Predicting Cuisines Using Machine Learning

Project Group 41:
Zeinep Sonkaya (2657241)
Vanessa Karlqvist (2657384)
Connor Hope (2642997)
Alex Boyko (2593301)
Taewon Hwang (2656996)

March 29, 2019

Abstract

The purpose of this paper is find out if *feature selection and oversampling will result in enhanced performance overall for different models in the case of data with class-imbalance.* To test this hypothesis we search for the best model for prediction using the processed data and compare the accuracy results from the raw data. We use a dataset of about 40,000 recipes, respectively matched with 22 different cuisines. Results show that the logistic regression model is the most accurate model with an accuracy of 0.7029. Subsequent results from testing on raw data show that balancing and feature selection did not improve the accuracy meaning that the hypothesis was disproved.

1 Introduction

In this report, various machine learning techniques are used to predict the cuisine that a dish belongs to, given the ingredients it contains. The dataset used - which contains both the cuisine and the list of ingredients for each respective dish, was obtained from kaggle.com's "What's Cooking" competition. Research paper written earlier our dataset has suggested the possibility of superior performance by cleaning and balancing the data. Thus the data is cleaned and processed in a variety of ways in order to increase the effectiveness of any models used. Furthermore, feature selection techniques will also be employed. To prove feature selection and oversampling will result in classifier's optimal performance where different classifiers are tested with data that has class-imbalance.This hypothesis is tested using five different methods : K-Nearest-Neighbors, Support Vector Machine, Logistic Regression, Decision Tree and Random Forest. Though our report first we analyze the dataset and discuss the preprocessing method before establishing a baseline performance through different models. Then we will discuss the accuracy and confusion matrices of tested models and compare them.

2 Data Exploration and Preparation

2.1 Data Exploration

The dataset contains 39,744 total dishes encoded in a .json file format, containing each dish's cuisine and ingredients. In total, 20 cuisines from all around the world are represented in the dataset. The number of dishes belonging to each cuisine can be seen in Figure1.

It is clear that Italian dishes are particularly overrepresented compared to the other cuisines. Conversely, other cuisines, such as Jamaican, are underrepresented. There are approximately 16 times more Italian dishes than Jamaican dishes. This class imbalance could lead to the models used being overwhelmed by the majority class,[1] leading to incorrect classification.[2] This problem would therefore need to be dealt with in some way.

The number of ingredients in each dish was also investigated. The results of this, grouped by cuisine type are shown in 2.

This graph shows that the average number of ingredients per dish in each cuisine is approximately equal, but that there is some variation. There are also some outlier dishes with many more ingredients than in the others. It was therefore decided that an additional feature, namely the number of ingredients, should be added to the dataset during the data preparation stage.

The high frequency of salt, as shown in Figure 3, also suggests that some ingredients may provide little to no clue as to which cuisine a dish belongs to. It was therefore decided that this should be investigated further during the data preparation stage, in Section 2.2.1.

Whilst looking through the dataset, it was found that there were some inconsistencies between ingredient names. For example, a large number of dishes contained *eggs* as an ingredient. However, this was not always written in the same way. In some dishes, it was listed as “eggs”, whilst in others it was simply “egg”. Furthermore, in some cases, adjectives such as “large” or “small” (and, in some cases, brand names) preceded the name of the actual ingredient. This means that any classifiers will see these essentially identical ingredients as completely different to each other, and any relationships between them which help identify a cuisine may not be found. The solution to this problem, to lemmatise and stem the ingredient names, is discussed in [6]. This is performed and explained later during the data preparation, in Section 2.2.1.

Simple tests were also performed on the dataset to validate its integrity. Any dishes which did not contain a cuisine or a list of ingredients were deleted. Any duplicate dishes were also deleted. It was also checked that all cuisine types were strings, and that all the ingredients were contained in lists.

2.2 Data Preparation

2.2.1 Ingredient Cleaning

As discussed above in Section 2.1, there were some problems with the ingredient names. The first of these was that some ingredients had extraneous words surrounding them. In order to solve this problem, the list of ingredients for

[1]Which, in this case, is Italian dishes.

each dish was converted into a string, which was then tokenised - put into a list where each word is its own list element.

This, however, resulted in a new problem: some ingredients, such as *olive oil*, were now separate list elements, even though the ingredient itself should be a combination of these two words. It was therefore decided that some words should be hyphenated together in order to preserve this information after tokenisation. This was achieved using K-means clustering, a process which resulted in revealing the words which were commonly in lists of ingredients together. Formally, this process involves the creation of k nodes (known as *cluster centres*), to which each (vectorised) ingredient is assigned to. The nodes are then updated to the mean of the ingredient vectors, and the process is repeated until there is no further change to this assignment.[12] Following this process, the top 10 closest ingredients to each cluster centre were printed. The resulting lists were examined manually to identify any pairs of words which should be hyphenated. This proved to be effective, however it is likely that human error caused some pairs to be missed due to the subjectivity of performing this task.

After tokenisation and hyphenation of certain words together, there was still problem with words such as skinless, boneless, extra. To solve this we have used a database of English Stop Words, as well as another subjective personalised list to eliminate tokens that should not be used as feature.

After this, we then stemmed the ingredient names in order to reduce them to their root forms. This meant that ingredients such as "egg" and "eggs" were both now "egg" and could correctly be identified as the same ingredients.

For using ingredients as feature we have performed a one-hot encoding.

2.3 Splitting the data

The data was split into training data, testing data and validation data so that the test data was left with the idea number of 10 000 instances which is about 25% of the whole dataset. After slitting, the training dataset was left with 22 365 instances.

2.4 Feature Selection

Generally when using high- dimensional data sets, features are selected based on some form of rating criteria. It was therefore decided, due to our high number of features (2623) and the low amount of computing power we had available, that we should do this and reduce the number of features. As well as increasing performance, this also has a second advantage of reducing the chance of overfitting.[13]

Feature selection was performed using the Chi squared test. This is a statistical test which measures the independence of features from class labels. It generalises well for nominal data, so is ideal for our dataset which is in binary form. The formula for the test is shown here:

$$t(tp, (tp + fp)P_{pos}) + t(fn, (tn + fn)P_{pos}) \tag{1}$$

$$+t(fp, (tp + fp)P_{neg}) + t(tn, (tn + fn)P_{neg}) \tag{2}$$

2.5 Balancing the Data

Since some cuisine were particularly overrepresented, balancing of the data needed to be done. According to Klein[8] balanced datasets are very important to generate robust prediction models.To make sure the class imbalance was not making the results biased, the data was balanced with the oversampling method. Oversampling means that you duplicate the recipes from the lesser cuisines to make sure you have an equal amount of recipes from each cuisines. On the contrary, undersampling reduces the recipes of the cuisines with more recipes to meet the quantity of the lowest one. The undersampling method will reduce the computation time, but there is a high likelihood of being left with a non representative or biased dataset.[8] Therefore, the oversampling method was chosen to not lose important data. Since the oversampling was done on the training data, the training dataset went from 22,365 to 88,600 instances. This is shown in 4.

3 Methods

Multiple models were tested to see which would be the most effective at predicting cuisines from the lists of ingredients as processed above. These are detailed below.

3.1 Decision Tree

A decision tree can be pictured as a series of nodes where decisions are made on which node to visit next. For example, one node may ask whether a given recipe contains the ingredient "chicken". If this is the case, the decision tree will continue onto one node, and if not, it will continue to another. This process will continue until the end of the tree is reached and a cuisine is selected.[11]

There are multiple ways in which the method of creating the decision tree can be adjusted. The first of these is by adjusting the hyperparameter which defines the maximum depth of the tree. If the tree's depth is allowed to be as large as possible, then there is a high chance that overfitting will occur.[5] This is a problem since if overfitting occurs, the model will only be good at predicting entries from the training dataset, and not at predicting real world data.

The formula used to calculate the information gain can also be adjusted. Two popular formulas are entropy and the gini index. If S represents the dataset, m is the number of classes in this dataset, and P_j is the relative frequency of class j in S, then these two values can be calculated using the below formulas:

$$Entropy(S) = -\sum_{j=1}^{m} P_j \log(P_j) \tag{3}$$

$$Gini(S) = 1 - \sum_{j=1}^{m} P_j^2 \tag{4}$$

The information gain is then defined in the standard way:

$$Gain(S, A) = Entropy(S) - \sum_{v \in A} \left(\frac{|S_v|}{|S|} Entropy(S_v) \right) \tag{5}$$

$$Gain(S, A) = Gini(S) - \sum_{v \in A} \left(\frac{|S_v|}{|S|} Gini(S_v) \right) \tag{6}$$

where v represents the possible values of a feature A and S_v is the subset of S where A has value v.[4] In general, the entropy is more expensive to compute than the gini index, due to the presence of the logarithm, however there is generally little difference in the results of the use of the formulas.[3]

3.2 K-Nearest-Neighbours

The K-Nearest-Neighbours (KNN) algorithm is a lazy algorithm. This means that it does not actually learn anything about the data, it simply remembers the data. In essence, when a new recipe is given to the model, the cuisine outputted is determined by the dominant cuisine of the k nearest training recipes to the inputted recipe. This means that there are two parameters which can be chosen.[7] The first of these is the value of k. The optimum value of this is dependant on the dataset - it is different for every problem. However, the larger the value of k, the longer it will take to train the algorithm. The other parameter is the distance function used. The most common is the Euclidean Norm, which is used in the implementation here. An advantage of this distance function is that it is a square, meaning that the orientation of the points does not matter.

3.3 Support Vector Machine

The method of support vector machines is a classification method that searches for a separating hyperplane in a high-dimensional feature space.[9] The optimal hyperplane will give the largest distance - thus, the margins - to the training data. The training data instances closest to the generated hyperplane are called support vectors.

Due to limitations in computing power, we chose the radius basis function kernel 'rbf', and the linear kernel 'linear' as the kernel type of the model. Hyper-parameter 'C' is tuned for the optimal penalty of the error term. Larger values of 'C' will produce a smaller margin with more correct classifications, and smaller values will look for larger margins, even if there are more misclassifications.

3.4 Logistic Regression

The primary objective of logistic regression is to model the mean of the response variable, given a set of predictor variables.[**?**] Logistic regression is primarily used as a binary classifier, but can also be used as a multinomial classifier. Multinomial logistic regression uses the softmax function:

$$P(y_i = k|X) = \frac{e^{\beta_k x_i}}{\sum_{j=1}^{K} e^{\beta_j x_i}} \tag{7}$$

where $P(y(i) = k|X)$ is the probability of the i^{th} observation's target value, $y(i)$ is class k, and K is the total number of classes is implemented instead of the sigmoid function in the case of generating a model to multiple class options.

Hyperparameter $C = frac1\lambda$ is used to control the complexity of the model. A smaller C value will increase regularisation strength, resulting in less complex models and less overfitting.

The Limited-BFGS algorithm is used as the algorithm for optimization for multiclass problems, as supported by scikit-learn. Because lbfgs only handles L2 penalty, the model implements "penalty" : "l2" for regularization.

3.5 Random Forest

Random forest is an ensemble learning method for classification or regression. Intuitively, in the case of classification, random forest will generate predictions by constructing a given number of decision trees and generate the most robust prediction given by the individual trees.[10] Although research by Segal suggest that random forest overfits for noisy datasets, the predominate opinion is that the Strong Law of Large Number prevents the random forest from overfitting with the increase in number of trees.[1] Subsequently, compensating the need for more computation, more trees will increase performance and output robust predictions.

The number of trees can be tuned with "n_estimators", the number of features put to consideration to split a node can be tuned with "max_features", and the minimum number of leaves required to split a node can be tuned with "min_sample_leaf".

4 Results

We established test accuracy from our default models with default hyperparameters as our baseline accuracy, and implemented grid search to find the optimal hyperparameters. The choice of the range of hyperparameters was made through assumption.

Initial testing in searching for models showed that using the entire preprocessed training data and validation set was difficult due to lack of computing power. Consequently, 10000 instances were randomly sampled from the training data to construct the model.

4.1 Descision Tree

A grid search on the hyperparameters was performed using module GridSearchCV to find the optimal hyperparameters. "auto", "sqrt", "log2" were used in searching for the best split, values for the minimum number of samples required to split an internal node ranging from 2 to 15, and values minimum number of samples required to be at a leaf node ranging from 1 to 11 were considered. The grid search chose "auto", 1 as the minimum number of samples required to split, and 2 as the minimum numbers of samples at a leaf node as the best hyperparameters, which was ultimately used for to test out test data.

The baseline model gave an test accuracy of 0.410, and the model with best hyperparameters gave a test accuracy of 0.5139.

Analysis on the confusion matrix shows that top predictions were on the recipes from the italian, followed by mexican, indian, southerns us and the chinese cuisines with a higher probability.

The respective confusion matrices are shown in Figure **??**.

4.2 K-Nearest-Neighbour

A grid search on the hyperparameters was performed using module GridSearchCV to find the optimal hyperparameters. k-values ranging from 5 to 10, leaf size ranging from 1 to 5, "uniform" and "distance" weights, and 4 different algorithms - "auto", "ball_tree", "kd_tree", "brute" were considered. The grid search chose $k = 8$, "distance" weight, and a "brute" algorithm as the optimal hyperparameter, which was ultimately used to test our test data.

The baseline model gave an test accuracy of 0.3976, and the model with best hyperparameters gave a test accuracy of 0.4297.

Analysis on the confusion matrix shows that top predictions were on the recipes from the italian, followed by mexican, indian, southerns us and the chinese cuisines with a higher probability.

The respective confusion matrices are shown in Figure 6

4.3 Support Vector Machine

A grid search on the hyperparameters was performed using module GridSearchCV to find the optimal hyperparameters. C values ranging from 6 to 12 and 2 different kernels - "linear", "bf" were considered. The grid search chose $C = 6$, kernel ' "linear" as the optimal hyperparameters, which was ultimately used to test our test data.

The baseline model gave an test accuracy of 0.3262, and the model with best hyperparameters gave a test accuracy of 0.6347.

Analysis on the confusion matrix shows that the top predictions were the recipes from the italian and the mexican cuisines, using the best hyperparameters. Using the baseline, the top cuisines were the Mexican followed by Indian and southern US. The confusion matrix for the baseline model shows that it misclassified the brazilian cuisine as Italian, Mexican and Southern US.

The respective confusion matrices are shown in Figure 7.

4.4 Logistic Regression

A grid search on the hyperparameters was performed using module GridSearchCV to find the optimal hyperparameters. $0.001, 0.01, 0.1, 1, 10, 100$ were tried for the inverse of strength of regularization. $100, 500, 1000$ were tried as maximum number of iterations. Whether solutions from previous model were considered was also a parameter which was tested. Doing this is known as a*warm start* The grid search revealed the best hyperparameters $C = 10.0$, maximum iterations 100, and using warm start, which were ultimately used to test out test data.

The baseline model gave an test accuracy of 0.7029 and the model with best hyperparameters gave a test accuracy of 0.6750.

The top predictions were the recipes from the italian cuisine, followed by mexican, indian and southern US cuisines with a higher probability.

The respective confusion matrices are shown in Figure 8.

4.5 Random Forest

A grid search on the hyperparameters was performed using module GridSearchCV to find the optimal hyperparameters. Two functions - “gini”, “entropy” were used as functions to measure quality of split, number of trees ranging from 10 to 30, minimum number of samples required to split ranging from 3 to 7, minimum number of samples at a leaf ranging from 1 to 3 was considered in the grid search. The grid search chose the “gini” function as measurement of split quality, 30 as number of trees, 1 as minimum number of samples at a leaf, and 4 as minimum number of samples required to split as the optimal hyperparameters, which was ultimately used to test our test data.

The baseline model gave an test accuracy of 0.6068, and the model with best hyperparameters gave a test accuracy of 0.6720.

Analysis on the confusion matrix shows that top predictions were on the recipes from the italian, followed by mexican, indian, southerns us and the chinese cuisines with a higher probability.

The respective confusion matrices are shown in Figure 9.

4.6 Comparisons with Training Data

Results from the previous models show that the default multinomial logistic regression has the highest test accuracy. Accordingly, the following tests will be performed with this model, but with training data in different stages of processing. Test accuracies will be compared to bring insight to our data pre-processing procedure.

Comparing results on entire processed dataset to sampled dataset The entire pre-processed dataset, before sampling 10000 instances was given to the logistic regression model and test accuracy is 0.7246. which was significantly higher than the training sample of 10000 instances whose accuracy was 0.7029.

Comparing results on not balanced, but feature reduced dataset to sampled dataset The initial dataset consisting of 22365 instances - with class imbalance, but further treated with the feature reduction procedure was given to the logistic regression model. Test accuracy was 0.7629. Again, results were significantly higher than the case with the training sample of 10000 instances.

Comparing results on balanced, but not feature reduced dataset to sampled dataset The dataset after treatment for oversampling, but without the feature reduction procedure was given to the logistic regression model. Test accuracy was 0.7235 - significantly higher than the sampled dataset, but lower than that of the case in which the dataset was not treated for imbalance.

The performance is summarised in Figure 10.

5 Conclusion

Logistic regression was the model that gave us the best accuracy. And the results show us that balancing and selecting the features was not improving the prediction outcomes, disproving the hypothesis of this paper. We could also see that the Italian cuisine, followed by the mexican, indian, chinese and southern us cuisines was the most distinguishable when predicting.

Models	**Baseline Accuracy (Default Hyperparameters)**	**Accuracy with Best Hyperparameters**
Decision Tree	0.5139	0.4310
K-Nearest Neighbour	0.3976	0.4297
Support Vector Machine	0.3262	0.6347
Logistic Regression	0.7029	0.6750
Random Forest	0.6068	0.6720

Table 1: Table showing the accuracy values for the models

6 Discussion

Despite the literature's claims to the contrary, it was found,in Section 4.6, that preprocessing and balancing the dataset had an insignificant effect when it improved the accuracy, and that it even sometimes resulted in worse accuracy than not balancing and selecting the features.

Since the dataset was very large and the computing power very low, we tried the models on a dataset of only 10000 instead of 88000. This could have affected which model is the best for the whole dataset.

The Logistic regression model proved to be the best model. That could have been because our grid wasn't big enough for the parameters. Only 4 parameters were tested per model. It is possible that changing this could have improved the results and not made the baseline more accurate than the other tested hyperparameters.

To improve the accuracy in the future, it is therefore recommended that researchers should have a large amount of computing power. This would allow them to ensure they train on the whole dataset, rather than a small subset of it.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[3] Chris Drummond and Robert C Holte. Exploiting the cost (in) sensitivity of decision tree splitting criteria. In *ICML*, volume 1, 2000.

[4] Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*, pages 1–8. Australian Computer Society, Inc., 2002.

[5] Amir H Gandomi, Mark M Fridline, and David A Roke. Decision tree approach for soil liquefaction assessment. *The Scientific World Journal*, 2013, 2013.

[6] M Ikonomakis, Sotiris Kotsiantis, and V Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4 (8):966–974, 2005.

[7] Sadegh Bafandeh Imandoust and Mohammad Bolandraftar. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *International Journal of Engineering Research and Applications*, 3(5):605–610, 2013.

[8] Kerenaftali Klein, Stefanie Hennig, and Sanjoy Ketan Paul. A bayesian modelling approach with balancing informative prior for analysing imbalanced data. *PloS one*, 11(4):e0152700, 2016.

[9] Alessia Mammone, Marco Turchi, and Nello Cristianini. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3): 283–289, 2009.

[10] Mark R Segal. Machine learning benchmarks and random forest regression. 2004.

[11] Himani Sharma and Sunil Kumar. A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR)*, 5(4):2094–2097, 2016.

[12] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.

[13] Mike Wasikowski and Xue-wen Chen. Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on knowledge and data engineering*, 22(10):1388–1400, 2010.

Appendix

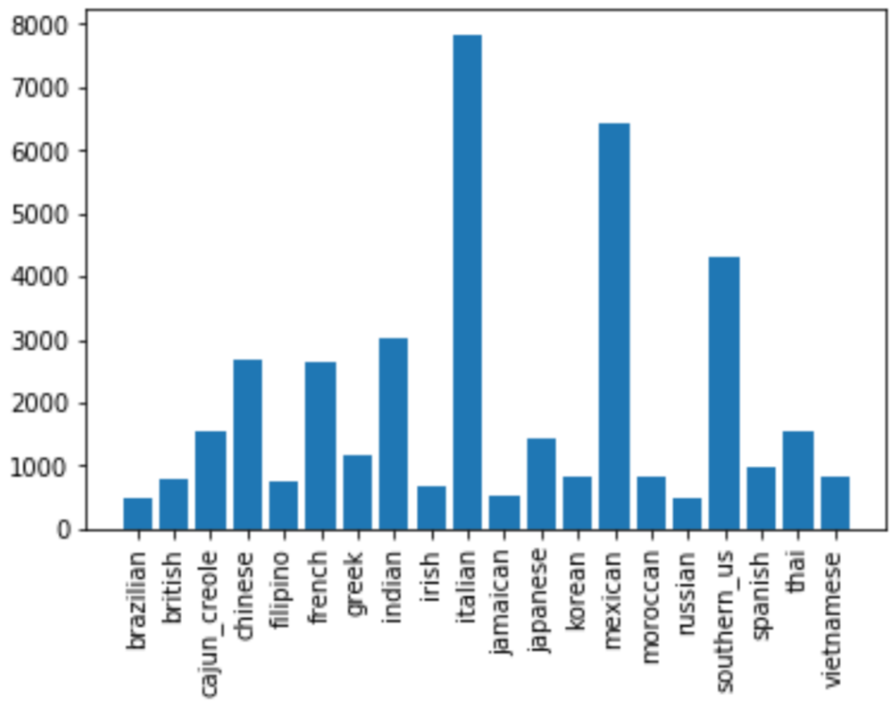

Figure 1: Graph to show the frequency of each cuisine in the whole dataset

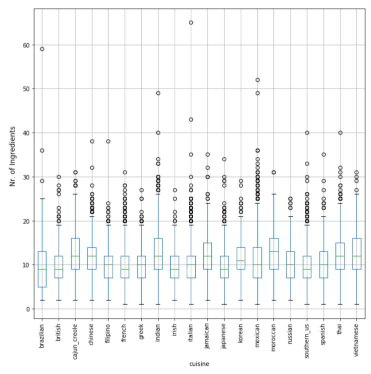

Figure 2: Graph showing the distributions of the number of ingredients in each cuisine

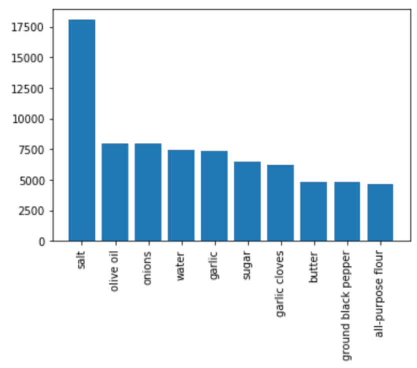

Figure 3: Graph showing the 10 most frequent ingredients in the dataset

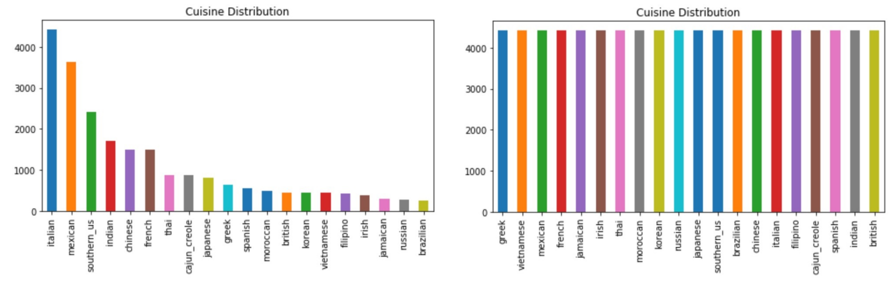

Figure 4: Graphs comparing the cuisine distributions before and after oversampling

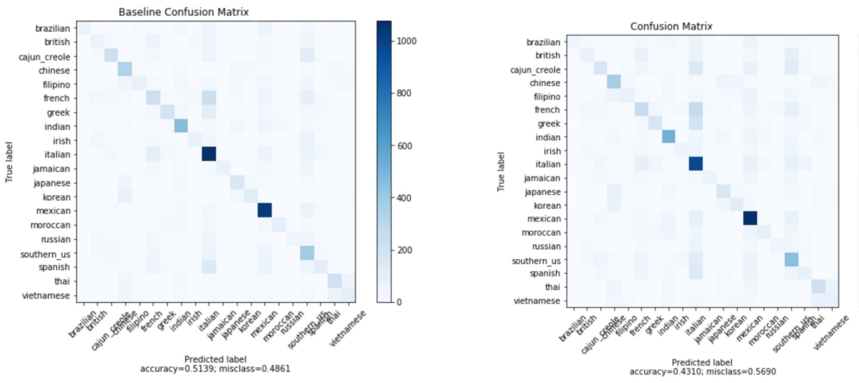

Figure 5: Confusion Matrices for Decision Trees

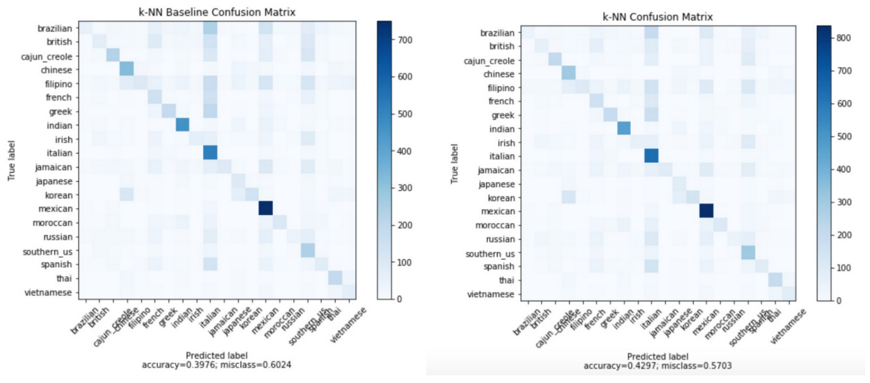

Figure 6: Confusion Matrices for K-Nearest-Neighbour

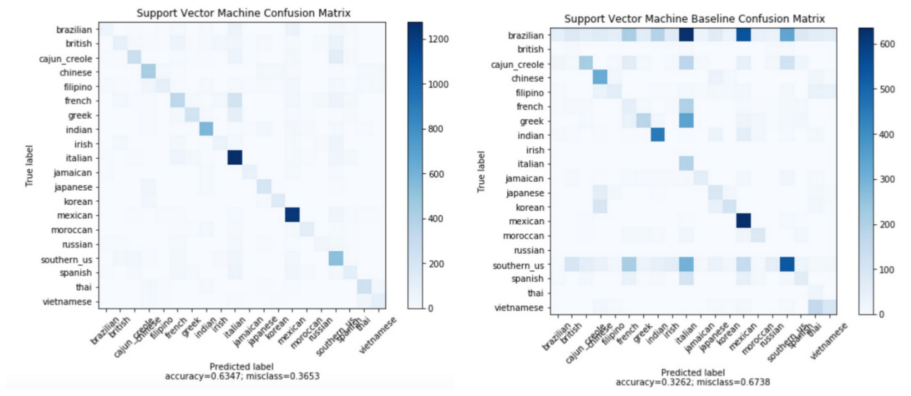

Figure 7: Confusion Matrices for Support Vector Machine

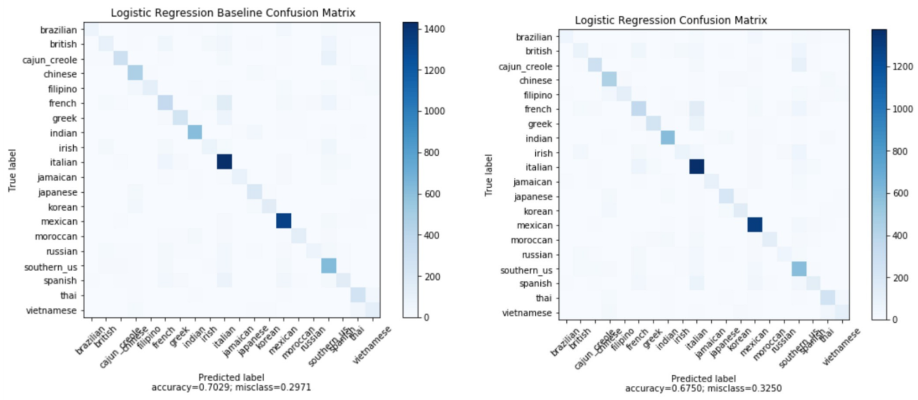

Figure 8: Confusion Matrices for Logarithmic Regression

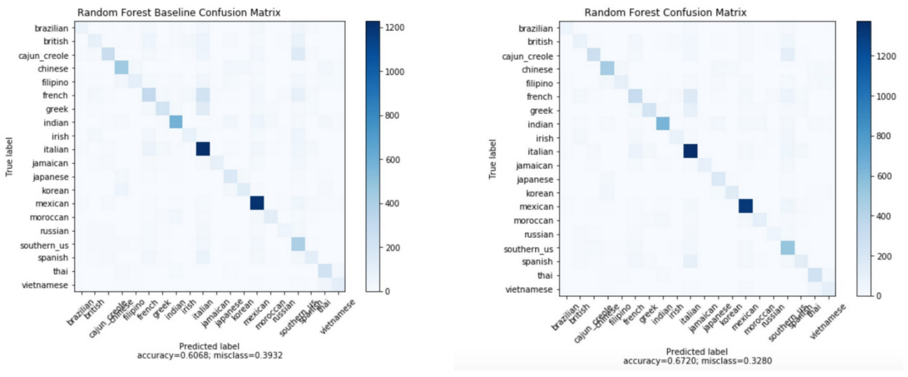

Figure 9: Confusion Matrices for Random Forest

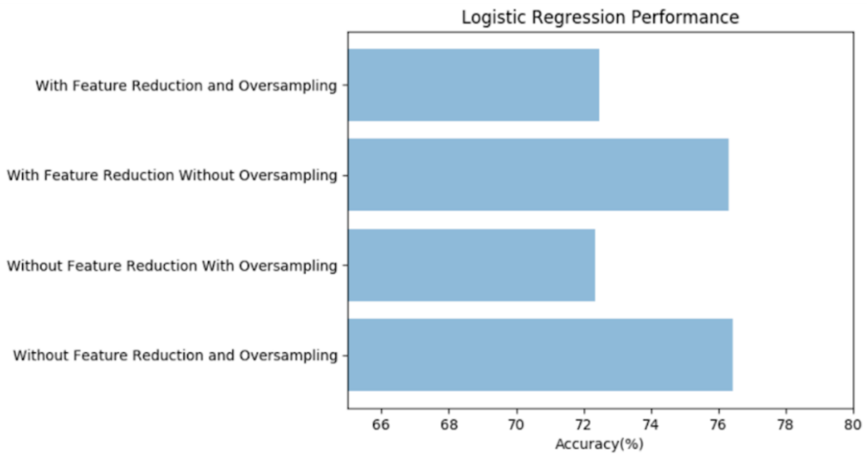

Figure 10: Summary of the accuracy of logistic regression on various datasets