CSE 255 Assignment 2
Cuisine Prediction/Classification based on ingredients

Rishabh Singh Verma (rsverma@ucsd.edu, A53103069)
Himanshu Arora (hiarora@eng.ucsd.edu, A53097039)
University of California, San Diego

Abstract:

In this paper, we consider different strategies for identifying the cuisine, given its ingredients. This project aims to explore what combination of ingredients is helpful in identifying a cuisine if the recipe is not given. This has been tackled as a problem of cuisine classification. We also explore different classification algorithms in tandem with approaches like taking combination of multiple ingredients for an exhaustive analysis of the results obtained.

Keywords:

Yummly, Cuisine, Ingredients, Classification

Introduction:

All over the world, food recipes vary a lot even if they have the same ingredients. And different recipes belong to different cuisines. So if someone is only given the ingredients, estimating a recipe is a problem that can be solved by looking at prior data (if there is sufficient data) but estimating a cuisine type (which is a superset of recipes, considering broader approach and wide variance in the use of ingredients) is not as easy. And what makes it even more interesting is, there could be many more versions of the same recipe [1]. However, we have followed the "standard cookbook" recipe approach for now, not considering all the variants at the same time.

Dataset:

The dataset for these recipes has been obtained from a Kaggle Competition [2] "What's cooking?" hosted by Yummly. It is a popular website and application which provides recipe recommendations tailored to the individual's preferences, semantic recipe search and a digital recipe box. It contains 39774 instances of cuisine and ingredient list pairings, in which each cuisine has several ingredients. There are a total of 2965 distinct ingredients in the entire training dataset for a total of 20 cuisines. A sample from the training set typically looks like

["id": 1, "cuisine": "greek", "ingredients": ["romaine lettuce", "black olives", "grape tomatoes"]]

The interpretation of the above mentioned example is obvious - for some arbitrary greek recipe, the ingredients used were romaine lettuce, black olives and grape tomatoes. An additional test set of 9944 instances has been provided with the training data in which cuisine value is missing and the task at hand for the competition was to predict the cuisine for each example.

["id": 234, "ingredients": ["Cheese", "Tomato", "Karela"]]

The distribution of the ingredients and cuisines is presented in the following graphs:

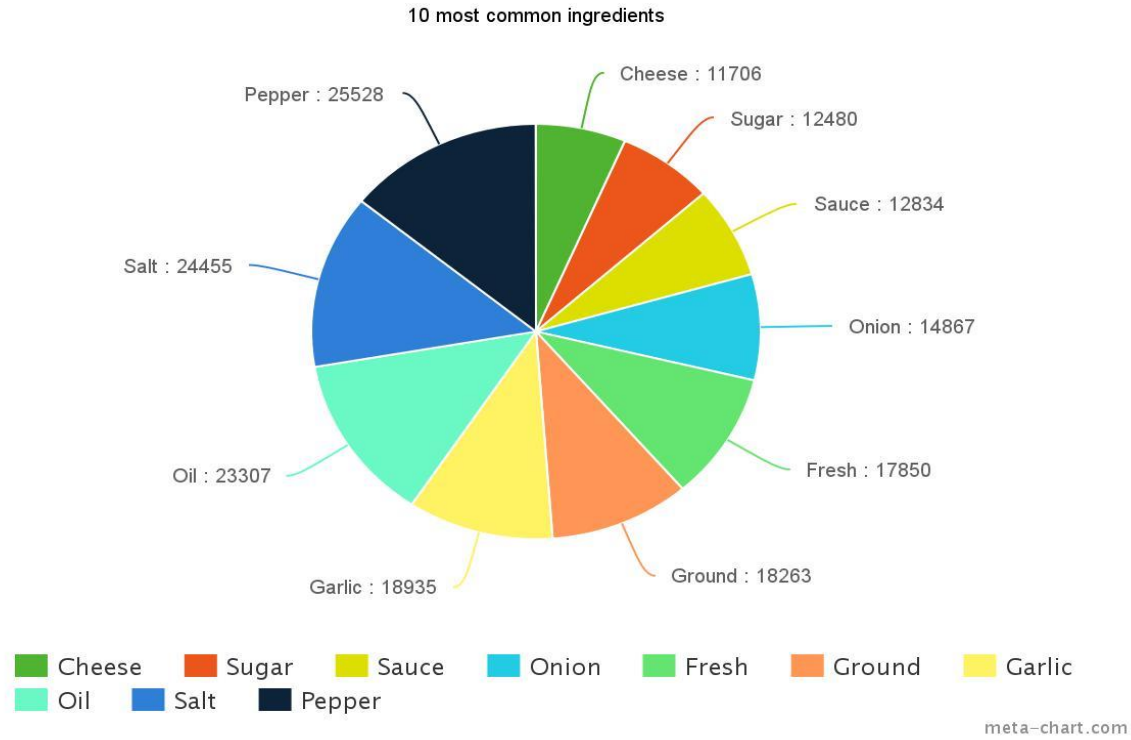


Figure 1: Counts of most common ingredients in training set data

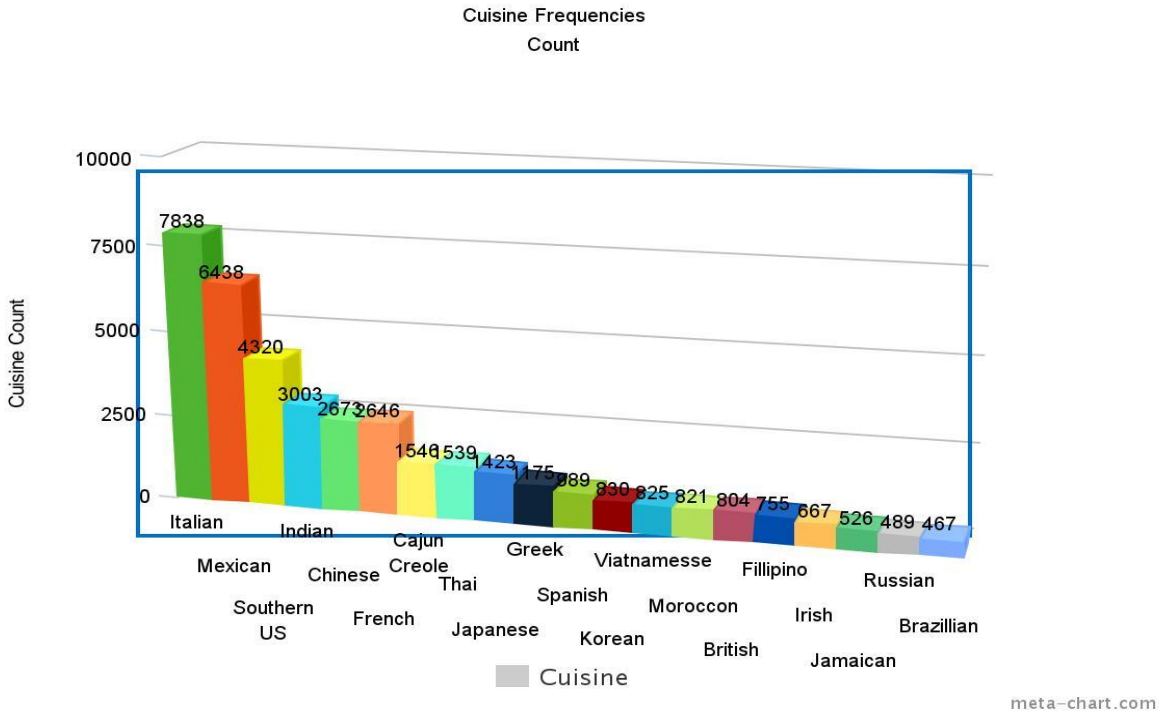


Figure 2: Counts of all cuisines in training set data

We observe that Italian dishes dominate the charts. Therefore, we already have the baseline in which we predict Italian cuisine all the time.

Exploration of dataset and intuition:

After observing the dataset, we pondered upon the mining techniques that could be applied to this dataset. We explored the possibilities of application of regression, dimensionality reduction and found that only classification could be applied to this dataset. Since the data is limited for estimating cuisines based only on ingredients, we used some prior knowledge about recipes. E.g. an instance containing flour, butter and sugar would have a high probability of having eggs in it. This not only presents a pattern with respect to ingredient duplets and triplets but also opens up a lot of possibilities for exploration despite the size limitation of the dataset. Therefore, bigrams and trigrams (along with unigrams) of ingredients have been taken for classifying the cuisines.
While exploring the data, we could also consider a pattern amongst ingredients if we look at cooking techniques. E.g. for a dessert, there is almost always a requirement of sugar and flour for a particular cuisine. Due to limitation of time, we plan to leverage this factor in future.

Preprocessing:

Since the data was taken from a Kaggle Competition, not much cleaning was required. However, the problem of multi-class text classification required cleaning and adjustment of data according to our needs and we worked on a general framework for achieving that. Input and output of data was done through Pandas library. All the text was converted into lowercase. Initial approaches involved the use of NLTK library and stemming the given words but it did not affect the dataset much. So this part was done only for the ingredient lists present in the dataset and not for cuisines as it would take up more time and not be much useful as there were only 20 unique cuisines.

There was another interesting aspect of the problem – some ingredients have more than one words in their designation like ‘star anise’, ‘garam masala’, ‘cinnamon powder’ etc. Our standard data cleaning approaches posed a problem with respect to this as star anise was further reduced to two ingredients namely “star” and “anise”. This problem is also evident in figure 1 where ingredients like “fresh” and “ground” are present within the list of top 10 ingredients – they could have been a part of multiple ingredients like “ground pepper”, “ground black pepper”, “ground cumin” and others for the word “ground” & “fresh lemon juice”, “fresh cilantro”, “chopped fresh chives” for the word “fresh”. Since this posed a problem of multiple counting while providing an unfair bias to some instances and also increased chances of weird terminology showing up in results, we decided to deal with this by considering bigram, trigrams and quadgrams along with unigrams for our approaches so that our classification accuracy could improve.

Lastly, 100 out of these cuisine-ingredients’ pairings were duplicate which meant that there were a total of 39674 unique examples given in the training set.

Chasing the General Framework through experimentation:

We started with the naïve approach of establishing the baseline by predicting Italian all the time as it was the most common cuisine. After that, we considered some more reasonable approaches. Since the data given was all text, some kind of processing was needed to represent the data in a form which could be easily handled by text classification algorithms. Moreover, ingredient features needed to be extracted from text as many ingredients were common among the cuisines. This initial approach lead us to the solution of count vectorization of ingredients, as count of an ingredient in a recipe can be considered as a feature in feature vector. Since an ingredient is either present or absent, the count vector was simply a binary vector. So, either an ingredient tuple for the entire vector would be "1" or "0".

Cuisine: Indian

Ingredients	.....	Paneer	Pepper	Potato	Parmesan	Pomegranate	.....
Counts	.....	1	1	0	0	0	.....

Later, we tried a different; better approach of term frequency (TF) and document frequency (DF) computation. Considering each training data as an instance, we check whether an ingredient is present in that instance to get the term frequency. Since the dataset contains only the name of ingredient, term frequency for a given instance would again either be 0 or 1.

In the case of document frequency, we calculated the frequencies for all ingredients to find the ones which had highest frequency among all the samples i.e. those which were most common amongst the ingredients.

To iron out the dominating effect of a single ingredient due to its ubiquitous nature, we used the following formula

$$\begin{aligned} \text{idf} &= -\log P(t|d) \\ &= \log \frac{1}{P(t|d)} \\ &= \log \frac{N}{|\{d \in D : t \in d\}|} \end{aligned}$$

Here, d represents a single document and D represents the set of all documents.

Inverse document frequencies (idf) for all ingredients in each instance were calculated. E.g. Pepper is a very common ingredient which was used in 25528 out of 39674 instances. So after tfidf application, the bias towards pepper and other common ingredients would be reduced. Through this process, we eliminated the ingredients having document frequencies above a certain threshold. Doing this experimentation of threshold could lead to overfitting, so we kept the max document frequency to 0.35.

After that, we applied stemming to ingredients and then again tried the above two approaches to get better results. Now we had two different vectors to play with. They could be tweaked

using different parameters and therefore, it became impossible to run every iteration to compare the best tuned settings. So we finalized upon the use of a pipeline for application of these techniques – stemming, vectorization, tfidf and then the classification algorithm with different attributes and parameters. Since each algorithm took time to converge, instead of changing the parameters every time, we inserted a range of parameters into the pipeline and then trained the classifier using GridSearchCV method. Grid search is a multi-core algorithm which gives the optimum parameters using the entire range of inputs that have been supplied to the algorithm.

Finally, we thought of combining classifiers i.e. using ensemble algorithms (combinations of different classifiers) for achieving better accuracies. We applied Random Forest Classifier and Voting classifier with Multinomial Gaussian Bayes, Passive Aggressive Classifier and Logistic Classifier with soft voting criteria. The latter turned out to be computationally intensive and crashed mid-way. The former gave a not so good accuracy and finally was rejected.

In the end, since term frequency counts could only be 1 or 0, we replaced the count vectorization process followed by tfidf vectorization by a singular TfIdf vectorization which gave a minor boost to the general framework.

Algorithms Explored:

Baseline Approach – This approach worked by predicted the cuisine that was the most common i.e. Italian. Although this was not very effective, this helped us set up the general framework defined above. Accuracy – 0.19268

For all the following approaches, we used GridSearchCV function which does a 5 fold cross validation on the training data to select the best parameters. We tried to define the most appropriate parameter values by searching websites like StackExchange [3] and reading description of parameters on the SKLearn documentation. We have summarised the best chosen parameters reported by GridSearchCV in the table 1.

Multinomial Naïve Bayes (MNB) [4] – Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. It follows that ingredients in an instance were independent of each other. But this is a wrong assumption to make, as we know from prior knowledge that ingredients for a given cuisine are highly correlated. Still, this performs better than predicting most probable cuisine all the time. We tried MNB with both CountVectorizer and TFIDFVectorizer, and as expected, TFIDFVectorizer performed better.

Multinomial Logistic Regression [5] - Logistic Regression is originally a binary classifier. In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is availed. We use the lbfgs solver for optimizing our solution.

Random Forest [6] - A random forest is a meta-estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. We used no. of estimators as 600.

Perceptron [7] - Originally a binary classifier, the perceptron generalizes naturally to multiclass classification. Here, the input x and the output y are drawn from arbitrary sets. A feature representation function f(x, y) maps each possible input/output pair to a finite-dimensional real-valued feature vector. The feature vector is multiplied by a weight vector w, and the resulting score is used to choose among many possible outputs.

Passive Aggressive [8] – The intuition behind this classifier is-
Passive: if correct classification, keep the model;
Aggressive: if incorrect classification, update to adjust to this misclassified example.
In passive, the information hidden in the example is not enough for updating; in aggressive, the information shows that at least this time you are wrong, a better model should modify this mistake.

SVC [9] – Like Logistic Regression, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. We have used linear kernel in our analysis.

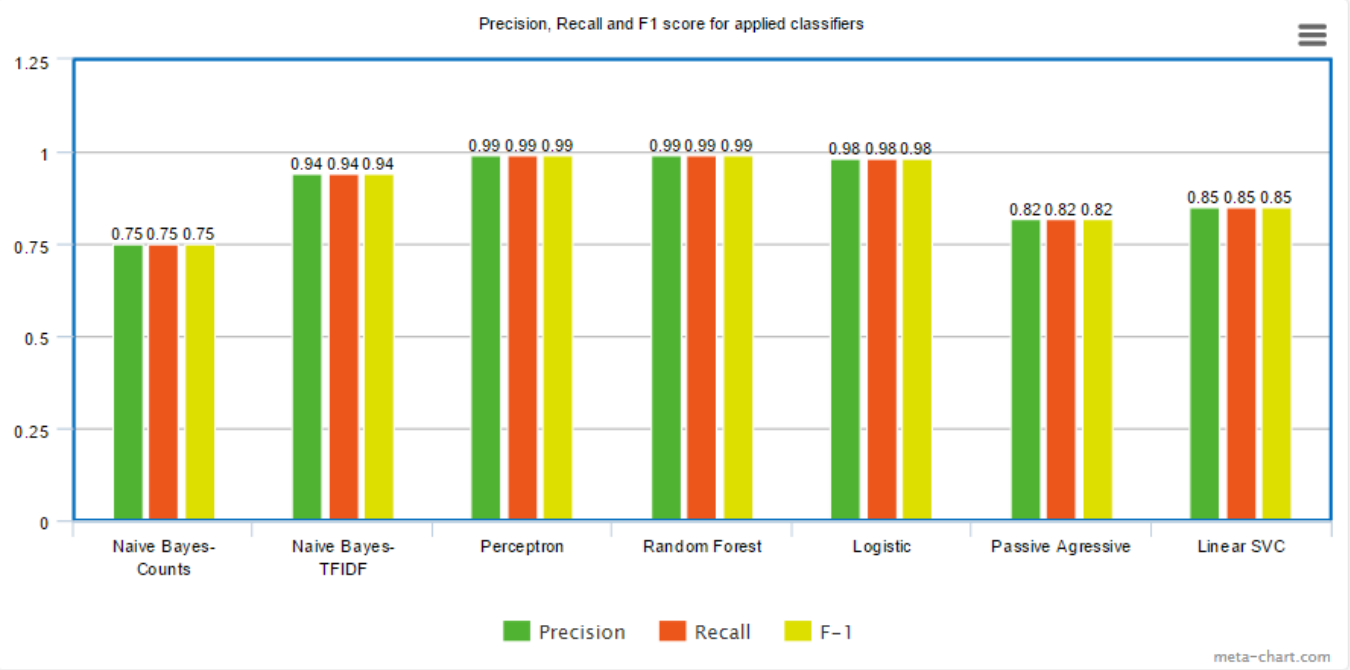
Classifier	Ngram-model	Alpha	C	Warm_start
Naive Bayes – Count Vectorizer	Unigram	0.01	-	-
Naive Bayes – TF-IDF Vectorizer	Bigram	0.01	-	-
Logistic	Bigram	-	10	-
Random Forest	Bigram	-	-	True
Perceptron	Bigram	0.0001	-	True
Passive Aggressive	Unigram	-	0.1	True
Linear SVC (linear kernel)	Unigram	-	0.9	-

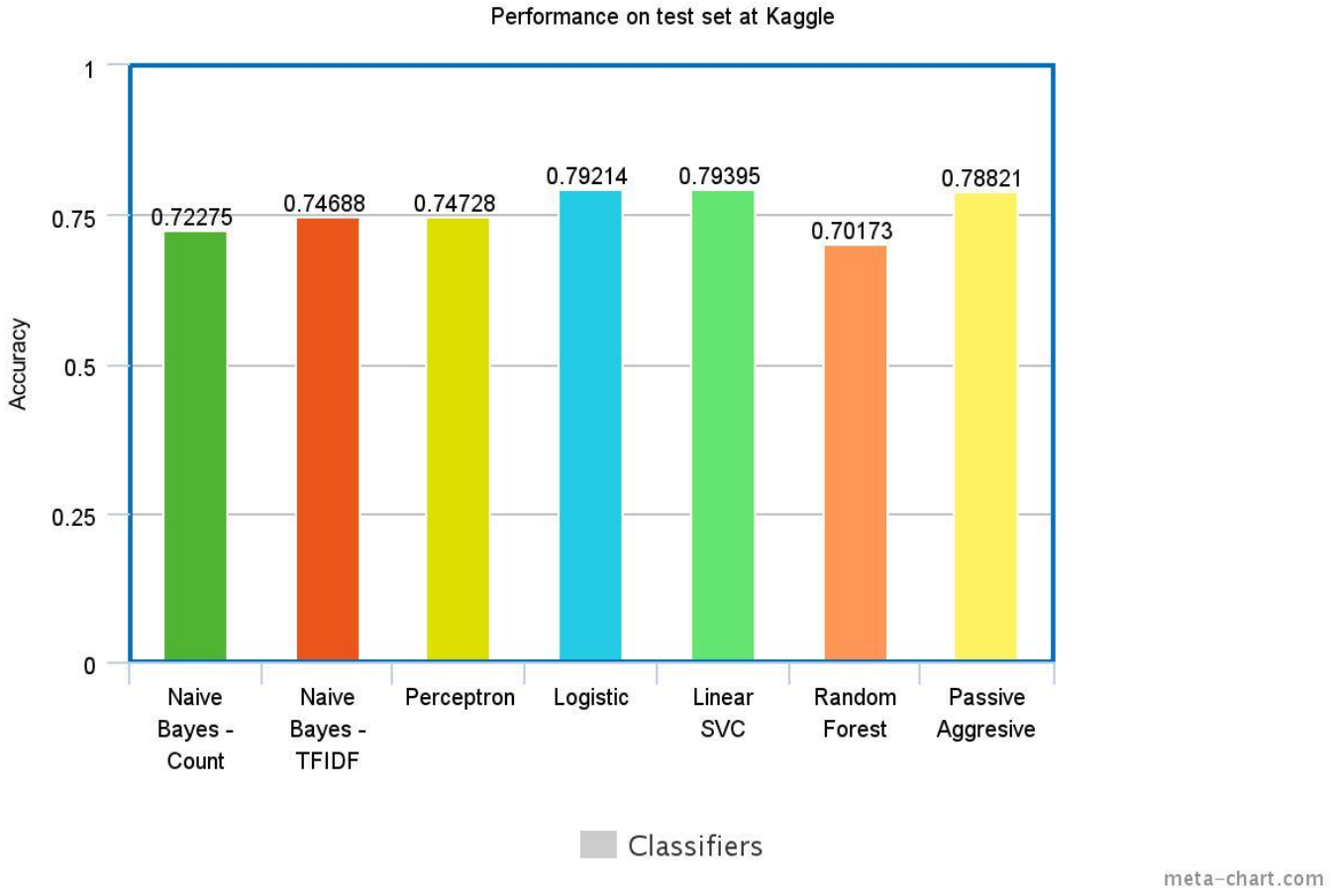
Table 1: Best performing parameters for classifiers using Grid Search

Conclusion:

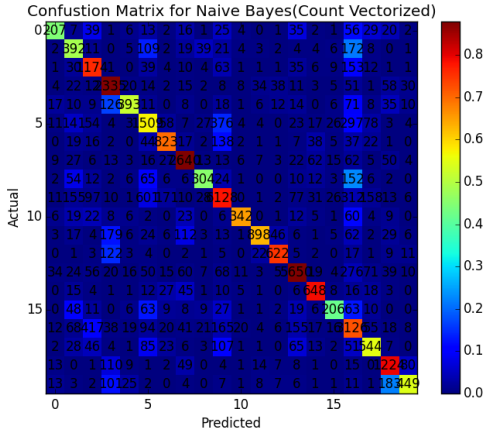
From the plots below, we can infer that LinearSVC and Logistic classifiers have similar performance. This was expected as both of them linear in nature. They also outperform most other classifiers by a small margin. Random forests, which is a part of ensemble classifiers performs relatively worse than other classifiers. A similar trend was noted for Adaboost classifier in which only 46.3% accuracy was achieved but only for n_estimators parameter as 600. For higher values of n_estimators, the computation was too intensive to continue processing further (which is the reason why it has not been mentioned here). In general, for our dataset, it was observed that ensemble algorithms (Random Forests, Adaboost and Voting Classifier) performed rather poorly than was expected. We have still not been able to deduce the reason for the same.

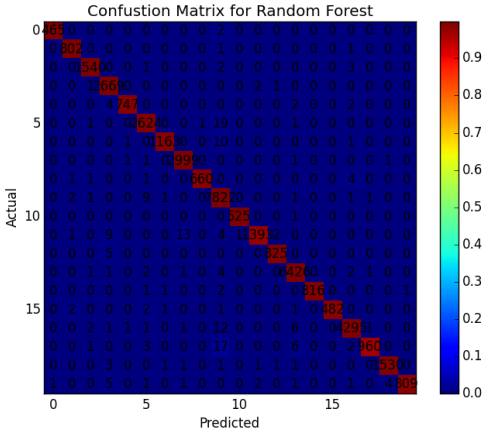
BEST CLASSIFIER – LinearSVC with 79.395% accuracy on the test set.

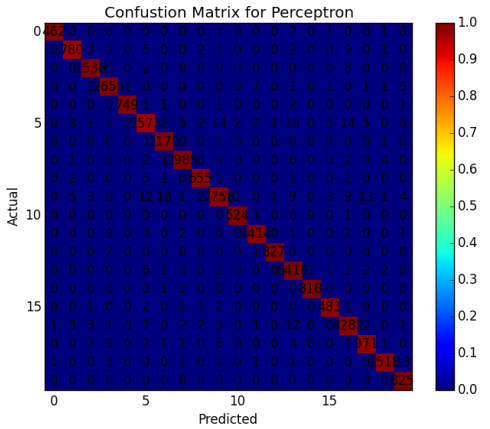

Precision, Recall and F1 score for applied classifiers
1.25
1
0.75
0.5
0.25
0
0.75 0.75 0.75
0.94 0.94 0.94
0.99 0.99 0.99
0.99 0.99 0.99
0.98 0.98 0.98
0.82 0.82 0.82
0.85 0.85 0.85
Naive Bayes-Counts
Naive Bayes-TFIDF
Perceptron
Random Forest
Logistic
Passive Agressive
Linear SVC
Precision
Recall
F-1
meta-chart.com

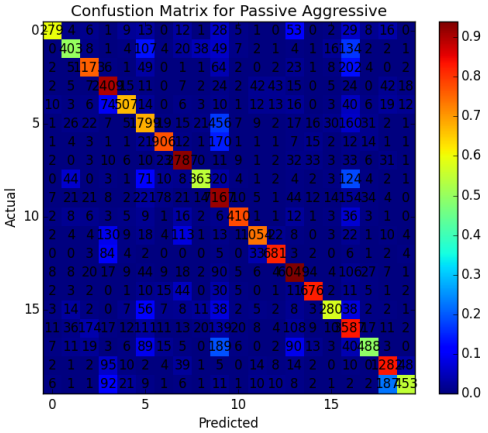

Performance on test set at Kaggle
Accuracy
1
0.75
0.5
0.25
0
0.72275
0.74688
0.74728
0.79214
0.79395
0.70173
0.78821
Naive Bayes - Count
Naive Bayes - TFIDF
Perceptron
Logistic
Linear SVC
Random Forest
Passive Aggresive
Classifiers
meta-chart.com

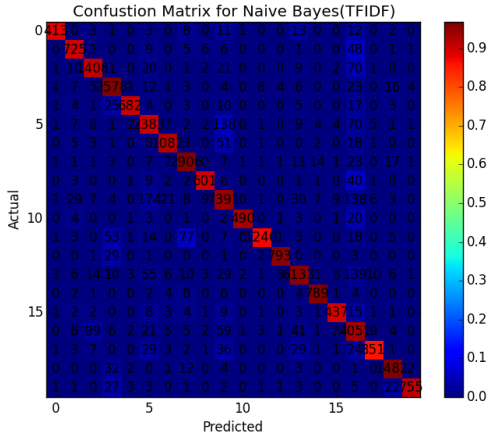
Confusion Matrices for different classifiers

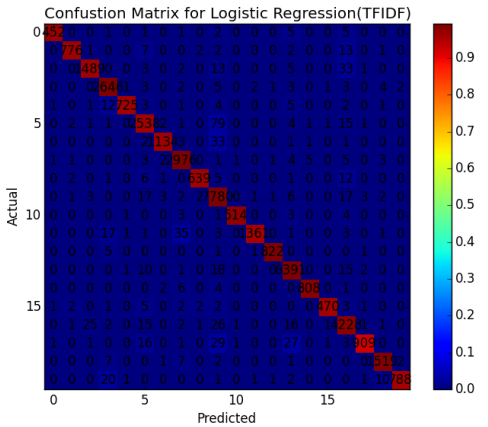












Related Work:

Similar work has been done by Su, Lin, Li and others [11] for automatic cuisine classification using ingredients. The difference between our and their dataset is, our dataset has limited features and was taken from Yummly.com whereas their dataset had many more features and was taken from Food.com. Additionally, they have also used associative classification for their analysis and have used SVD along with SVM for dimensionality reduction before classification. But the biggest difference is that their dataset contained recipe names along with ingredients and cuisines whereas we only had ingredients and cuisine for training which makes our task more challenging and interesting.

Future work:

If we had more time, we would leverage the prior knowledge of cooking techniques and apply it to various ingredients as cooking techniques help in higher correlation between cuisines. This would help with finding out inter-cuisine correlations along with intra-cuisine correlations for the same set of ingredients. E.g. If a given set of ingredients is eggs, sugar, butter, flour and chocolate chips, there is a high probability that the cooking technique would be baking as these ingredients in combination are primarily used in baking. We could use a lower dimensional representation in this case (lower than the number of different ingredients) and use that to classify the test data into cuisines. A similar possibility exists for dish type – if we already know that a given combination of ingredients say lemon, salt, orange, chicken, vegetable stock etc. is savory, we could further narrow down the cuisines to the ones which have a higher number of savory tuples rather than dessert or spicy. We would also try exploring the different variants of the same recipe with a different dataset.

Lastly, we would also work towards application of boosting/ensemble algorithms for classification of these cuisine-ingredient tuples.

Kaggle Usernames for verification of the claims made in this paper

Rishabh Singh Verma - https://www.kaggle.com/rsverma/results
Himanshu Arora - https://www.kaggle.com/hiarora/results

Acknowledgement

We would like to thank Prof McAuley for his guidance and suggestions, especially for giving us the idea of ingredient combinations.

References:

1. References from Dan Jurafsky, Stanford - LINGUIST 62N The Language of Food
2. Yummly-Kaggle. 2015. Kaggle – What's Cooking? (2015). Retrieved November 28, 2015 from https://www.kaggle.com/c/whats-cooking
3. Stack exchange – StackOverflow, Cross Validated

4. McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1998.
5. Böhning, Dankmar. "Multinomial logistic regression algorithm." Annals of the Institute of Statistical Mathematics 44.1 (1992): 197-200.APA
6. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
7. Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer.s
8. Crammer, Koby, et al. "Online passive-aggressive algorithms." *The Journal of Machine Learning Research* 7 (2006): 551-585.
9. Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and computing, 14(3), 199-222.
10. Stack exchange – StackOverflow, Cross Validated
11. Han Su, Ting-Wei Lin, Cheng-Te Li, Man-Kwan Shan, and Janet Chang. 2014. Automatic recipe cuisine classification by ingredients. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp '14 Adjunct). ACM, New York, NY, USA, 565-570. DOI=http://dx.doi.org/10.1145/2638728.2641335
12. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.