Cuisine Classification

By

Mili Shah – 13BCE105
Naisargee Vora – 13BCE130

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

Cuisine Classification

Mini-Project 3

Submitted in fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Engineering/Information Technology

By

Mili Shah – 13BCE105
Naisargee Vora – 13BCE130

Guided By
Dr. Priyank Thakkar
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

CERTIFICATE

This is to certify that the Mini-Project entitled "Cuisine Classification" submitted by Mili Shah(13bce105) and Naisargee Vora(13bce130), towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. Priyank Thakkar
Associate Professor
Department of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

Dr. Sanjay Garg
Dept. of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

We would like to thank our guide Dr. Priyank Thakkar, Department of Computer Engineering, Institute Of Technology, Nirma University, for his constant guidance and encouragement. We would also like to thank our parents and friends for their support and appreciation and Nirma University for giving us the opportunity to work on this project.

ABSTRACT

The project is to categorize the cuisine using recipe ingredients. This project is based on a playground competition on "Kaggle" (names "What's Cooking") which asks to predict the category of a dish's cuisine given a list of its ingredients. The focus of project is to build a classification model that can successfully classify the food dishes (based on the ingredients used) into cuisines they belong to. The project is mainly based on the supervised learning, which implements vaious classification models to improve prediction accuracy.

TABLE OF CONTENTS

	Certificate	3
	Acknowledgement	4
	Abstract	5
	Table of Contents	6
Chapter 1	**Introduction**	
	1.1 General	7
	1.2 Objective of study	8
Chapter 2	**Data**	
	2.1 Data Source	9
	2.2 Data	9
Chapter 3	**Classifiers**	
	3.1 Logistic Regression	11
	3.2 Support Vector Machines	11
	3.3 Decision Trees	12
	3.4 Random Forest	12
	3.5 AdaBoost	13
Chapter 4	**Models**	
	4.1 Model 1	14
	4.2 Model 2	19
	4.3 Model 3	23
	4.4 Model 4	23
	4.5 Model 5	23
	4.6 Model 6	23
	4.7 Model 7	24
Chapter 5	**Libraries**	
	5.1 Introduction	25
	5.2 Numpy	25
	5.3 Pandas	25
	5.4 Scikit-learn	26
	5.5 cPickle	26
Chapter 6	**Conclusion**	**27**
References		**28**

Chapter 1: Introduction

1.1 General

While strolling through local, open-air market, What do one smell? What one can make of that smell? Can one determine the cuisine just based on that smell? If yes, then is it just ingrediens or recipe of the dish ? Is it really possible to tell the cusine based on the ingredients used in the food dish ? If yes, then why can't mchine do it ?

In this project, we try to answer above questions, by developing a classification model that attempts to categorize food dish into 20 diffrent cuisines based just on the ingrediennts involved in the recipe.

The project has been implemented in python 3.4. Various pythiin libraries are used such as numpy, pandas, pickle, scikit-learn, pybrain.

The data is taken from the dataset provided by kaggle. Data is processed to extract features. Document term matrix is developed on extracted features. This DTM is then trained by classifier to predict cuisine.

In order to predict cusine one needs to determine what are the features on which it depends and which of them are important for classification. This is called Feature Extraction, extracting importamt features that affects most in determining the cuisine. After deciding these features we developed a document term matrix to clearly distribute feature in all the trainning tuples. This

document term matrix is then passed to a classifier which is trained by provided trainning tuples.

1.2 Objective Of Project

The objective of project is to categorize the cuisine using recipe ingredients. This project is based on a playground competition on “Kaggle” which asks to predict the category of a dish's cuisine given a list of its ingredients. The focus of project is to build a classification model that can successfully classify the food dishes (based on the ingredients used) into cuisines they belong to.

Chapter 2: Data

2.1 Data Source

Data source of this project is Kaggle. Kaggle is a platform which hosts various machine learning competitions. According to wikipedia :

"**Kaggle** is a platform for predictive modelling and analytics competitions on which companies and researchers post their data and statisticians and data miners from all over the world compete to produce the best models. This crowdsourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know at the outset which technique or analyst will be most effective."

This project is based on one such compettion named "What's cook ?". Link for tha same is : https://www.kaggle.com/c/whats-cooking

2.2 Data

The dataset downloaded from kaggle consists of two JSON file :

1. train.json : consists of 39774 tuples
2. test.json : consists of 9944 tuples

Total number of types of cuisines are 20.

Each training tuple contains id, cuisine and list of ingredients. Each test tuple contains id and list of ingredients.

Example of training tuple :


```
{
        "id": 24717,
        "cuisine": "indian",
        "ingredients": [
        "tumeric",
        "vegetable stock",
        "tomatoes",
        "garam masala",
        "naan",
        "red lentils",
        "red chili peppers",
        "onions",
        "spinach",
        "sweet potatoes"
        ]
},
```


Example of test tuple :


```
{
  "id": 18009,
  "ingredients": [
    "baking powder",
    "eggs",
    "all-purpose flour",
    "raisins",
    "milk",
    "white sugar"
  ]
 },
```

Chapter 3: Classifiers

The following classifiers were used in this project:

3.1 Logistic Regression

Binary Logistic Regression simply builds a linear function. This function takes as input a feature vector and outputs a number. A threshold value on this output values separates the two classes. If the output is above the threshold, it belongs to class A, otherwise to B. If there n classes (n > 2), n binary classifiers have to be trained where for each class there is a classifier that tells whether the given tuple belongs to that class or not.

The function is trained by an algorithm called gradient descent. Here, the features and their combinations are taken as variables in the function and their co-efficients are called parameters. These co-efficients are initialized to some values and then the function is tested on all the tuples. Based on the errors in the output of the tuples, the value of a function called the cost function is calculated. The parameters are then 'corrected' (changed according to the output). These changes are determined by how changing them affects the cost function.

This classifier is used often to determine a baseline performance, that is, as a baseline to measure the performance of other classifier functions.

3.2 Support Vector Machines

These classifiers find a plane, called the supporting hyperplane or the decision boundary which separates the data points belonging to

different classes. Similar to logistic regression, these are binary classifiers, so that if there n classes (n > 2), n binary classifiers have to be trained where for each class there is a classifier that tells whether the given tuple belongs to that class or not.

Several methods to separate the data points can be used. This is called using ‘kernels’. These kernels are simply functions that enable us to make more complex decision boundaries.

3.3 Decision Trees

One of the most popular classifiers is the decision tree classifier. This classifier builds a ‘decision tree’. The leaf nodes of this tree are the classes whereas the non-leaf nodes are the ‘splitting criterion’. These criterion are basically conditions which decide which of its child a tuple must go to next. The tuple enters from the root node. Now based on the condition at the root node, the tuple goes to the next node and finally reaches the leaf node which is a prediction of the class it belongs to. So, this tree is basically like a flowchart often very similar to one human expert would make from the data. However, decision trees are prone to overfitting.

3.4 Random Forest

A random forest is an ensemble of such decision trees, that is, the classifier trains a number of decision trees, often from random subsets of the dataset and takes the mode of the predictions that all these decision trees make. This helps in reducing overfitting.

3.5 AdaBoost

Boosting algorithms are algorithms that train multiple classifiers while learning from the previous classifiers and improving for each successive classifier. Each classifier is of the same type.

AdaBoost (from adaptive boosting) selects a weak classifier, often a shallow decision tree, and trains multiple instances of that classifier. Initially, it assigns equal weights to all tuples. For each classifier, it calculates the error rates. It increases the weights of the misclassified tuples and decreases those of properly classified tuples. The successive classifier uses these weighted tuples for training.

Of these, SVM gave the best results.

Chapter 4: Models

4.1 Model 1

The overall approach of model 1 is shown in Figure 1.

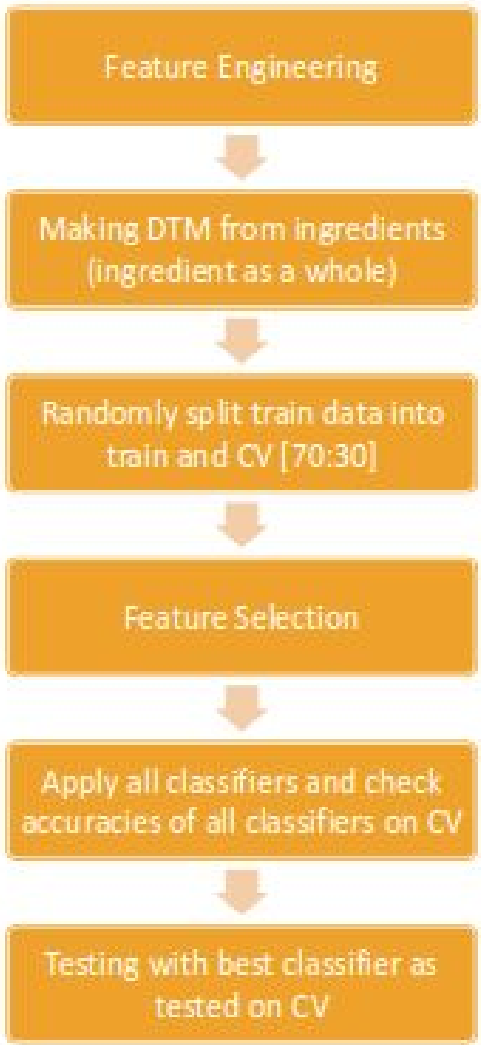

Figure 4.1.1. Model 1

Training:

First, a pandas data frame was constructed with columns containing the id of the dish, the cuisine of the dish, and the column for ingredients where each cell was a list of ingredients. Then some preprocessing (feature engineering) was done on all the ingredients. After being preprocessed, a document term matrix was made of the ingredients. The dtm was simply a count of the bag words, implemet using CountVectorizer of scikit learn. For this, for each ingredient, the individual words were joined by underscores, so that the whole ingredient was taken as a feature, and not the words separately.

Once the dtm was made, it was split randomly, along with the ids of the dish and the cuisines in two sets with 70% of all the tuples in the first, which was used as train data, and the other 30% as validation data.

Then, the train data was put in a pipeline, wherein first feature selection was applied, and then a classifier was applied This model was then tested on the cross-validation data and the accuracy noted. This was using with a number of different classifiers. The classifiers that had some random element were trained a number of times. Of all these, the classifier which on training gave the best accuracy on the validation data was further selected for testing.

Feature Engineering:

The feature engineering consisted of the following :

- Clean of data by removing extra spaces, non-alphanumeric characters and weights (For eg. (16 0z.)).
- From each ingredient, keep only the words that are nouns, or are words are not recognized as English. This was done with the help of POS tagger available in scikit learn.
- For certain ingredients, keep only keywords (eg . cheese, flour, etc.) These keywords were hand-picked manually by observation of data. For each such keyword, after applying the above two, only that keyword was kept using regex, without trying with all the other keywords. If by processing this keyword, there was any improvement in the accuracy obtained by testing on the validation data, this processing was kept in the final model training, otherwise the keyword was not used in the final training. Chicken and salt were the keywords whose processing led to better results and so these were kept, and other discarded.
- Remove some common words that were not removed by the POS tagger and appeared very frequently in a lot of ingredients. These too were hand-picked manually by observation of data. (fresh, low fat, etc.) For each such common word, after applying the first two steps, only that keyword was removed using regex, without trying with all the other keywords. If by processing this keyword, there was any improvement in the accuracy obtained by testing on the validation data, this processing was kept in the final model training, otherwise the keyword was not used in the final training. Leaves, large, fresh, shredded, plain, crushed,

medium and ground were the keywords whose processing led to better results and so these were kept, and other discarded.

- Change certain keywords – denote all the words that might possibly point to the same thing by the same word. These too were hand-picked manually by observation of data. (Eg : lime to lemon). This did not give any improvement, in fact, the accuracy was worsened, and hence this was not used.
- Stemming and Lemmatization:

  Three of the stemmers that were available in scikit learn were used :

  1. Porter Stemmer
  2. Lancaster Stemmer
  3. English Stemmer

  One lemmatizer was also tried - WordNet lemmatizer.

  All of these were tried one-by-one. However, none of these resulted in any improvements, in fact, the accuracy was worsened, and hence this was not used.
- Noise Removal :

  A graph was plotted which is shown in Figure 2. This graph shows the mean counts of the most common ingredients in each cuisine class. As is clear from the figure, the number of times the ingredients appear for each cuisine vary across the cuisines, and therefore, removing the most common ingredients would not very useful and would only result in loss of good features. Hence, this was not done.
- Count of ingredients :

  Figure 3 shows the box plot of number of ingredients of each dish, in each cuisine. As is clear from the figure, the number of ingredients for each cuisine vary widely, but the range of number of ingredients across all

cuisines is nearly the same. Therefore, including the count of ingredients in a dish would not very useful and would only result in extra noise. Hence, this feature was not used.

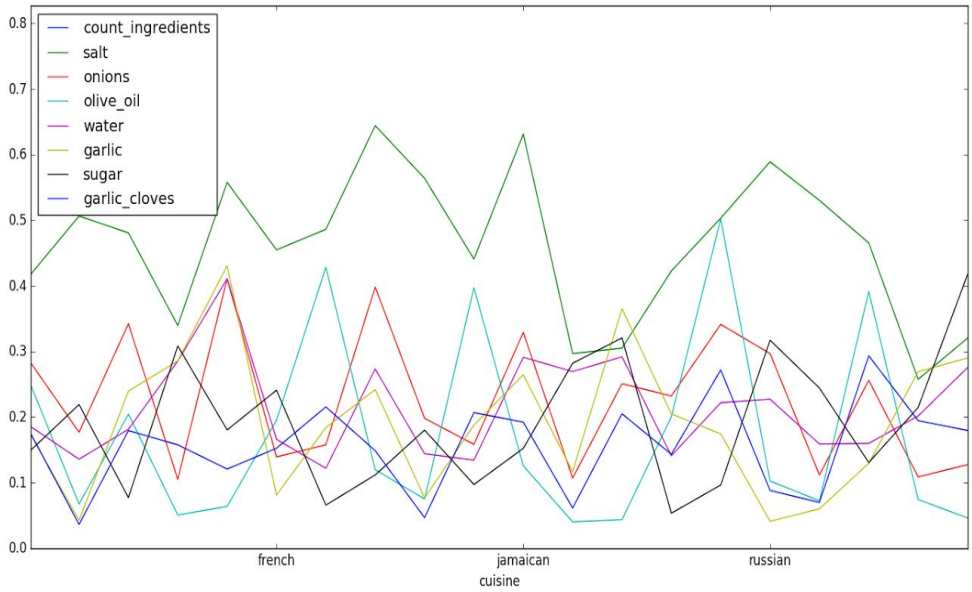

Figure 4.1.2 The mean counts of the most common ingredients in cuisine class

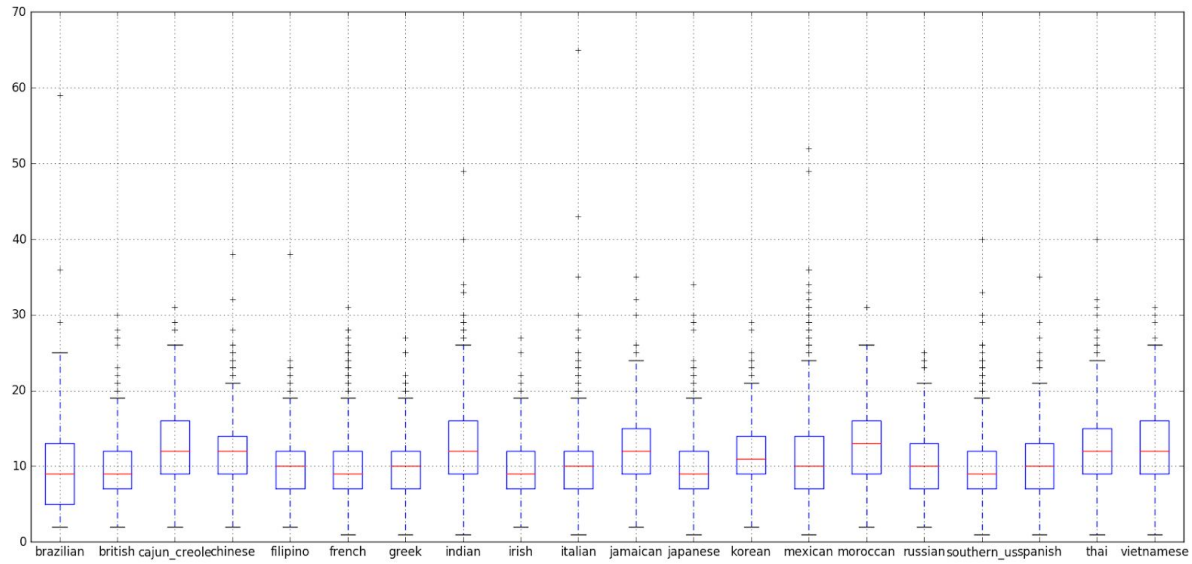

Figure 4.1.3 Box Plot of number of ingredients of each dish, in each cuisine

Testing:

The same preprocessing that was applied to the train data was applied to the test data. Then, the best classifier as tested on validation data was used, that is, the SVM classifier. This resulted in an accuracy of 76.881

4.2 Model 2 - Two Level Classifiaction

The overall approach of model 2 is shown in Figure 1.

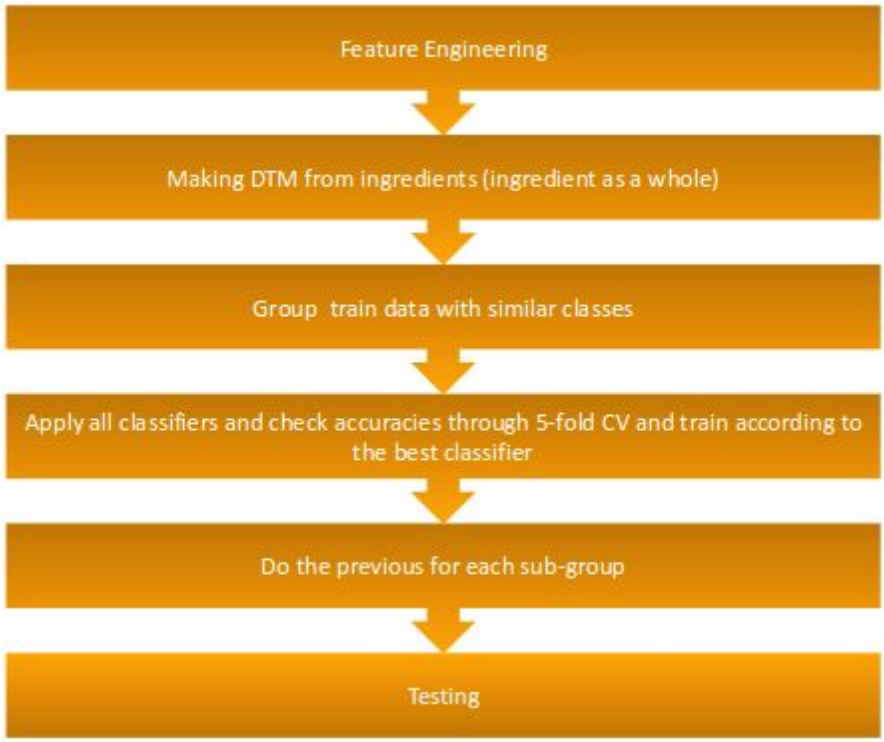

Figure 4.2.1. Model 2

Feature Engineering and DTM :

This step is exactlu same as the Model 1.

Group Similar Classes :

Gruping of the cuisine classes have been done in three ways:

1. Clustering

One way of groping the cusine was o make 4-5clusters using clustering algorithms. Clustering algorithms we used are agglomeraive clusters,GMM (Gaussian Mixture Model) and K-Means Distribution. Resulting Clusters from these algorithms are shown in Figure 2,3 and 4.

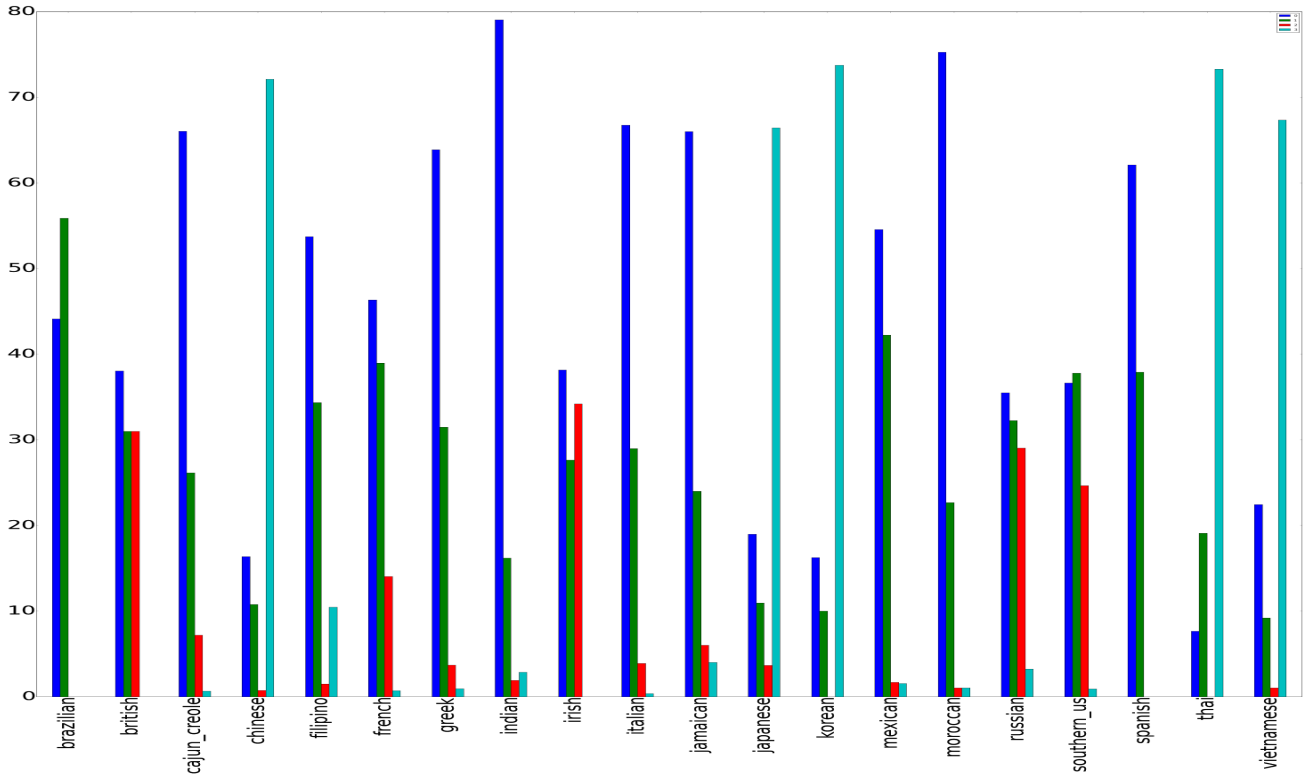

Figure 3.2.2 Agglomeartive Clustering

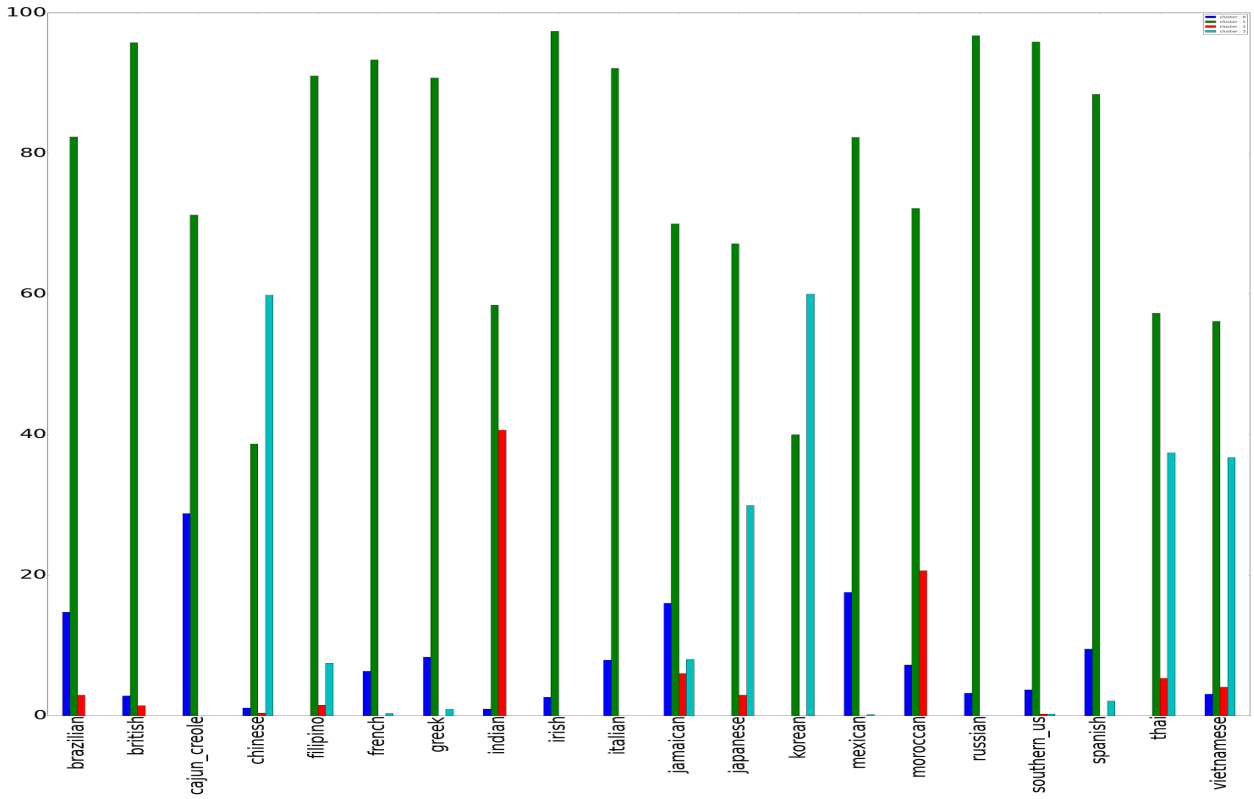

Figure 3.2.3 GMM Clustering

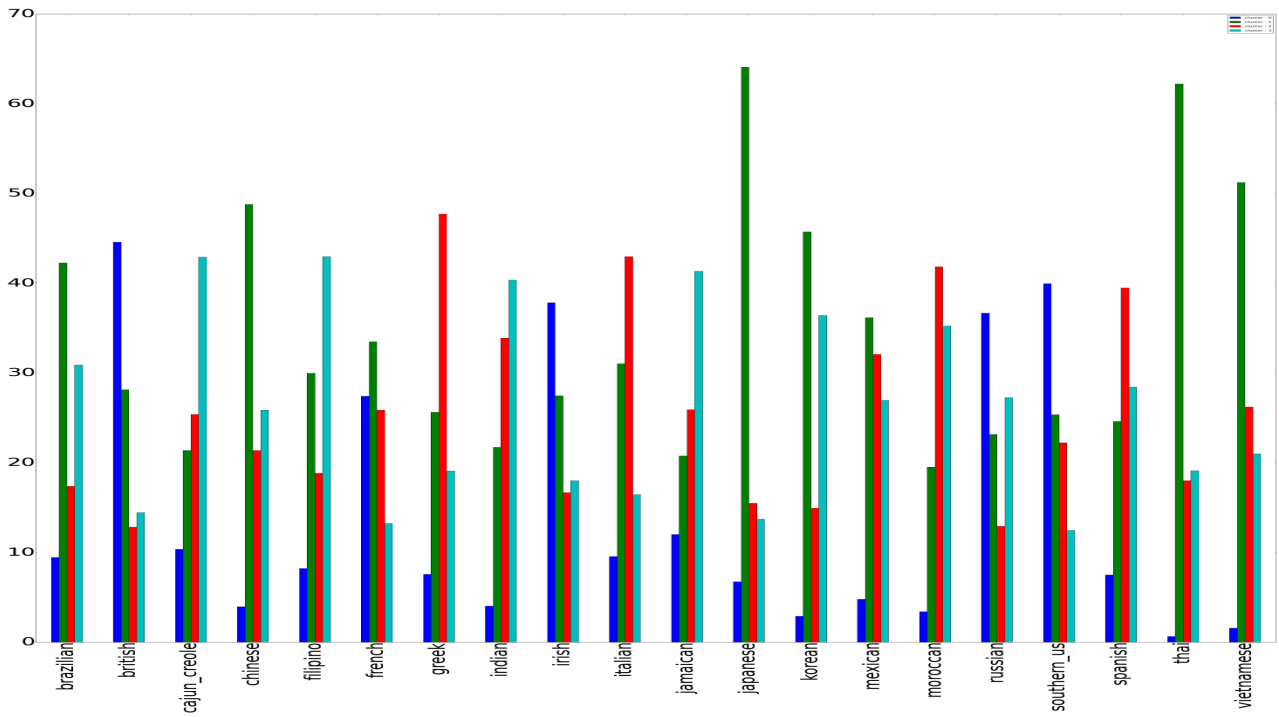

Figure 3.2.4 K-Means Clustering

2. Manual Grouping 1

As oppose to using a clustering algorithm, we also tried grouping cuisines manually by observtions.

3. Manual Grouping 2

As oppose to using a clustering algorithm, we also tried grouping cuisines manually by observtions as well as experience in test.

Accuracy and Testing :

Accuracy for classifier in each group has been done using 5-fold CV (Cross Validation). Whichever classifier is best suited is accepted for predicting on test tuples.

Results of this model on test tuples are as follows:

	1	2	3
Grouping	Clustering	Manual Grouping	Manual Grouping
Classifiers	SVM, Logistic Regression	SVM, Logistic Regression	SVM, Logistic Regression, Decision Trees, Random Forests, ADA Boost
Test Accuracies	75.08 %	75.533 %	75.503 %

Figure 4.2.5 Test Accuracy

4.3 Model 3

This model differs from Model 1 in following two ways :

- A new feature added – the first-level prediction for each tuple from the previous model
- 5-fold CV

Best classification result of this model is 76.663% usin SVM classifier.

4.4 Model 4

Model 4 is almost similar to Model 3 except that the DTM is created on individual words. For example previosuely, “choco milk” was one ingredient but now it’s divided into two ingredient “choco” and “milk”.

Best classification result of this model is 76.438% usin SVM classifier.

4.5 Model 5

Model 5 is almost similar to Model 1 except that the DTM is created on individual words.

Best classification result of this model is 76.212% usin SVM classifier.

4.6 Model 6

Model 6 is an extension of model 5. Model 6 also uses Stemming and lemmatization in data preprocessing.

Best classification result of this model is 76.031% usin SVM classifier.

4.7 Model 7

Model 7 is the simplest model. DTM is generated directly from the given list of ingrdients without any of feature engineering.

Best classification result of this model is 78.309% usin SVM classifier.

Chapter 5: Libraries

5.1. Introduction

This chapter discusses the various python libraries used to perform various tasks in the project. All the libraries used are open source.

5.2 NumPy

Numpy is a python library that provides for and enables handling of arrays and matrices, especially large ones. It also provides implementations of high-level mathematical functions for use on these arrays and matrices. In this project, the data set is often stored in NumPy arrays and computations are done on these multi-dimensional arrays.

5.3 Pandas

Pandas is another python library which provides data structures for effective handling of data. One of the data structures that pandas provides is the data frame which is often used in this project. The advantage of data frames is that they are very easy to manipulate and to operate on. The data frame is a vector of lists of equal sizes. The lists are named so in effect, the data frame provides named columns. Moreover, referencing and filtering specific rows or columns is also very easy. All of this makes feature engineering very easy.

It also provides functions for writing and reading csv files that are used in the project.

5.4 Scikit-Learn

Scikit-learn is a python library which provides support for a large number of machine learning tasks. It works with numpy arrays.

All the classifiers used in this project are classes provided by scikit-learn. Also, data preprocessing tasks like PCA, feature scaling, feature selection use scikit-learn implementations. The function used to measure accuracy in this project is also provided by scikit-learn. It also provides a pipeline to line up successive tasks. In this project, the feature selection and classification tasks are pipelined.

Also, the feature extraction tasks of dtm and tf-idf use scikit-learn classes. The classes used are CountVectorizer, HashingVectorizer, TfidfVectorizer and TfidfTransformer. CountVectorizer builds a simple dtm. TfidfVectorizer builds a tfidf matrix. HashingVectorizer also builds simple dtm, but it is especially designed for out-of-core learning.

5.5 cPickle

cPickle is a python library that provides support for serializing and deserializing objects, that is, storing objects in files and retrieving them. It has been used extensively in the project to store feature spaces and classifiers.

Chapter 6: Conclusion

The best acuracy achevied is in model 7 , which is 78.369.

The conclusion from all above models are as follow :

Model No	**Test Accuracy (in percentage)**
Model 1	76.881
Model 2	75.533
Model 3	76.663
Model 4	76.438
Model 5	77.212
Model 6	77.013
Model 7	**78.369**

Refrences

1) https://www.kaggle.com/c/whats-cooking

2) https://www.coursera.org/learn/machine-learning/home/welcome

3) http://scikit-learn.org/stable/