# Programming assignment

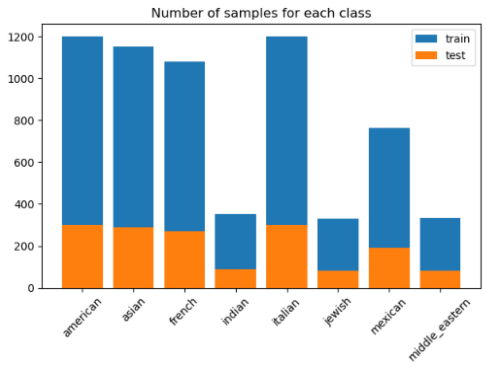
Recipe's classification – Inchingolo Michele 482748

## I. Problem identification

The goal of the programming assignment was to build a model that can recognize the correct cuisine country of a given recipe. Since the number of the possible cuisine is 8 than this machine learning problem belong to the multiclass classification problem and so proper models have been selected to be used. Hence the models that I chose are:

1. Multinomial naïve bayes: simple, fast and easy interpretable model. Since the computed weights parameters give us an idea of how much a certain word is distributed among classes' documents, so it is possible looking at the weight to determine which words are more, or less, relevant for a class.
2. Multinomial logistic regression: classic model for this kind of problem. As discriminative model it has been chosen in order to perform a comparison with the naïve bayes, that is a generative model. Indeed, it result to be also more complex and tuneable thanks to grid search selection on hyperparameter.
3. Support vector machines: also, a discriminative model used with different kernels in order to test if more complex and non-linear decision boundary could improve the accuracy.
4. Multilayer perceptron classifier: this model probably is the most complex and less interpretable respect to the others but has been chosen in order to test if complex architecture could lead to better results.

## II. Data exploration and analysis




Scikit-learn provides mainly two ways for text vectorization: one through **TfidfVectorizer** class and one with **CountVectorizer** class. The first class, despite the second, does not provide the occurrences of a certain feature but gives the **tf-idf** statistics instead. This measure is composed by two quantities: **tf** and **idf**. The first represent how much a word is present within a document and the second instead represent the distribution of the word among all the documents. Then td-idf statistic is the product of these two measures. The most frequent words are those who have a lower idf and a relatively high tf (calculated for each feature taking the mean on documents). In fact, these words represent the ones that are distributed among many documents and with a high frequency. This analysis has been performed for each class, and then the words that were in common in all classes have been added to *stopwords.txt* file, so words like "minutes", "cup", "add" etc. have been added, that are intuitively common words that I attended to find in recipes. This analysis in my idea aim to remove words that are not useful for discriminate among classes. The histogram above shows that samples for each class are scarce, maximum 1200, so no validation set has been created because removing samples from train set could have a relevant impact on model performance.

## III. Features selections

Since the dataset is composed by text documents it was necessary to use the bag of words representation for transform documents into numerical feature vectors, composed by the occurrences of the most recurrent n-words among the whole data set. Before doing this each recipe has been properly formatted eliminating unnecessary punctation, symbols and numbers. The files have been read with UTF-8 encoding since special characters, like Chinese or Korean ideogram, are present. Then tokenization has been performed and all these tokens are fed in input to **CountVectorizer** class of **scikit-learn** in order to perform the bag of words on documents and simultaneously create the vocabulary composed by the most recurrent n-words. During the tokenization process have been discarded all tokens of length lower than two along with the option to perform also stop words elimination and stemming. So, the number of vocabularies created depend by the number of features, by the application of stemming and/or stop words discarding and by which section of recipe has been considered (only ingredients or all recipe text). The possible number of features selected are 1000, 3000, 5000, and 10000, for 32 possible vocabulary configurations. For each vocabulary configuration will correspond a data set, that will be used to train a model. Potentially, taking also into account the model selection with hyperparameters tuning, many models could be trained and evaluated. For these reasons I decided to use model implementation provided by scikit-learn that are well optimized and faster.

## IV. Models implementations

As I said before for each vocabulary configuration correspond a dataset to which correspond multiple models, in relation to hyperparameter selection. So, for each data set has been selected the best model, that is the one that gave the highest accuracy

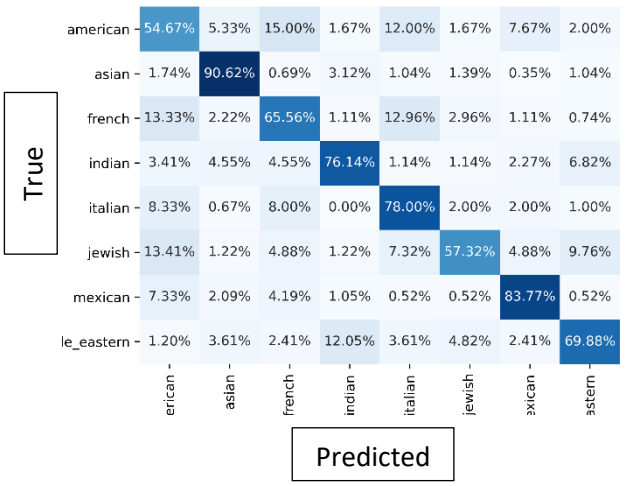
on test set among all the possible models configuration. This approach has been used for all model types. This models selection has been performed with **Parfit** library, that takes as input a grid, with hyperparameters values and scikit-learn model implementation. The results of these comparison will be shown through plots in next sections and are constructed in this way: two trends one for each section of recipe, on x axis are present the vocabulary configurations and on y axis are present both train and test accuracy for the model that has been selected as the best from the model selection, for that dataset, among all the possible hyperparameters combinations.

# V. Multinomial naïve bayes

This model is the simplest and faster among all the models and give us the possibility to easily interpret the weights obtained. In fact, these weights represent how much a certain word is present in relation to the other features, among the documents of its class. Since these ratios are fed into a logarithmic function all the weights will be negative, and more a weight is near to zero more frequent that word will be. This model does not need to change any kind of hyperparameter, so its performance will mainly depend on the type of vocabulary configuration. Here are present the results of my experiments on test set, without normalization:

| | All recipe's text | | | | Only ingredients | | | |
|---|---|---|---|---|---|---|---|---|
| features | Basic | Stop words | Stem | Stop words and stem | Basic | Stop words | Stem | Stop words and stem |
| 1000 | 65.29 | 66.85 | 66.66 | 68.60 | 72.35 | 71.22 | 70.72 | 70.84 |
| 3000 | 69.78 | 70.97 | 71.1 | 72.47 | 72.34 | 72.66 | 72.03 | 72.28 |
| 5000 | 70.41 | 72.16 | 71.78 | 72.53 | 72.22 | 72.40 | 72.22 | 72.60 |
| 10000 | 71.10 | 72.03 | 71.91 | 72.90 | 72.22 | 72.40 | 72.22 | 72.60 |

Each column corresponds to a configuration subdivide by the section of interest of the recipes. First column considers all words, second exclude stop words, third means that stemming has been applied and the last four column both stemming and stop words filter has been applied. As expected, increasing the number of words in vocabulary increase accuracy and apply stop words filter and stemming result to be a good idea. Despite the best result has been obtained considering the whole recipe's text, it is possible to notice that considering only ingredients section will return good results in prediction too.




To the side has been shown the confusion matrix for the best model, normalized over the rows of the matrix, in order to return the recall for each class. One conclusion that I can make observing the matrix is that the most characterized class (Italian, Asian, Mexican, Indian) have highest recall despite to the others, probably for the presence of typical preparation or ingredients that are not present in other cuisine. Instead, American cuisine, despite the relative high number of training samples, resulted to be one of the worst predicted class. This could be due to the mixture of culture present in that country. Bad accuracy also could be linked to a scarce training set such as for Jewish cuisine. These observations are also supported by looking at the weights of the model for each class. Below are reported the 10 most common words for the best model.

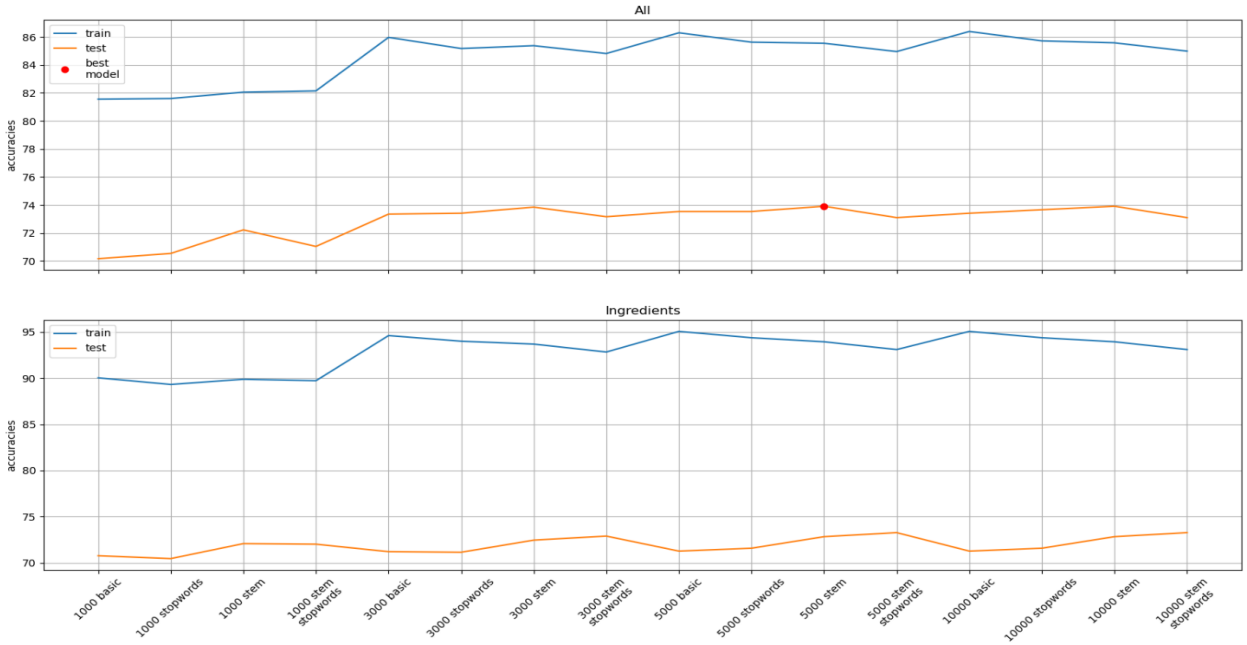
| American | Asian | Indian | French | Italian | Jewish | Mexican | Middle eastern |
|---|---|---|---|---|---|---|---|
| Bake \| -4.47 | cook\|-4.38 | Chicken\| -4.84 | bake\| -4.68 | bake\| -4.87 | bake\| -4.61 | chile\| -4.35 | cook\| -4.81 |
| Butter\|-4.42 | fresh\| -4.68 | cook\| -4.36 | butter\| -4.36 | cook\| -4.31 | cover\| -4.97 | cook\| -4.72 | cover\| -4.91 |
| Cook \|-4.91 | inch\| -4.58 | cover\| -4.84 | cook\| -4.74 | fresh\| -4.88 | water\| -4.88 | cover\| -4.88 | fresh\| -4.84 |
| Cut \| -4.79 | rice\| -4.60 | fresh\| -4.59 | egg\| -4.79 | garlic\| -4.71 | egg\| -4.55 | fresh\| -4.65 | garlic\| -4.81 |
| Fresh\|-4.83 | Sauc\|-4.02 | ground\| -4.84 | inch\| -4.56 | inch\| -4.65 | inch\| -4.80 | garlic\| -4.736 | ground\| -4.89 |
| Inch\| -4.43 | stir\|-4.29 | inch\| -4.68 | mixtur\| -4.87 | oliv\| -4.70 | mixtur\| -4.84 | inch\| -4.80 | juic\| -4.89 |
| Sugar \|-4.50 | sugar\| -4.76 | onion\| -4.77 | pan\| -4.81 | pasta\| -4.70 | oven\| -4.92 | onion\| -4.42 | lemon\| -4.75 |
| Mixtur\|-4.86 | Tablespoon\|-4.66 | seed\| -4.58 | stir\| -4.67 | stir\| -4.36 | pan\| -4.70 | stir\| -4.70 | onion\| -4.89 |

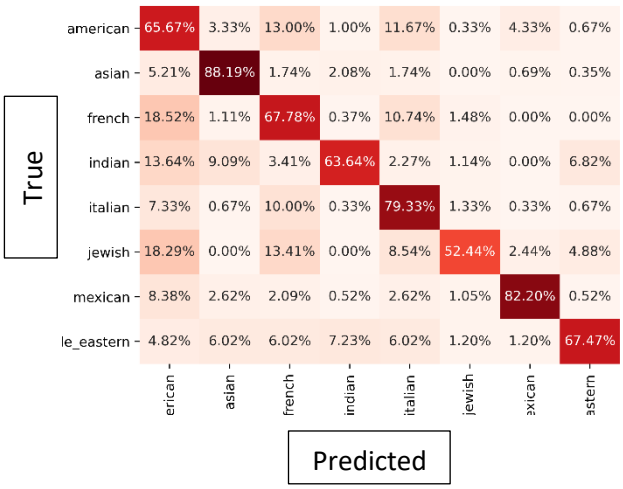
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pan\|-4.86 | Water \|-4.33 | stir\| -4.21 | sugar\| -4.53 | tomato\| -4.83 | stir\| -4.99 | tortilla\| -4.60 | stir\| -4.63 |
| Stir\|-4.55 | Slice \| -4.74 | water\| -4.64 | water\| -4.79 | water\| -4.54 | sugar\| -4.59 | water\| -4.81 | water\| -4.43 |

In fact, looking at the table it is possible to notice that Italian, Indian, Asian and Mexican had some high relevant words, highlighted in red, that do not belong to any other cuisine and this for sure will help to better distinguish them from the others, instead American and French for example share different relevant words and so are more difficult to distinguish between them.
With L2-normalization the best model reached 66% on test accuracy with 1000 number of features, stop words discarding, no stemming and considering only ingredients section despite to the previous best model that needed 10000 features and, for its best, all recipe's text was considered. Then train accuracy with normalization was 67% and without 79%, so no overfit occurred.

## VI. Multinomial logistic regression

For this model now it is possible to select hyperparameters in order to perform model selection. In this case the parameter that I choose with scikit-learn implementation was only the inverse of the regularization coefficient for l2 penalty, indicated as C. So, the values used are 1, 0.01, 0.001, 0.0001, that inverted led to experiments with strong regularization, trying to achieve a satisfying generalization. The maximum number of iterations has been set to 1000. The graphs below show the accuracies trend both on train and test and in red the model with highest test accuracy is highlighted, without normalization. The best model reached a 73.90% in test accuracy with a C coefficient of 0.01, that means a lambda of 100. Something that it is possible to notice is that this kind of model perform better respect to naïve bayes on train set, especially for the ingredients section for which model seems overfit, but on test set performances are almost the same, with the difference that logistic regression requires a shorter vocabulary to reach its best, 5000 words respect to 10000 of naïve bayes.



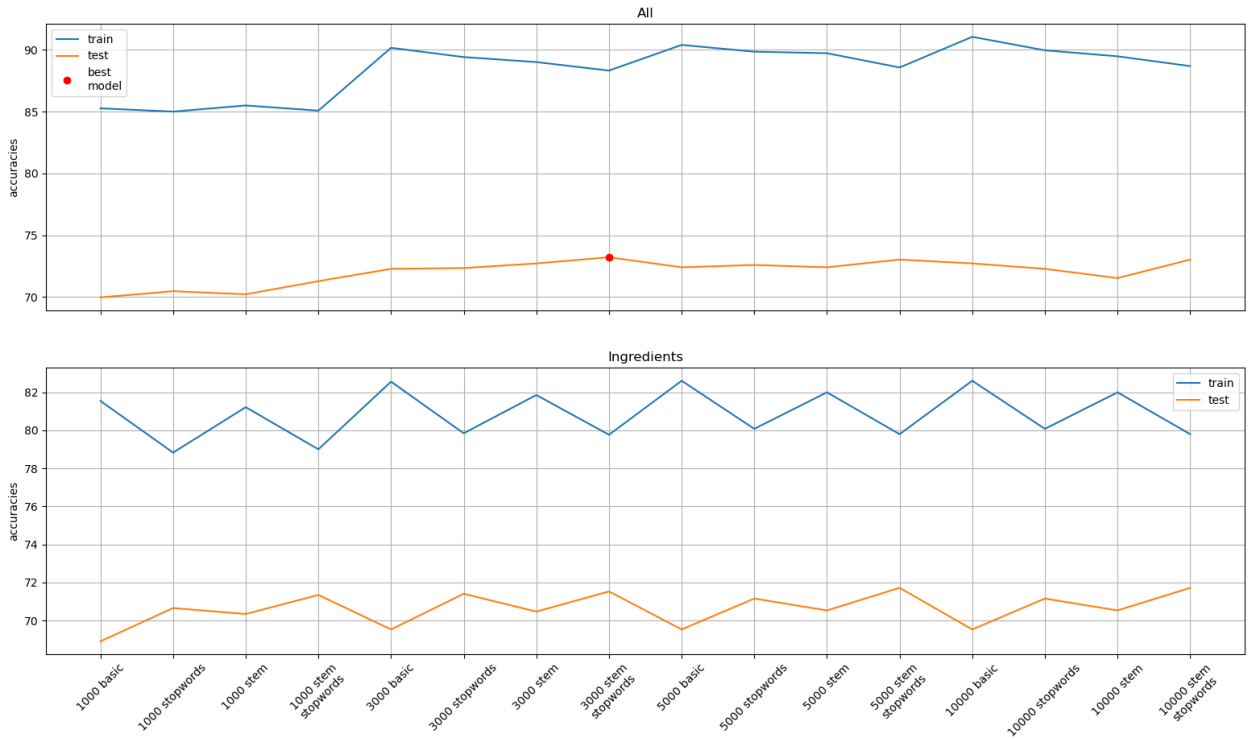



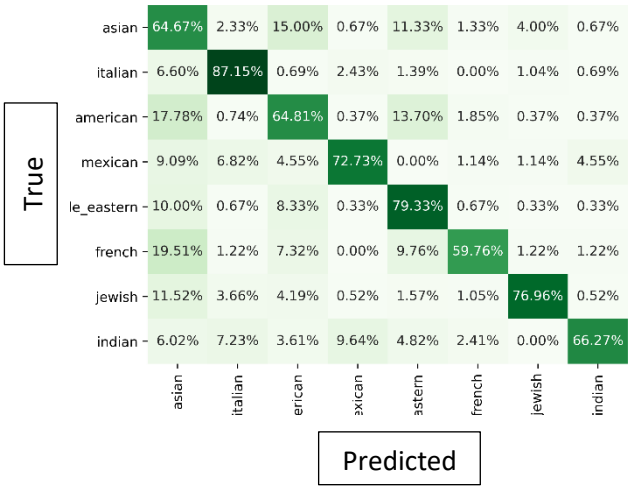

Here on the left has been shown the confusion matrix for the best model normalized over the rows, getting recall for each class. This time the performance for the Indian class has gotten worse despite for Italian, Asian and Mexican that remain the most distinguishable cuisine. Experiments with L2-normalization have been performed and the best model reached on test 72.40% with 76.60% on train with 1000 features discarding stop words, with stemming, and C of 1 considering only ingredients section.

## VII. Support vector machines

This time the scikit-learn implementation offers the possibility to change different hyperparameters, for my experiments I selected the 1, 0.01, 0.001, 0.0001 values for the inverse regularization coefficient for l2 penalty, indicated as C, so small values mean

strong regularization, like in logistic regression, then different kernel (linear, polynomial and radial), the one versus rest strategy for multiclass-classification and a maximum number of iterations equals to 1000.
As I said values for C coefficient have been selected values in order to progressively give more importance on regularization terms, that for support vector machine models means larger-margin hyperplanes, hence giving less importance to correctly classify samples, trying to reach a higher generalization. General expression for polynomial kernel is: $(\gamma\langle x, x'\rangle + r)^d$, so setting properly parameters it is possible to obtain both linear and higher degree polynomial kernels. To obtain linear kernel *gamma*, *r* and *d* have been set one, instead for polynomial *d* have been set 3, 5, 7 with *gamma* and *r* left one. Furthermore, also radial kernel has been tested. Given the expression for radial kernel: $e^{-\gamma\|x-x'\|_2}$, *gamma* parameters tested are 1, 0.01, 0.001, 0,000000001. In this case gamma determine how far the importance of training sample reaches, and for low values also distant samples counts and so generalization of the models increase, instead with greater values of gamma nearest training sample counts more and so this could led to model that fit too much training data causing overfit. Below the trends are plotted and the best model reached test accuracy of 73.22% with linear kernel and a C coefficient of 0.01 that means a lambda coefficient of 100, without normalization.



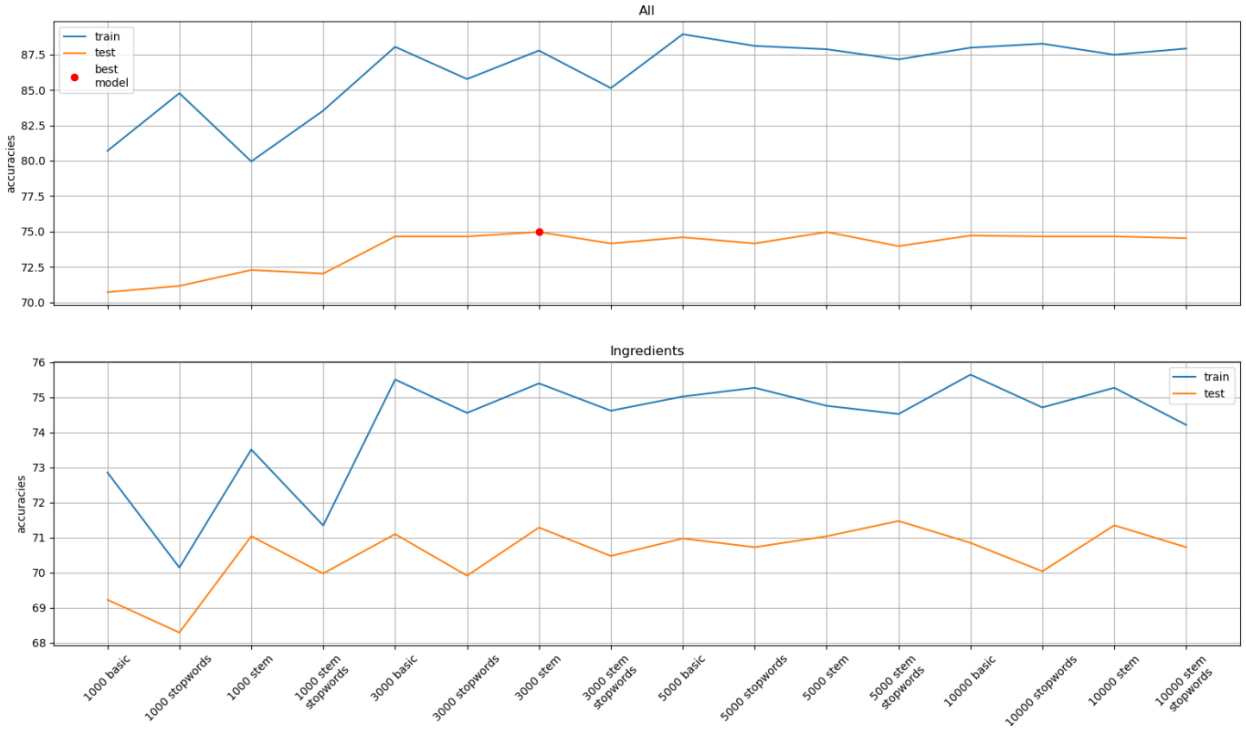



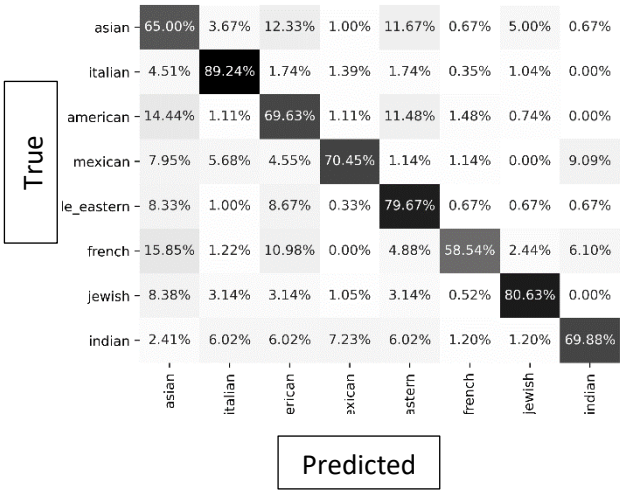

The confusion matrix for the best model. Respect to the others models only Italian cuisine remains the most distinguishable. From the matrix it is also possible to notice that classes with a few numbers of training samples, such as Jewish and Middle Eastern, have increased their recall respect to previous models. With L2-normalization the best model reached 76.27% of test accuracy but with a higher train accuracy of 99.99%, with 5000 number of features with stop words discarding, stemming considering all recipe's text, with C of 1, polynomial kernel and gamma 1.

## VIII. Multilayer perceptron

The hyperparameters for multilayer perceptron are 1, 0.001, 1000 for the regularization coefficient for l2 penalty, a momentum of 0.9, a batch size set to "auto" option for scikit-learn that means minimum between 200 and numbers of samples, "relu" as activation function, initial learning rate of 0.0001, this value could be changed during training phase if there is no significative decreasing in loss function. All these parameters have been tested with three different architectures all having one hidden layer but different number of neurons, 10, 20 and 50, respectively. Stochastic gradient descent has

been selected as optimization algorithm. In this case the best model reached 74.97 % on test accuracy with one hidden layer and 50 hidden neurons and regularization coefficient of 1, without normalization.




Here the confusion matrix for the best model and similarly for support vector machines the classes with less samples improved their recall such as Jewish and Middle eastern.
With L2-normalization the performances worsen and on test set only 46.94% of accuracy for best model with one hidden layer and 10 hidden neurons, and regularization coefficient of 0.001, with 1000 features discarding stop words, with stemming and considering ingredients section.

## Conclusions

Among all models the one that reached the highest test accuracy was the best support vector machine model with normalization but looking at the train accuracy this model seems to overfit, so multilayer perceptron model with no normalization resulted to be the best model with test accuracy of 74.97 %. Despite its simplicity naive bayes implementation performs quite good for this classification problem. Comparing the results obtained from all the recipe's text with only ingredients section, it is possible to obtain also good results in prediction considering only ingredients. Normalized models seem to prefer smaller vocabulary that focus only on ingredients section. With more training data for each cuisine country the results could be better.

## Run instructions

naive_bayes_clf.py, log_reg_clf.py, svm_clf.py, mlp_clf.py training phase with validation on test set and model selection for naïve bayes, logistic regression, support vector machines and multilayer perceptron, respectively. Results are saved in json files in properly directories under *models* directory. L2-normalization have been performed using **pvml** library and for experiments the code has been manually modified and results are under *normalization* directory.
vocabulary.py have been used for properly read and format recipes text and for load vocabulary that are saved on disk according to a proper selected configuration. DataSets.py takes formatted text and create dataset, along with vocabulary if this not already exist, and save them on disk or load them in case of existent ones. data_exploration.py has used for data analysis and exploration, before starting to train models. utils.py contains useful methods along with methods for plotting accuracies graphs.

*I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.*