general model with at the same time not overfitting. The results from this selection process can be seen in figure 12. As shown in this table the same pattern is visible from the previous accuracy test. When using the user attributes a higher classifier accuracy is found with cross validation, but (much) lower accuracies are found when testing with the test set. Again this shows that by using the current user attributes the model might be overfitted to that data.



Besides attribute selection, the J48 parameters were also looked at. As with previous attempts this again showed little improvement in the classifier accuracies. Raising the minNumObj parameter led to the greatest improvement in accuracy. This too was only limited to a few percentage points, which is also an indication that the model can be created simpler without leading to lower accuracies. Even raising that parameter to 200 still led to similar accuracies.

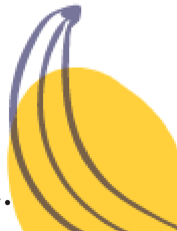
Attributes used	**J48 accuracy (10-fold CV)** *ZeroR: 61,47%*	**J48 accuracy (with test set)** *ZeroR: 76,47%*	**J48 accuracy (Balanced test set)** *ZeroR: 61,70%*
Recipe attributes: All User attributes:None	J48: 62,31%	J48: 77,78%	J48: 67,02%
Recipe attributes: All User attributes:Only BMI	J48: 75,46%	J48: 71,90%	J48: 56,38%
Recipe attributes: All User attributes:Only BMI and Gender	J48: 75,62%	J48: 70,59%	J48: 55,32%
Recipe attributes: All User attributes:Only Gender	J48: 68,34%	J48: 69,93%	J48: 58,51%
Recipe attributes: All User attributes:Only Continent	J48: 66,90%	J48: 52,29%	J48: 42,55%
Recipe attributes: All User attributes:All	J48: 75,76%	J48: 50,33%	J48: 40,43%

Figure 12: Table representing the accuracy test



Data mining conclusions

From the above attribute selection and model analysis several conclusions can be taken from the creation of our general model. The main conclusion is that currently with our dataset and including attributes it is difficult a general model that performs well. The general J48 models that we have tested do not outperform the baseline by much or in some cases not all. Moreover we see that by using the current user data and the including attributes the model is overfitted to that user data. This is most likely caused by our limited user variation. For the creation of a more accurate model, data is needed from a more



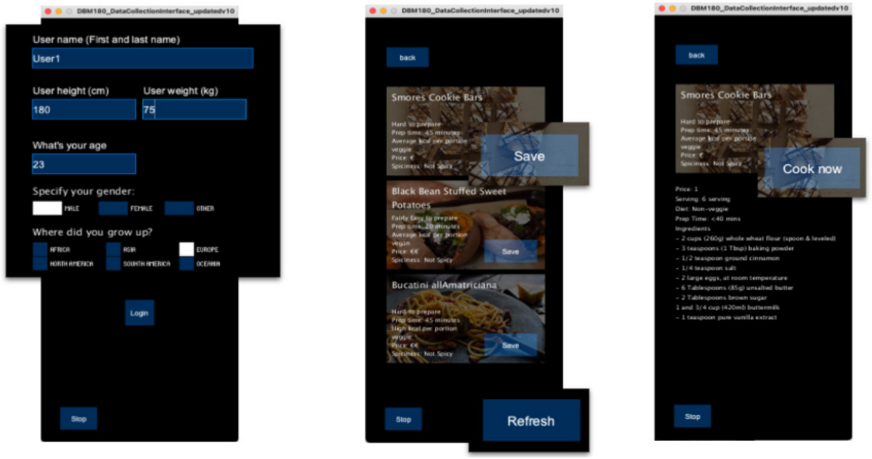
diverse user base. Currently, the decision tree overfits the data to the limited user base. Even when the minNumObj parameter is raised or when most user attributes are removed. Examples of our limited user base are that we only had one male from Asia, only one person above 180 centimeters and all people were of ages 22 or 23. However, what needs to be said is that after deployment our general model adapts itself to the individual user. Currently, the model is trained with the default dataset and has not adapted itself to its user (or in our model analysis the user from the test set). This means that the starting accuracy may be low, but it will improve over time.

Final remarks on using J48 as classifier

Throughout the data mining process the J48 classifier was most accurate in most cases. For that reason we could keep using J48 as the classifier for the creation of our model. However, another important reason needs to be mentioned. In our application we also wanted to show what recipe attribute was the most important for the user when rating the recipes. To do this it needed to be known what the user looks at first when rating the recipes. This was most best, if not only, possible in Processing by using the J48 classifier. In processing we could find the first node of the decision tree quite easily, but it was much more difficult to find the most important recipe attribute for other classifiers. Moreover, J48 is also an unstable learning scheme and could adapt itself therefore more easily through an individual user's preferences. Where other classifiers use all recipe attributes from the get go, J48 only uses the recipe attributes that are important to the user's preferences. With that through usage of the application a user might end up with a completely different decision tree compared to the decision tree of the general model that we would have created.

Demonstrator

The final demonstrator of this project is actually a combination of an aesthetical and functional prototype. The aesthetical prototype is already described in the concept section and includes the aimed user experience and styling. This section will go into more depth about the functional prototype (see figure 13), which uses the aesthetical prototype as a guideline for the layout. A demo of the functional prototype can be found here: [video link to demo interface].


DBM180_DataCollectionInterface_updatedv10

User name (First and last name): User1
User height (cm): 180
User weight (kg): 75
What's your age: 23
Specify your gender: MALE [✓] FEMALE OTHER
Where did you grow up? AFRICA ASIA EUROPE [✓] NORTH AMERICA SOUTH AMERICA OCEANIA

Smores Cookie Bars
Hard to prepare
Prep time: 45 minutes
Average kcal per portion
veggie
Price: €
Spiciness: Not Spicy

Black Bean Stuffed Sweet Potatoes

Bucatini allAmatriciana

Price: 1
Serving: 6 serving
Diet: Non-veggie
Prep Time: <40 mins
Ingredients
- 2 cups (260g) whole wheat flour (spoon & leveled)
- 3 teaspoons (1 Tbsp) baking powder
- 1/2 teaspoon ground cinnamon
- 1/4 teaspoon salt
- 2 large eggs, at room temperature
- 6 Tablespoons (85g) unsalted butter
- 2 Tablespoons brown sugar
1 and 3/4 cup (420ml) buttermilk
- 1 teaspoon pure vanilla extract

Figure 13: Functional prototype created using Processing

Limited user experience

Note that the functional interface does not meet the experience qualities that the overall concept is aiming for, e.g. clicking the "cook" button on the detailed screen will bring you back to list view instantly and refresh the dish. This would not be desired in a real situation, as you may still want to access the recipe details. The reason for these shortcomings is twofold.

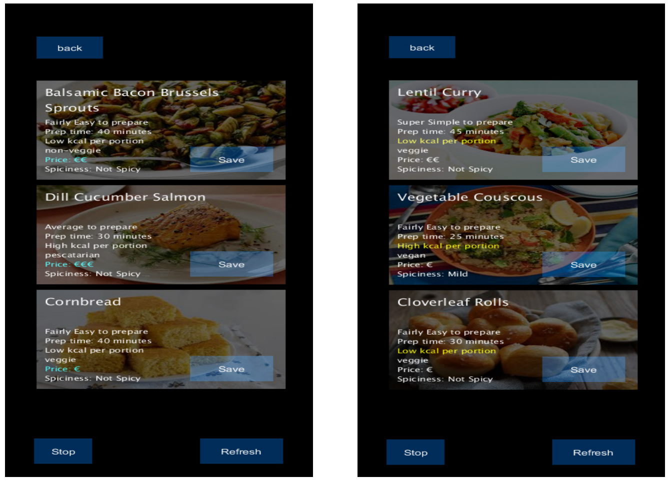
Firstly, simplicity. The purpose of the functional interface is the implementation of the algorithm to make a working prototype first, and afterwards make it aesthetically pleasing.

Secondly, reliability of the training data. If the dish would not be reassigned, it could be clicked more than once in a row, creating duplicate instances in the dataset, or, if the user goes back to the list view and clicks refresh, an additional instance would be added with the "refresh" label, which contrasts with the desired "cook" label. Ultimately, this would make the training data very unreliable. Another approach to prevent this issue from arising would be to remove the cook button once it is clicked, and ignore that dish when the page is refreshed. Though, this would be more difficult to implement (see point 1).

Code foundation

The functional prototype is an interface made in Processing. As a starting point, the provided example code by Chiang (2020) was used and expanded over the duration of the project. This code already included some main interface elements from the ControlP5 library and a connection to DataFoundry by means of the OOCSI library.

Additionally, the J48 example code by Hu (2020) was used to implement the J48 algorithm of the Weka library into the interface. This code was modified to fit the structure of the dataset and extract additional information like the most essential attribute by accessing the top node of the decision tree. This attribute is used for personalization of the interface (currently limited to highlighting the attribute with a unique color, see figure 14) and transparency regarding the algorithm's decision making.



Balsamic Bacon Brussels Sprouts
Fairly Easy to prepare
Prep time: 40 minutes
Low kcal per portion
non-veggie
Price: €€
Spiciness: Not Spicy

Dill Cucumber Salmon
Average to prepare
Prep time: 30 minutes
High kcal per portion
pescatarian
Price: €€€
Spiciness: Not Spicy

Cornbread
Fairly Easy to prepare
Prep time: 40 minutes
Low kcal per portion
veggie
Price: €
Spiciness: Not Spicy

Lentil Curry
Super Simple to prepare
Prep time: 45 minutes
Low kcal per portion
veggie
Price: €€
Spiciness: Not Spicy

Vegetable Couscous
Fairly Easy to prepare
Prep time: 25 minutes
High kcal per portion
vegan
Price: €
Spiciness: Mild

Cloverleaf Rolls
Fairly Easy to prepare
Prep time: 30 minutes
Low kcal per portion
veggie
Price: €
Spiciness: Not Spicy

Figure 14: highlighted personalized most essential attribute: price and calories

Loading dishes

During the startup of the program, a connection to an Iot dataset and Entity dataset on DataFoundry is made using the OOCS library. The J48 model is loaded to make predictions about dis choices. The recipe dataset is loaded including the attributes c all individual dishes. And the interface control elements fror the ControlP5 library are loaded and the elements for the firs window are shown. The shown interface elements are based o a finite state machine that displays the elements based on one c three states: login, list view and detailed view.

The user logs in using very basic personal information including name, weight, height, age, gender and continent of growing up. There are two reasons for asking for rather basic personal information. Firstly, it makes starting to use the app less burdensome. Secondly, it allows us to build a more complete dataset for training a model. Asking for a choice between every country would make finding patterns much more difficult with limited training data.

As soon as the user goes to the list view (after logging in), the list of recipes is accessed using the "assignRandomDishes()" function from which three random dishes and their attributes are loaded into the 2D String array "randomDishes[][]". The first index represents one of the three dishes that are displayed on the interface, and the second index holds all of the attributes of those three individual dishes.

Predicting preferences

The algorithm is used to predict whether the selected random dish will actually be cooked, which is the desired result of the app. If the selected random dish results in the prediction "cook", it will be kept, otherwise another random dish is selected.

To give false-negative predictions (i.e. not predicted to be cooked, while the user would actually cook the dish) a chance to present themselves to the user, the number of assignment iterations is limited. The higher the allowed number of iterations, the higher the chance of getting dishes with a "cook" prediction. The downside of a higher number of iterations is the loading time, although ten iterations per dish (i.e. max 30 in total) still feel quite snappy from a user perspective. Thirdly, the limit on the iterations prevents an infinite loop through dishes, may it ever happen that the algorithm cannot find any dishes with the prediction "cook". Note that keeping the iteration value at one results in no effect of the algorithm, since the prediction outcome cannot induce another iteration. This setting is used to collect the initial training data. Also, iterating over the length of the recipe dataset, while skipping duplicate random values, will ensure three "cook" predictions, if that is possible.

Sending data

Once a button has been clicked that is linked to submitting data (i.e. "cook", "save" or "refresh"), the whole set of attributes is sent to the IoT dataset and the personal profile is updated in the Entity dataset. The cooked or saved dish will be reassigned according to the aforementioned steps. In case refresh was clicked, all three dishes will be reassigned.

Accessing data

The current interface does not automatically use the sent data from either of the datasets. By manually replacing the default dataset in the data folder of Processing with a dataset that includes personal data, the effect of personalisation by means of the J48 algorithm can be mimicked.

However, the idea is that at a certain interval (e.g. every 100 new data points or every week), the newest personal dataset is accessed from DataFoundry (by filtering on personal details from the login screen) or local storage which is used to retrain the model. In theory, the interval would not be necessary, but in practise that would mean every time a new instance is appended, the model would be retrained, which would over time harm the speed of the application when the size of the dataset grows too big.

Personalisation

The newest dataset would include a default training set that is used to initially train the model, so it can be used without havin
to train it first, and a personal extension of that dataset (t
feedback loop). Over time, most of the data will be person
making the resulting model also increasingly personal a
should therefore increase the accuracy of the predictions as we

Since each user will have a unique dataset, they will also ha
a unique model. Especially since J48 has an unstable learni
scheme. This gives different attributes the possibility o
becoming more important, and could also change the top node of the model, i.e. the highlighted most essential attribute.

Discussion and Conclusion

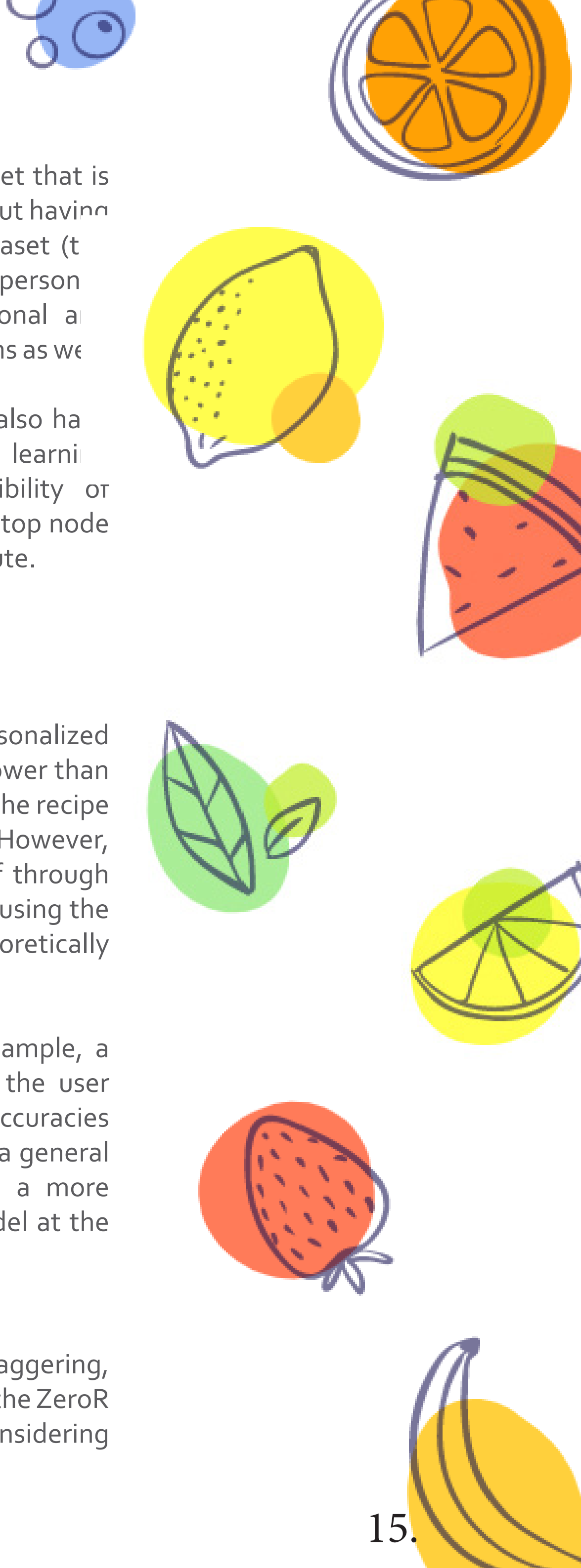
Limitations

Up to now, there remain some limitations in the personalized model. Firstly, the accuracies for the test set are still lower than it is expected, with only a 50,33% correct rate using all the recipe attributes and user attributes in the general J48 model. However, the idea is that the general model personalizes itself through the usage of the app. If the duration and frequency of using the app are guaranteed enough, the latest model will theoretically perform better than the original one.

Moreover, due to the limited variation in the user sample, a problem of overfitting arises in the case of adding the user attributes to the training process. It also leads to low accuracies when testing with the test dataset. In order to obtain a general model that can better fit different types of users, a more diversified user group will be needed to train the model at the very beginning.

Recap of results

Whilst in the end the accuracy of our J48 model is not staggering, it was pleasing, since the J48 still performs better than the ZeroR in the case that the user attributes are not included. Considering



the inclusion of the user attributes leads to overfitting, future work should focus on avoiding such a problem.

As it is mentioned in the Demonstrator section, at the end of the course our prototype has basically possessed the ability to recommend preferred dishes based on a dynamically updating model. It is expected that each user will have a unique model only extracting the most essential attributes for them, thus making the recommendation more personalized.

In conclusion, more value of the AI tool will be seen via this project. It indeed provides more solutions in implementing domain-specific technologies to the domain of design.

Future work

Regarding the further refinement, there is some future work that needs to be done. Firstly, the conceptual design should be linked to the processing interface to associate aesthetic consideration with the functional section. Secondly, given that there is still room for developing the current conceptual design, more research on app design should be launched. Thirdly, since the personalized model will be constantly trained with the personal dataset, it is also of importance to focus on how to automate this procedure. Otherwise, it would be the case that every time the datasets are updated, the algorithm has to be manually adjusted by operators. Fourthly, in order to optimize the model performance, especially in testing session, more concise datasets in terms of the number of instances are needed. Also the dataset should be more broad in terms of user variation. Fifthly, research needs to be done on how to best create the general model using general datasets. The trade-off problem remains to be solved that we do not maintain the validity of user attributes as well as the conciseness of general datasets. Lastly, during the data mining process it was found that raising the minNumObj parameter, which simplifies the J48 model, did not lead to lower accuracy. Therefore for the creation of a general model this could be applied in the future.

References

Catalina, J. (2020, December 09). Colorful Fruits. Free PowerPoint Template &amp; Google Slides Theme. Retrieve January 25, 2021, from https://www.slidescarnival.con aumerle-free-presentation-template/2571

Dulenko, V. (2019, August 10). How Tinder Design Hooks Yc Up. Retrieved January 25, 2021, from https://uxplanet.org/hov tinder-design-hooks-you-up-60201d78501f

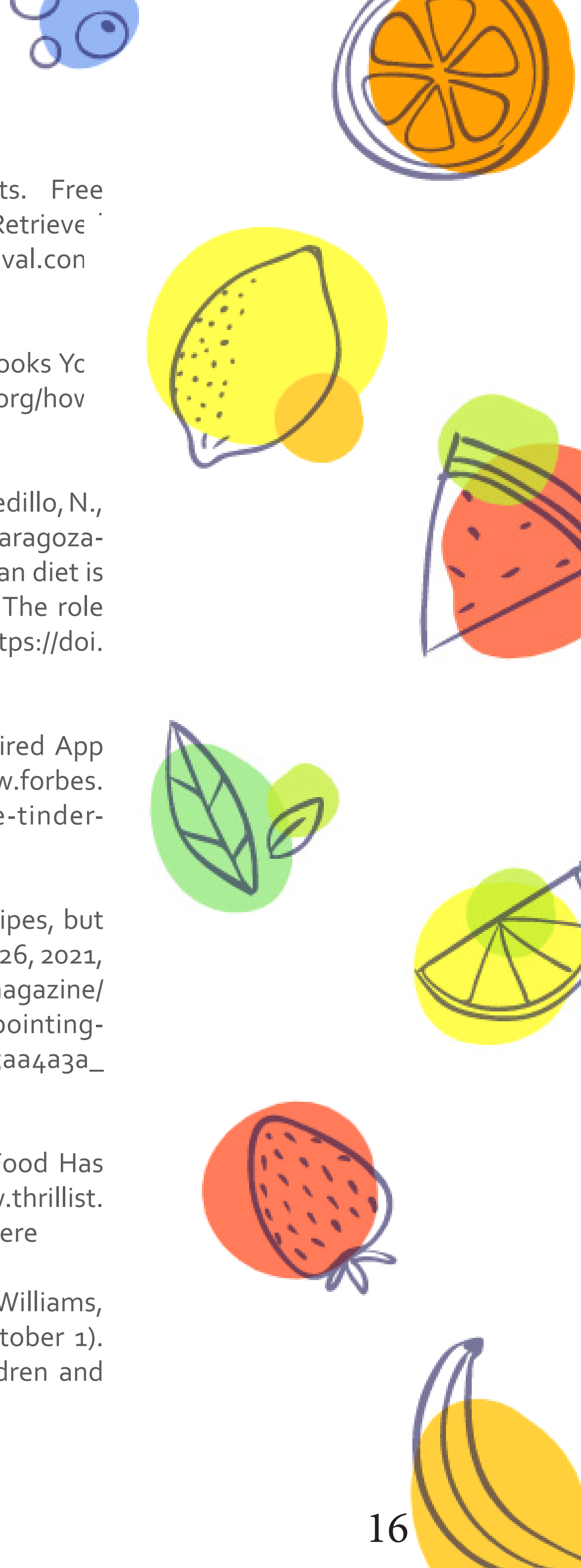
Ferrer-Cascales, R., Albaladejo-Blázquez, N., Ruiz-Robledillo, N., Clement-Carbonell, V., Sánchez-Sansegundo, M., & Zaragoza-Martí, A. (2019). Higher adherence to the mediterranean diet is related to more subjective happiness in adolescents: The role of health-related quality of life. Nutrients, 11(3). https://doi.org/10.3390/nu11030698

Huen, E. (2015, July 30). Get Tender, the Tinder-Inspired App for Food. Retrieved January 25, 2021, from https://www.forbes.com/sites/eustaciahuen/2015/07/30/get-tender-the-tinder-inspired-app-for-food/

Judkis, M. (2015, October 08). Tender is Tinder for recipes, but with similarly disappointing choices. Retrieved January 26, 2021, from https://www.washingtonpost.com/lifestyle/magazine/tender-is-tinder-for-recipes-but-with-similarly-disappointing-choices/2015/09/24/1f308df8-51c4-11e5-8c19-0b6825aa4a3a_story.html

Meilus, L. (2015, July 20). Inevitably, the Tinder for Food Has Arrived. Retrieved January 25, 2021, from https://www.thrillist.com/eat/nation/tender-app-tinder-for-food-is-finally-here

O'Neil, A., Quirk, S. E., Housden, S., Brennan, S. L., Williams, L. J., Pasco, J. A., Berk, M., & Jacka, F. N. (2014, October 1). Relationship between diet and mental health in children and



adolescents: A systematic review. American Journal of Public Health. American Public Health Association Inc. https://doi.org/10.2105/AJPH.2014.302110

Sayers, L., &amp; Sayers, C. (2018). Recipe Database - Airtable Universe. Retrieved January 25, 2021, from https://airtable.com/universe/expHZcS7kWEyq5gUH/recipe-database

Appendix

Appendix A: Processing Code and Datasets - See ZIP file (DBM180_group4_interface)

The processing folder contains the following files:
- Code
- Recipe dataset
- Biased datasets for testing the algorithm's adaptivity
- Default training dataset as a result from data mining
- Images + image references

Appendix B: Informed Consent