Predicting cuisines with machine learning models

Project group 7 | Carola Robbemond (2616637), Bram Vermeulen (2630261), Suzanne Voorn (2627482), Ruben Wapperom (2545415), Thomas Webbers (2560695)

March 30, 2018

Abstract

The aim of this report is to predict the type of cuisine based on the given recipe. Predictions are made using a dataset of nearly 40,000 unique recipes. The experiments in this research show that the best model, based on multinomial logistic regression, performs well with an accuracy of 0.779 which is better than just guessing the most common cuisine all the time. Cuisines that are more present in the dataset are easier to predict.

1 Introduction

In this research project various machine learning algorithms are used. The goal is ultimately to predict what type of cuisine a recipe belongs to, using the ingredients it contains. The data used to train the best algorithm is from kaggle.com and the competition is called What's Cooking. It contains a list of nearly 40,000 cuisine-recipe pairs consisting of a type of cuisine and a list of ingredients. Furthermore there are 20 different types of cuisines in this dataset. In this research many various types of models will be trained to determine the best model for predicting cuisine types. The ultimate goal is to find the best model to predict the kitchen of origin given the ingredients of a recipe.

2 Data inspection and preparation

2.1 Data inspection

The dataset has 6,714 unique ingredients. There are a total of 39,774 recipes in the data set, distributed over 20 cuisines. An overview of the amount of recipes per cuisine can be found in figure 1. The Italian cuisine is clearly the majority class. Other cuisines with a large frequency of recipes in the data are Mexican and Southern-us.

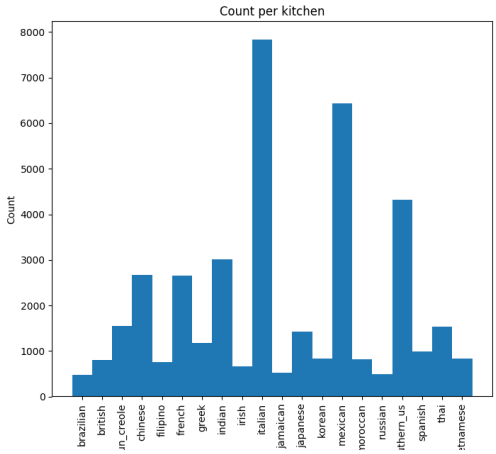

Figure 1: Recipe amount in the dataset per cuisine.

2.2 Data preparation

In order to determine the best method, five different methods are tested. The methods are: k-nearest neighbors, support vector machines, logistic regression, decision trees, and lastly neural networks. These methods do not only use different techniques, they also require different types of data. Because of this, the data had to be transformed to the appropriate format for each method. All of these transformations were done using Python. The What's Cooking dataset is by default a list of recipes with a label which cuisine it is and a list of ingredients used to make each of those recipes.

For k-nearest neighbors and support vector machines the Term Frequency Inverse Document Frequency (TF-IDF) matrix is used, which is explained in the next paragraph.

Support vector machines also use another sort encoding namely binary encoding. The binary encoding of the dataset is a series of vectors where every vector has a length equal to the number of unique ingredients. First the vectors are created as zero-vectors. Next if a recipe contains an ingredient the 0 is changed to a 1 for the index of that ingredient. Logistic regression also uses binary encoding.

Decision trees and neural networks require tensors. Tensors in machine learning are multidimensional arrays. In the case of this research, it turns out that binary encoding suffices as a two dimensional tensor used in both methods.

2.2.1 Term Frequency Inverse Document Frequency

The binary matrix which indicates whether or not an ingredient occurs in a recipe. Without any further calculations, this matrix doesn't give any information how often an ingredient occurs in recipes of certain cuisines and what kind of influence it has in classifying a particular cuisine. An ingredient with a small frequency is a good classifier, because it implies a strong connection with a particular cuisine.

For some machine learning techniques like k-nearest neighbors and support vector machines, this information is besides the binary matrix important input data for the algorithms. The Term Frequency Inverse Document Frequency (TF-IDF) combines both notifications in one matrix. The formula is shown below:

$$w_{\mathrm{i,r}} = tf_{\mathrm{i,r}} \times log(\frac{|R|}{tf_{\mathrm{i,R}}})$$

$tf_{\mathrm{i,r}}$ is the number of times an ingredient i appears in a recipe r. Since each ingredient can only appear zero or one times in the recipes. This is equal to the binary matrix. The second term computes the inverse document frequency with R the total number of recipes and $tf_{\mathrm{i,R}}$ the number of recipes that contain ingredient i. By applying this transformation to the data, a matrix with weights w for ingredient i and recipe r is created. The inverse frequency ensures that ingredients that are less common, get higher weights $w_{\mathrm{i,r}}$ [1].

To obtain the TF-IDF matrix for the What's Cooking data, the TfidfTransformer from sklearn was applied.

3 Methods

3.1 K-Nearest-Neighbors

The k-nearest-neighbor (KNN) algorithm is a method to solve classification and regression problems. The main idea of this algorithm is to classify point x using the K points that are the closest to x. The K points are already classified. Point x is classified as the majority class of the K points.

The nearest-neighbor classifier has a high variance and a low bias, this can result in overfitting when the training data is limited, noisy or not representative. The size of K needs to be determined. The bigger K is, the smoother the boundary becomes, also the bias increases and the variance decreases. When K is very small, the boundaries tend to overfit. As a rule of thumb $K = \sqrt{n}$ is used to determine the size of K, with n the number of instances [2].

The TF-IDF matrix is used as the input data for the KNN algorithm of the What's Cooking competition. The complete dataset doesn't give any results for relatively small K, due to large computation times and memory limitations when the size of K increases. Therefore three smaller samples of the original dataset were tested. These datasets have the size of 20,000, 25,000 and

30,000 recipes and the recipes were randomly selected. After the selection, the datasets were divided into 80% training data and 20% test data.

3.2 Support Vector Machine

Support vector machine (SVM) is a method normally used for binary classification. It works on the principle that it draws a linear line between the data of the two classes that maximizes the margins. A softer approach is to allow a data point of a class to be on the wrong side of the line modeled by a penalty in the objective function also known as the soft margin SVM.

However there is a possibility to use support vector machine for a multi-classification problem. The linear SVM has a one-versus-rest strategy for multi-classification. It trains n binary classifiers. For every class it is calculated if the data instance belongs to that class or not with certain probabilities. It assumes independence between the different classes. From all the classifiers the highest positive value is searched. The class that has the highest positive probability is the classification for the data instance [3].

Another strategy is formulated by Crammer-Singer using kernels. With one-versus-rest, the correlation between cuisines is not considered. However, Crammer-Singer method does take this correlation into account [4].

For this research the Crammer-Singer method is chosen to be used in the model since it often outperforms the one-versus-rest method.

Since the What's Cooking dataset is quite large, the processing time to build this model is expected to be quite extensive because there are 6,714 unique ingredients. Therefore in this research it is chosen to apply principal component analysis (PCA) and reduce the used components to 1,000. Principal Component Analysis transforms the data to a new coordinate system that is in line with the covariance. The first principle component is the line along which the data has the most variance. The remaining variance is used to find the other principle components.

The SVM method does not work on text data like many other methods. The binary-encoding can be used to mark the presence of an ingredient in a recipe. Also the frequency-based TF-IDF matrix of the ingredients used in a cuisine is given by a numerical representation of the data. Both input data structures will be used when testing the support vector machine with PCA model.

3.3 Decision Trees

One of the best known machine learning models is a decision tree. A decision tree can best be envisioned by an inverted tree. This means that the root can be found at the top and the leafs of the tree constitute the terminals. In between there are the so called nodes. At each node a decision is made utilizing the features of a dataset. In this research this is a binary branching at every node [5].

Multiple algorithms can be utilized for running a decision tree model. Among others, there are the ID3, C4.5 and CART algorithms. The first two mentioned algorithms are both devised by Ross Quinlan. He introduced his first model, ID3, in 1968. Later on he published the C4.5 model as an extension to his earlier model, which is able to also deal with classification tasks. Both previously mentioned models aim to get the highest information gain, obtained by lowering the entropy. The algorithm used in this report is CART, introduced by Breiman in 1984. The main reason for this decision is that SciKit Learn uses this algorithm as implementation for their decision trees. Another good reasons could be that CART is able to deal with missing values and that it has an intermediate speed compared to ID3 and C4.5 [5].

The CART algorithm starts out with L_0, which is the top node in the decision tree. From here on the decision tree starts branching. The total dataset is presented by S_0. This all can be formulated as:

$$L_q(A_L^i) = \{s_j \in S_q | v_j^i \in A_L^i\}$$

To control for the impurity at each node, the algorithm makes use of the Gini index [6].

$$Gini(S_q) = 1 - \sum_{k=1}^{K} (p_{\mathrm{k,q}})^2$$

3.4 Neural networks

Artificial neural networks are machine learning techniques that are inspired by the functioning of biological neural networks in brains. Similarly to biological neural networks, artificial neural networks use the "input" of a nervous system, to determine whether or not neuron(s) are triggered. Artificial neural networks generally consist of three layers:

1. Input layer: in this layer the input of the neural network, in the form of an N-dimensional tensor, are fed to the network.
2. Hidden layer(s): in these layers information is extracted from the input layer of the neural network and transformed into new features.
3. Output layer: In this layer the input of the hidden layer is translated into the preferred output scale.

First the training data was transformed into a 2-dimensional tensor using the sklearn package in Python. Then the neural network was designed. Subsequently networks with multiple hidden layers (2 up to 10) were constructed, but this resulted in unreasonable computation time, as the training data is quite large. Thus, it was decided to minimize the amount of hidden layers in the neural network, to guarantee a reasonable computation time. In the end, a neural network with one hidden layer was constructed. For training the network, the optimizer ADAM was used with the recommended default learning rate $\alpha = 0.001$ [7]. Three hyperparameters of the network were tested: the number of neurons in the hidden layer, the activation function of the hidden layer and lastly the dropout rate of the hidden layer. The number of neurons in the hidden layer that were considered are 1, 10, 100 and 1000. For the selection of the amount of neurons, the Relu activation function and no dropout were used. This procedure yielded the following results:

Neurons	1	10	100	1000
Train accuracy	0.197	0.197	0.832	0.197

Table 1: Validation accuracy for 1, 10, 100 and 1000 neurons.

The neural network with 100 neurons in the hidden layer, clearly has the best accuracy. The hyperparameter, amount of neurons in the hidden layer, was set to 100. The next hyperparameter is the activation function. The most commonly used activation functions are Relu, Sigmoid and Tanh [8]. Three neural networks were trained (without dropout) to evaluate the performance of the activation functions:

Activation function	Relu	Sigmoid	Tanh
Train accuracy	0.832	0.783	0.719

Table 2: Validation accuracy for Relu, Sigmoid and Tanh.

The accuracy for the Relu and Sigmoid activation functions are very similar, but Relu performed slightly better. Thus, for the hyperparameter activation function, Relu is chosen. The last hyperparameter is the dropout rate. For this parameter the values no dropout, 10%, 25% and 50% are considered:

Dropout rate	No dropout	10%	25%	50%
Accuracy	0.832	0.682	0.605	0.524

Table 3: Validation accuracy for 0%, 10%, 25% and 50% dropout rate.

Increasing the dropout rate, clearly has negative effects on the accuracy. Thus, the hyperparameter dropout rate, was set at 0%. This means the neural network won't use the dropout functionality in the hidden layer. In conclusion, the selection of the hyperparameters resulted in the following hyperparameters:

Amount of neurons in the hidden layer	100
Activation function of the hidden layer	Relu
Dropout rate of the hidden layer	No dropout

Table 4: Hyperparameters.

3.5 Logistic regression

Logistic regression can be seen as an mathematical modelling approach that helps to describe the relationship between on one side several input variables and on the other side a normally binary dependent variable. The logistic regression model makes use of the logistic function:

$$f(z) = \frac{1}{1 + e^{-z}}$$

This function has a range between zero and one. This is a useful property, especially when wanting to compute probabilities [9]. For binary classification the sigmoid function is used to compute the probabilities of belonging to one of the two classes. The formula to calculate the probability that data point y belongs to class i, using the sigmoid function, is written the following way:

$$sigmoid(y_i) = \frac{e^{y_i}}{e^{y_i} + 1}$$

This method can be adjusted in a way that it calculates probabilities for the data to belong to one of multiple class options. The one-versus-rest logistic regression trains n binary classifiers like in SVM one-versus-rest and select the best of the classifiers after comparing them.

Another option to use logistic regression, when the dependent variable has more than two categories, is multinomial logistic regression. This is also known as softmax logistic regression.

Instead of the sigmoid function the softmax function is used to calculate probabilities when choosing from multiple class options. The formula to calculate the probability that data point y belongs to class i following the softmax function is written the following way:

$$softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

An advantage of multinomial logistic regression is that it does not assume independence between classes [10].

Both logistic regression methods are used to make a prediction for the type of cuisine for certain recipes. Results can be found in the next chapter.

4 Results

To test all the methods, the dataset was split into 80% training data and 20% testing data. To test hyperparameters a validation set was used.

4.1 K-Nearest-Neighbors

The KNN-algorithm is tested on all three datasets, described in the chapter Methods, with different sizes of K. The accuracy results are shown in figure 2.

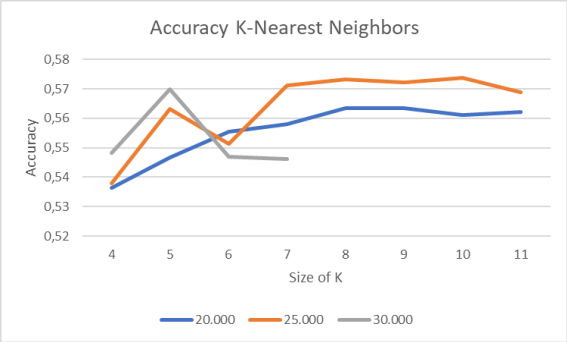

Figure 2: Accuracy tested on three extents of the dataset with multiple k's.

The figure shows that the dataset of 30,000 recipes doesn't give any results when K>7, due to computational limitations. Hence the smaller datasets were tested to see if the accuracy improves when K increases. The highest test accuracy 0.574 with training accuracy 0.671 is accomplished with the dataset of 25,000 recipes and K=10. However this accuracy is slightly better than the dataset of 30,000 recipes, K=5 where the test accuracy is 0.570 and the training accuracy 0.737. The confusion matrix of the latter is shown in figure 3.

The rule of thumb $K = \sqrt{n}$ is also applied on the datasets of 20,000 and 25,000 recipes with K=141 and K=158. The test accuracy results are 0.475 and 0.511 and give therefore considerably worse results than the smaller size of K. The corresponding training accuracy results are 0.493 and 0.509 The confusion matrix of 25,000 recipes and K=158 in figure 4. demonstrates the bias towards Italian cuisine, which is the class with the highest frequency.

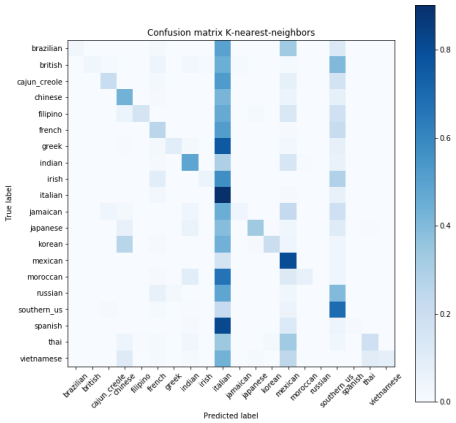

Figure 3: Confusion matrix of 25,000 recipes and k=158.

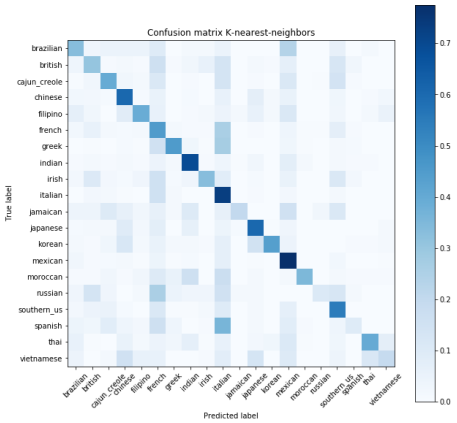

Figure 4: Confusion matrix of 30,000 recipes and k=5.

4.2 Support vector machine with Principal Component Analysis

When testing the SVM method with PCA = 1,000 using the Crammer-Singer solver for multi-classification with the data input encoded using binary encoding, the train accuracy was 0.262 and the test 0.249. The model makes bad predictions. It tends to predict cuisines as Indian and Greek, but the highest probability is around 0.6 which is quite far from the ideal value 1. The confusion matrix can be found in figure 5.

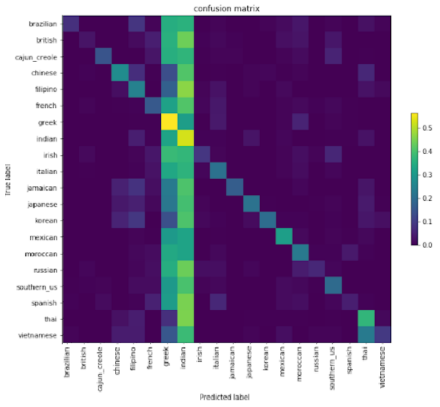

Figure 5: Confusion matrix of SVM with PCA (1000) and Crammer-Singer method on ingredient occurrence data structure.

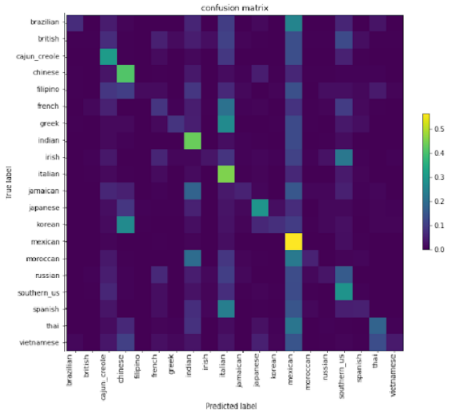

Figure 6: Confusion matrix of SVM with PCA (1000) and Crammer-Singer method on ingredient frequency data structure.

The other data structure that is tested in this research is the SVM method with PCA = 1,000 on, is a data structure that indicates ingredient frequency in a certain cuisine. The solver again is the Crammer-Singer method. The accuracy on the training set was found to be 0.502, the test accuracy was 0.503. Although the accuracy is not very high, it performs much better compared to the SVM model on the ingredient occurrences data structure. The confusion matrix in figure 6 indicates that the Mexican, Italian, Indian and Chinese cuisine are better identified but the probability is still low, around 0.4 and 0.5.

4.3 Decision trees

Multiple tests were carried out to arrive at a reasonable outcome for the model. A first run was carried out only setting the parameter random state to 0. In this way the tests can be performed multiple times, and still getting comparable results. This was carried out once. The result was a train accuracy approaching 1 very closely, and a test accuracy of 0.619.

As can be seen in the section on data inspection, there is a bias in the dataset. Especially a lot of labels contain the Italian kitchen, while the Mexican and Southern-us kitchen are also very present. In an attempt to reduce this class imbalance, the parameter class weight has been adjusted. This is changed into balanced, so that it can slightly make up for the aforementioned imbalance. Now different results show up, yielding a little lower accuracy, which makes sense. The train accuracy again was approaching 1 closely. This time the test accuracy was 0.578.

A last adjustment to the decision tree classifier was to make it finite. This was done with the purpose to prevent overfitting. The parameter now changed is the maximum amount of leaf nodes. This has been tested multiple times, until the values of the accuracy did not change anymore. The train accuracy again approached 1 closely, but the test accuracy was 0.580.

Finally with these settings a confusion matrix was created, which can be seen in figure 7.

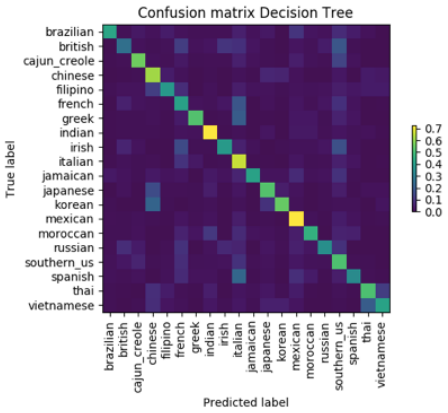

Figure 7: Confusion matrix of the decision tree classifier.

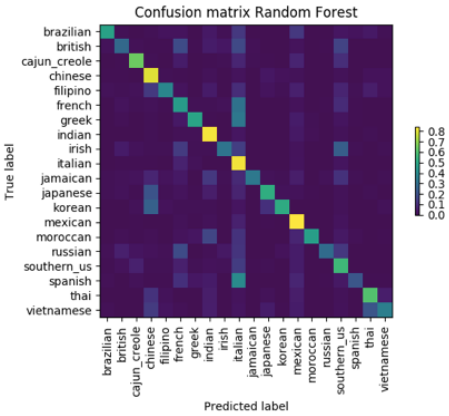

Figure 8: Confusion matrix of the random forest classifier.

Also with the random forest classifier multiple tests were carried out to arrive at a reasonable outcome for the model. However, since this is an ensemble, the default settings also yield a very decent result. The random state parameter was set to 0 for a first run, for the same reasons as for the decision tree. The train accuracy resulting was 0.992 and the test accuracy was 0.673.

Now to make an adjustment to the settings of this classifier and try to get it a little better, it is tried to make up for the class imbalance. Again the parameter class weight was set to balanced. Now the train accuracy was a bit lower, resulting in 0.991, and also the test accuracy dropped a little to 0.659.

Finally with these settings a confusion matrix was created, which can be seen in figure 8.

4.4 Neural networks

The constructed neural network has a test accuracy of 0.727 and a train accuracy of 0.856. The confusion matrix is given by figure 9. The confusion matrix shows a diagonal line. In general this implies that the network predicts very well. However, this doesn't hold for all cuisines, as the Italian cuisine is frequently misclassified as Moroccan. Italy and Morocco are both Mediterranean countries. The network could thus be good at recognizing if a recipe originates from the Mediterranean region, but have some trouble with distinguishing the different Mediterranean cuisines. This holds for different geographical regions. Another example is the Vietnamese cuisine. The Vietnamese cuisine is occasionally misclassified as Indian or Korean. These countries are all Asian and the network might have some trouble distinguishing the different Asian cuisines. In conclusion the network is thus capable of recognizing the geographical region of the cuisine, but might have trouble distinguishing some specific cuisines in a geographical region.

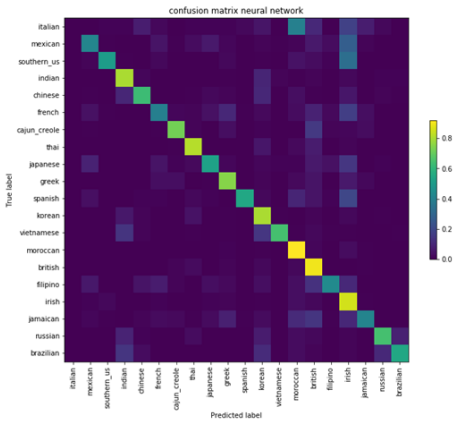

Figure 9: Confusion matrix neural network.

4.5 Logistic regression

First a logistic regression model with the addition of using the one-versus-rest principle, for predicting one of multiple classes, was made. The train accuracy was 0.409 and the test accuracy was 0.402. Indicating that there was no overfitting on the train data, but many cuisines are not predicted right. The confusion matrix figure 10 shows that most recipes are getting classified as being Italian with a high probability. Since Italian is the majority class of the data this can be explained. Only for the Chinese, Indian, Mexican and Moroccan cuisines the probability of predicting the right cuisines is above 0.5 besides the Italian cuisine. This can be due to the fact that those cuisines are more present in the data than other cuisines. However this argument cannot be applied to the Moroccan cuisines.

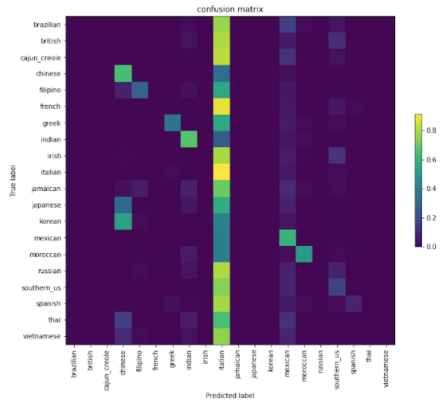

Figure 10: Logistic regression multi class One-Versus-Rest.

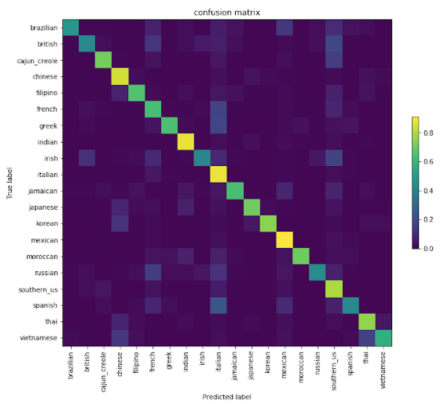

Figure 11: Multinomial logistic regression Newton-CG solver.

Second a multinomial logistic regression model is fitted using the different solvers sklearn offers. The solver newton-CG was found to be the best. It gave a train accuracy of 0.890 and a test accuracy of 0.779.

Newton-CG is a variant of the newton method for high-dimensional problems. The newton method tries to find a good approximation for the root of a real-valued function. An advantage of the newton-CG method is that it maintains the rapid convergence property of the newton method, while the adjustable controls the accuracy of the solution. Besides that it only requires the Hessian vector [11].

Figure 11 shows the confusion matrix for the multinomial logistic regression with the newton-cg solver. It shows that recipes from the Chinese, Indian, Italian and Mexican cuisine are predicted well with a probability close to 1. These cuisines are also more present in the data. Cuisines that are predicted right with less certainty are Brazilian, British, Irish, Russian and Spanish with a probability around 0.4. The other cuisines have probabilities between 0.5 and 0.8.

5 Discussion

The model that gives the best accuracy in predicting the cuisine based on a recipe, being a list of ingredients, was multinomial logistic regression using the newton-CG solver. An overview all the applied methods can be found in table 5 below.

Methods	Accuracy
Knn with k=5	0.570
SVM with PCA=1,000	0.503
Decision tree	0.619
Random forest	0.673
Neural networks	0.727
Multinomial logistic regression	0.779

Table 5: Overview of methods and highest accuracy.

To improve the accuracy in the future, some adjustments in the research can be made. In the data preparation phase, ingredients that imply the same ingredient, but are written down differently are identified as different ingredients. This is also known as word embeddings. For example bay leave and bay leaves are different ingredients in the ingredient vector. Combining these two vectors as one might give better information and better accuracy when processed by the models.

For some ingredients the amount of the ingredients was presented in the dataset. This information was not used in the research, since it was not present for all the data, but could be an extra feature in future research.

During data preparation nothing was done about the class imbalance that is present in the data. The recipes of the Italian cuisine are largely more present in the data set. Only when making decision trees the class imbalance was tried to be balanced. However this gave a poorer accuracy since the majority classes didn't weight as much as before. For future research the class imbalance could be handled during the data preparation by either removing recipes from the majority class or adding recipes from the other classes.

Principal component analysis was used in the support vector machine method. Only a PCA of 1,000 was used. Other amounts of principal components were not tested due to computation time. In future research the number of principal components can be viewed as a hyperparameter and tested using a validation set. Besides PCA is not used for every method, in future research this could be applied for more methods.

For future research, it would be interesting to further investigate the amount of neurons in the hidden layer. Due to limited computation power, exclusively the values 1, 10, 100 and 1,000 were tested. If more computation power was available, a Python script could be written to evaluate the values 1 up to 1,000. Without improved computation power, future research should focus on using PCA to reduce the computation time and should aim to find a balance between the amount of input dimensions, the accuracy of the network and the computation time.

References

[1] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 133–142, 2003.

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[3] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of machine learning research*, vol. 5, no. Jan, pp. 101–141, 2004.

[4] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of machine learning research*, vol. 2, no. Dec, pp. 265–292, 2001.

[5] H. Sharma and S. Kumar, "A survey on decision tree algorithms of classification in data mining," *Vol-5 Issue*, vol. 4, 2016.

[6] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "The cart decision tree for mining data streams," *Information Sciences*, vol. 266, pp. 1–15, 2014.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[8] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[9] D. G. Kleinbaum, M. Klein, and E. Pryor, "Logistic regression: a self-learning text. 2002."

[10] B. Krishnapuram, L. Carin, M. A. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 6, pp. 957–968, 2005.

[11] Y. Tsuboi, Y. Unno, H. Kashima, and N. Okazaki, "Fast newton-cg method for batch learning of conditional random fields.," in *AAAI*, 2011.