Class Project: Cuisine Recognition

Student Name(s): Srijita Das, William Muldoon

December 11, 2015

We look into a cuisine dataset from the Kaggle website. The problem is to identify various classes of cuisine from the ingredients that are used in various recipes across that cuisine. In this project, we do data preprocessing to clean the data, feature engineering to extract the useful and relevant features. After that, we run a couple of classification algorithms with required hyper parameter changes and do significance testing to show that some algorithms work better on this dataset than other algorithms. Finally, we choose algorithms that perform better than others and using their best parameters, we make a submission to Kaggle for the test dataset and report the results. For general training adnaccuracy reporting, we consider a part of the training set as validation set. After learning the suitable model with best parameters, we run it on the test set and upload it to Kaggle.

1 Description of dataset

The dataset provided by Kaggle was in json format and had 3 fields: recipe id, ingredients and the cuisine this recipe belongs to. The training set had 39775 instances of recipes with each recipe belonging to one of the 19 classes of cuisines. different classes of cuisines were brazilian, british, cajun_creole, chinese, filipino, french, greek, indian, irish, italian, jamaican, japanese, korean, mexican, moroccan, russian, southern_us, spanish, thai and vietnamese. The dataset was not balanced and the class distribution varied widely with the maximum being 7838 instances of italian cuisine and the minimum being 467 instances of brazilian cuisine. The class distribution graph is shown below with the bar graphs denoting each cuisine in the order mentioned above.

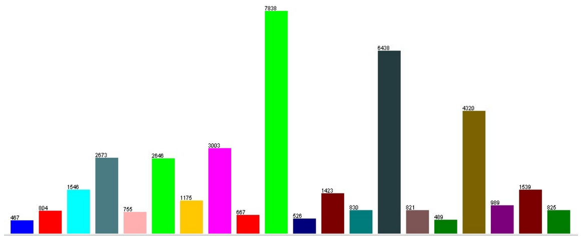


Figure 1: Class distribution of cuisines

2 Construction/Implementation

2.1 Data Preprocessing

For running Machine Learning algorithms, we had to clean the the dataset provided by Kaggle. Every recipe in the dataset consisted of recipe id and a list of ingredients. We employed some basic techniques like stem word removal, punctuation removal, quote removal from the list of ingredients so that we get a clean list of ingredients to analyze for all the recipes. This process was done in R by using the “tm” package. Also the ingredients that appeared as singular and plural values were combined together to form one single ingredient. So, on a basic level, this preprocessing of the ingredient list helped us to trim few ingredients and left us with the total number of unique ingredients as 6700. This list of ingredient is also too huge for a classification algorithm to work properly. So, next we applied a feature extraction technique to retrieve useful features (ingredients) that correctly identifies a cuisine from this set of 6700 unique ingredients.

2.2 Feature Extraction

Retrieval of useful features is a very important aspect of Machine learning as it embeds the minimum amount of useful information required to identify a particular class label. We applied an intuitive technique for extraction of useful ingredients that could uniquely identify a particular class of cuisine.We got *rid of the less meaningful feature* and *kept the most prominent features.* For that, we filtered the features based on how prominent one feature was across one class and less prominent across other classes of cuisines.

In the very first step, we retrieved the 100 most used ingredient across every class of cuisine, because these features in some sense identify that particular class of cuisine since they have been used frequently in the recipes for that cuisine.However, since the number of recipes across each class varied widely, we considered only those ingredients whose usage was more than 5% across a class of cuisines. This approach helped us to normalize the usage of ingredients across a particular class. Our observation was that the ingredient that was among the list of 100 most used ingredients for a class of cuisine was actually used 30% of times in that class of cuisines. Hence, removal of ingredients whose usage was less than 5% across a class helped us get rid of all those ingredients that were actually used quite less for that class but came up in the list of 100 most used ingredients because the number of instances for that class of cuisine was small as compared to some other class of cuisine. This step of feature extraction left us with all the features that were prominent across a particular class of cuisine.

In the second step of the feature extraction process, we removed all those ingredients which were used more often across all the classes. Our assumption is that features which are most likely to occur frequently across all class labels does not add any meaningful value. We saw that common ingredients like sugar, salt and water were there among the most used ingredients across all cuisines and hence, these features were redundant. In this step of our feature extraction process, we removed all such features whose usage was less than 5%(first criteria of our feature extraction process) in the class and which occured in more than 75% of the cuisines. This left us with 243 features on which we ran the learning algorithms.

Our approach of feature engineering was based on Entropy or Info-gain based methods where we exclude features that do not cause significant drop in Cross-entropy between the class and the feature. It is more time efficient than correlation based methods for feature reduction, however it may not be able to filter out as many features as Correlation based methods might do.

2.3 Feature vector creation

With the new reduces feature set, we created a feature vector for each recipe where a value of 1 indicated the presence of that ingredient in the particular recipe and a value of 0 indicated absence of the feature for that recipe. We now train the model using such feature vectors for each of the recipes. The training set thus formed constituted of a sparse matrix with most of the features as 0 since we filtered out 243 features to work with from a set of 6700 unique features.

3 Description of methods

We tried 5 classification algorithms on this dataset for a random sample of 25000 training points and 10000 test points. We will discuss in details the accuracy of these algorithms. From these 5 algorithms, we picked the best 3 and tried doing statistitcal tests to choose the best parameter and create the best model using cross validation strategies. We will discuss the results of these algorithms one by one in the following subsections.

3.1 K-nearest neighbour

k-nearest neighbour is one of the most simple algorithms for multiclass classification where for a new data point, the number of neighbours close to a data point is evaluated and this test point is then classified according to the majority of the class of it's neighbours. We used Sklearn's ready-made package to run this algorithm. The parameter k which indicates the number of neighbours in this algorithm is variable and we tried running the algorithm for various values of k and the accuracy for this algorithm on our training data is as jotted below:

k	Accuracy
3	55.6
6	54.07
10	53.52
15	52.0
20	52.06

This algorithm on an average gives an accuracy of 55%.One thing we noticed about this algorithm as is evident from the results is that as we increase the number of neighbours, the accuracy of this algorithm drops. One suspicion is that this might be due to the highly imbalanced data in the training dataset. As the dataset contains more instances of a particular claass, so as we increase the value of k, it is highly likely that the class which have more instances will dominate the model and hence, more test instances will get classified into the dominating class. So, k nearest neighbour is not a good ideaa for imbalanced datset. Also, we tried running this algorithm taking 300 samples from each class but it's accuracy did not approve much. This is because, 300 samples of each class is too less a number to train the data point, especially when we have filtered out a lot of features. So, our overall finding is that k-nearest neighbour is not a suitable algorithm for this particular dataset. The running time of this algorithm is also extremely slow.

3.2 Naive Bayes

William's writing

3.3 Multinomial Logistic Regression

We also ran the multinomial Logistic Regression model with softmax activation. We used Sklearn's Logistic Regression package with lbfgs solver for gradient descent. We used L2 regularisation and the package internally implements ove vs rest classifier where a binary classifier is built for each of the classes classifying the data points into positive and negative. At the end, an unseen datapoint is classified to the class label for which one of the classifiers have the highest probability. Logistic Regression was also ran on a random training sample of 25000 and a test sample of 10000. The accuracy of varying Regularisation parameter on Logistic Regression is as below:

Regularisation	Accuracy
0.001	66.8
0.01	65.64
0.1	65.92
1	65.11
10	64.43
100	56.87

The accuracy of Multinomial Logistic Regression is much better than k-nearest neighbour. We noticed that the weights were becoming large and so, regularisation had a significant impact on this algorithm. We tried running this classifier for various values of regularisation and could increase the accuracy of this classifier to an extent of 63%. THe results of Cross validation with this algorithm is jotted down in the later sections. It seems from the data above that a regularisation parameter of 0.001 gives the best accuracy for this classifier.This classifier seems to have a much lesser running time than K-nearest neighbour.

3.4 Neural Network

We even tried out Neural network on this training set by running the algorithm in "Weka" but did not get significantly good results. The algorithm was taking long time to run. Since, there were 243 hidden features, we tested a Neural Network model with 300 hidden nodes, 20 output nodes for multiclass classification, learning rate as 0.2, momentum as 0.20 and number of repetitions over the sample as 10. We could only get an accuracy of 57.39% with Neural network.

We think that for a domain based dataset like this, Neural network is not a very good choice. Our feature set selection strategy may not be that appropriate and it is pretty possible that Neural network performed pretty poorly because of the sparse matrix generated by our feature set. For a more systematic problem, where the features ares more general and contains lesser noise, Neural network may be one of the best classification algorithms to work with. This is our intuitive opinion and we are not really sure why Neural network did not perform well on this dataset.

3.5 Random Forest Classifier

Random Forest classifier is an ensemble learning technique where multiple weak decision trees are created and then an unseen data point is classified as majority votes from these several weak classifiers. The decision trees picks a subset of features from the feature set having the most information gain and then splits according to values in these features. We used Sklearn's Random forest Classifier to implement Random Forest. Random Forest has many parameters like the number of ensemble used, the depth of the tree, number of leaf nodes used etc. However, since we were already working with a reduced feature set, we did not cosider the other parameters important. We mainly tested Random Classifier with different number of ensemble as parameter and jotted below are the Random Forest Classifier accuracy for various values of the number of classifiers used.

No of ensemble	Accuracy
10	73.47
20	74.54
30	74.94
40	74.63
50	76.2

Till now, among all the classifiers, Random Forest is the one giving us one of the highest accuracy. We vary the number of ensemble used for majority voting and see that when we set the number of ensemble to 50, the accuracy goes as high as 76.2 %. These results are though not reported after cross validation and this accuracy might go down a little bit when we use 10-Fold cross validation.
Among all the classifiers used for this project, Random Classifier is the one giving quite a high accuracy. Random forest creates several weak classifiers and this might be the reason why this classifier does good. Our problem is from cooking domain and an efficient feature extraction for tis problem would require specific domain knowledge. Domain knowledge about this problem will help us to merge several ingredients as one and remove several ingredients. We do not have any domain knowledge and hence, our feature extraction process is not a very strong one. In such case, classifying the entire data using one classifer might result in building a weak classifier. But, If we take the votes of several weak classifier, it might lead to developing a strong classifier. I think, because of the above mentioned reasons, Random Forest works best for our problem domain. A Random forest also selectively chooses features at every tree node based on the feature that gives maximum information gain.So, that way for a weak feature selection model, Random Forest gives good predictions.

4 Results

We compared Naive Bayes with multinomial distribution, multinomial Logistic Regression and Random Forest Classification algorithms.Our experimentation setup is as described below. We split the training set into random sample of 20000 and 10000 respectively. The randoom sample of 10000 poits was our test set and we just ran the model learnt on the training and validation sets on this testset. We did not use this test set for any other purpose. We performed 10 fold cross validation on the training set for each of the parameters of these Classification algorithms.For Multinomial Logistic Regression, we varied the regularisation parameter from 0.0001 to 10000 and for Random Forest Classifier, we varied the number of ensemble from 5 to 50. 10- fold cross validation was done to see the accuracy for these algorithms for each of these parameters and then the parameter of the classification algorithm was chosen which gave the best accuracy. Now, using this parameter, the model was run on the entire training set and thw weughts were learnt. Using this new model with the best parameters,we then ran it on the test set and took the accuracy readings. We repeated this entire process for 30 times so that we have enough data to do significance testing between these three algorithms to to compare their performance.The results of t-test between these three algorithms are as below:

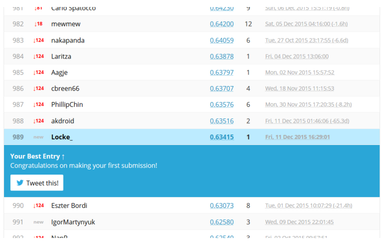
```
anova(lm(Accuracy~Method,data=data))
Analysis of Variance Table

Response: Accuracy
```

```
          Df  Sum Sq Mean Sq F value    Pr(>F)
Method     2 1389.90  694.95  3843.7 < 2.2e-16 ***
Residuals 87   15.73    0.18
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
####################Pairwise t-tests############################

t.test(RF.acc$accuracy,Log.acc$accuracy,alternate="greater")
Welch Two Sample t-test

data:  RF.acc$accuracy and Log.acc$accuracy
t = 56.4068, df = 57.971, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 6.713971 7.208029
sample estimates:
mean of x mean of y
 72.88867  65.92767

t.test(Bayes.acc$accuracy,Log.acc$accuracy,alternate="greater")

Welch Two Sample t-test

data:  Bayes.acc$accuracy and Log.acc$accuracy
t = -22.4466, df = 48.374, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.481361 -2.073450
sample estimates:
mean of x mean of y
 63.65026  65.92767

 t.test(RF.acc$accuracy,Bayes.acc$accuracy,alternate="greater")

Welch Two Sample t-test

data:  RF.acc$accuracy and Bayes.acc$accuracy
t = 89.5738, df = 47.726, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 9.031003 9.445807
sample estimates:
mean of x mean of y
 72.88867  63.65026
```

We did a pairwise t-test between these three pair of algorithms. From the above pairwise comparison of t-tests we see that Random Forest's accuracy is significantly better than Naive Bayes(p-value:2.2e-16,CI:[9.031003,9.445807]) with the confidence interval being pretty thin. Also, Naive Bayes performs better than Logistic Regression(p-value: 2.2e-16,CI=[-2.481361,-2.073450]). Similarly, Random Forest performs statistically better than Logistic Bayes.(p-value:2.2e-16,CI=[6.713971,7.208029]). From these statistical tests, it is clear that Random Forest and Naive Bayes performs better than multinomial Logistic Regression and so, we tried these two algorithms for the Kaggle competition.

Below are the screenshots of our Kaggle submission:

We build the model by Multinomial Naive Bayes algorithm and below is the screenshot of the evaluation by Kaggle. (William: put in the accuracy of Random Forest on the small feature set).We achieved an accuracy of 63.41%.



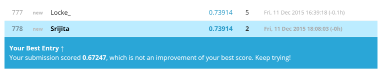
981	↓81	Carlo Spalocco	0.64230	9	Sun, 06 Dec 2015 15:51:19 (-0.8h)
982	↓18	mewmew	0.64200	12	Sat, 05 Dec 2015 04:16:00 (-1.6h)
983	↓124	nakapanda	0.64059	6	Tue, 27 Oct 2015 23:17:55 (-6.6d)
984	↓124	Laritza	0.63878	1	Fri, 04 Dec 2015 13:06:00
985	↓124	Aagje	0.63797	1	Mon, 02 Nov 2015 15:57:52
986	↓124	cbreen66	0.63707	4	Wed, 18 Nov 2015 11:15:53
987	↓124	PhillipChin	0.63576	6	Mon, 30 Nov 2015 17:20:35 (-8.2h)
988	↓124	akdroid	0.63516	2	Fri, 11 Dec 2015 01:46:06 (-65.3d)
989	new	**Locke_**	**0.63415**	**1**	**Fri, 11 Dec 2015 16:29:01**

Your Best Entry ↑
Congratulations on making your first submission!
Tweet this!

990	↓124	Eszter Bordi	0.63073	8	Tue, 01 Dec 2015 10:07:29 (-21.4h)
991	new	IgorMartynyuk	0.62580	3	Wed, 09 Dec 2015 22:01:45

Figure 2: Multinomial Naive Bayes on 243 features training data

Another observation that we saw is that when we ran Random Forest on the entire training set with all features(6700) we achieved an accuracy of 73% as evaluated by Kaggle. This is valid because all the features will contain the entire data without any useful features being thrown out. But in real life scenario, it is difficult to work with such a huge training set containing so many features and so feature extraction is important. Without proper domain knowledge, we might not have been very accurate in our feature extraction strategy.



777	new	Locke_	0.73914	5	Fri, 11 Dec 2015 16:39:18 (-0.1h)
778	new	**Srijita**	**0.73914**	**2**	**Fri, 11 Dec 2015 18:08:03 (-0h)**

Your Best Entry ↑
Your submission scored **0.67247**, which is not an improvement of your best score. Keep trying!

Figure 3: Random Forest on entire training data with 6700 features

5 Conclusion and discussion

Summarize the overall findings. Discuss some next steps that your findings have uncovered. In particular, pose a scientific question/hypothesis that could be explored next, now that you have an idea of what algorithm(s) can perform well on this dataset.