A Comparative Analysis of Multi-label Classification Models for Classifying Recipes into Cuisines

Aashay Mokadam
Department of Computer Engineering
San Jose State University
San Jose, USA.
aashay.mokadam@sjsu.edu

Krupa Vadher
Department of Computer Engineering
San Jose State University
San Jose, USA.
krupadhirajlal.vadher@sjsu.edu

Pranav Dixit
Department of Computer Engineering
San Jose State University
San Jose, USA.
pranavsudhir.dixit@sjsu.edu

Vishnu Narayana
Department of Computer Engineering
San Jose State University
San Jose, USA.
vishnu.narayana@sjsu.edu

***Abstract*—Culinary arts are a historic, integral part of human culture. In this report, we detail our efforts in trying to predict the cuisine to which a recipe belongs based solely on its list of ingredients. Our aim was to study the performance of 5-6 machine learning algorithms and rank them based on their prediction accuracy. We used a dataset provided by yummly.com for the purposes of a Kaggle competition referenced in the paper, containing almost 50,000 recipes across 20 cuisines. The test set results were checked by submitting our predictions to the competition in order to receive our final scores. We found that SVMs yielded the best results, followed closely by Neural Networks, with other models trailing behind. Of these, Logistic Regression was in the lead, making it the third most effective machine learning approach for this problem.**

***Index Terms*—SVC, neural networks, Naive Bayes, k-nearest neighbors, logistic regression, ensemble learning**

I. Introduction

This project is based on identifying models which deliver the best solution to the Kaggle competition, "Whats Cooking". The objective of this competition is to train a machine learning model on a data set of recipes labeled according to their cuisine (e.g. Chinese, Indian, Southern US, etc.), and then use that model to predict the cuisine labels of an unlabelled test set (both provided to Kaggle by Yummly). The underlying problem to be solved is a text-based multi-label classification problem where categories are mutually exclusive. While the route we describe in this report is tailored to the confines of the competition, the approach can be generalized to a wide array of domains which present similar problems. Classifying music, poetry, or short stories into genres based on their lyrics/word-counts, clustering shopping mall customers based on their shopping carts, ranking the impact of talking points to approval ratings for political speeches, and identifying the features of a product which factor in a positive versus negative review are just a few examples of real-world applications our solution can be adapted into. Additionally, insights drawn from the contribution factor assigned to each ingredient can be used as a jumping off point for designing an automated cuisine-based recipe generator. We start by tackling the pre-processing operations that need to be performed on the dataset, discussed in-depth in section III of this report. Once complete, we applied a host of different machine learning techniques, comprising of Support Vector Machine Classifiers, k-Nearest Neighbors, Nave Bayes, Logistic Regression, Neural Networks, Random Forest Classifiers, and Extra Tree Classifiers. Various hyper-parameters are tested using validation and k-fold cross-validation, and the best performers from each model were selected as candidate solutions. Finally, we trained the contenders on the entire training set and made cuisine predictions on the unlabelled test set. The solutions provided by each of the models were uploaded to the Kaggle competition page to receive their respective accuracy scores. The details of the implementation of each model are discussed in section IV, and the final scores are listed in Table 3, with their discussion in section V, Part B.

Kaggle Competition "What's Cooking" at https://www.kaggle.com/c/whats-cooking

II. Survey of Related Work

Text-based classification problems are a popular field for research and application projects, and there are a host of publications which discuss solutions relevant to our problem.

During the course of this project, we referred to publications that detailed the work done on the classification of recipes into cuisines [1] [2] and the classification of songs into genres based on lyrics [3].

We also referred to Youngjoong Ko (2012), which describes in detail the TF-IDF transformation for text data [4]. This is a

technique that we ultimately implement in the pre-processing of our training and test datasets.

Reference [5] discusses methodologies we adopted for the Support Vector Machine approach used in our solution (See Section IV. F.). This deals with multiclass support vector machines and fits well with our problem.

III. Pre-processing of Data

A. Challenges Faced

We tackled numerous challenges in the pre-processing part of our solution. After extraction from the JSON format, the string data comprising of recipe ingredients is found to contain irregularities in the form of numbers, units, special symbols, accented characters, and both singular and plural forms of items. Furthermore, ingredients are not standardized, often with dozens of variations for a single ingredient. For example, Fig. 1 shows nearly two dozen different kinds of Mozzarella Cheese alone. Additionally, different adjectives describing the state, quality, or method of preparation of ingredients are present in fairly high frequencies throughout the dataset. Words like fresh, chopped, diced, and so on are unhelpful in distinguishing between cuisines, but manifest frequently enough to throw off our prediction algorithm. Finally, spelling mistakes and typos are also present in the dataset, which likely contributed to a decrease in the overall accuracy of our predictors.

```
shredded low-fat mozzarella cheese
shredded mozzarella cheese
KRAFT Shredded Low-Moisture Part-Skim Mozzarella Cheese
nonfat mozzarella cheese
2% milk shredded mozzarella cheese
KRAFT Shredded Mozzarella Cheese
SargentoA(r) Traditional Cut Shredded Mozzarella Cheese
Kraft Slim Cut Mozzarella Cheese Slices
low fat mozzarella
mozzarella string cheese
fresh mozzarella
chees mozzarella stick
smoked mozzarella
preshred low fat mozzarella chees
chees fresh mozzarella
low-fat mozzarella cheese
part-skim mozzarella
mozzarella cheese
reduced fat mozzarella
KRAFT Reduced Fat Shredded Mozzarella Cheese
fresh mozzarella balls
buffalo mozzarella
part-skim mozzarella cheese
mozzarella balls
low moisture mozzarella
```

Fig. 1. Mozzarella Cheese Variants.

Another issue which had to be taken into account was the fact that different recipes had different numbers of ingredients, meaning input features of different lengths. This had to be normalized in some manner in order to allow them to be fed into our machine learning algorithms.

B. Approach Used

To solve the discrepancy between singular and plural nouns, the entire bag of words was lemmatized using the NLTK Python library. To contend with the problem of variation in our feature set, both in size of the ingredient lists and the variation in individual ingredients, we opted for a bag-of-words based transformation of the feature space, followed by a TF-IDF vectorization [4]. TF-IDF stands for Term Frequency-Inverse Document Frequency and is a product of two ratios: the frequency of a word in the recipe and the frequency of the recipes that contain this particular word in the entire collection.

```
'romaine'    (0, 158)    0.2079
'lettuce'    (0, 218)    0.1465
'black'      (0, 480)    0.1517
'olive'      (0, 683)    0.3481
'grape'      (0, 888)    0.3167
'tomato'     (0, 1005)   0.4043
'garlic'     (0, 1010)   0.1097
'pepper'     (0, 1087)   0.356
'purple'     (0, 1429)   0.2773
'onion'      (0, 1761)   0.1395
'seasoning'  (0, 1763)   0.1118
'garbanzo'   (0, 1880)   0.1039
'bean'       (0, 2046)   0.2491
'feta'       (0, 2156)   0.357
'cheese'     (0, 2257)   0.2396
'crumbles'   (0, 2600)   0.1481
```

Fig. 2. Term Frequency-Inverse Document Frequency Transformation

The equations below represent this calculation. Here, t denotes the terms, d denotes each document, and D denotes the collection of all documents. $tf(t,d)$ represents the frequency of the term t in document D.

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{1}$$

$$idf(t,D) = log\frac{|D|}{1+|\{d \in D : t \in d\}|} \tag{2}$$

This transformation enables us to treat each recipe as a row in an incredibly sparse matrix of constant size, where each index represents a word, preferably an ingredient, present in that recipe (see Fig. 2). Each individual recipe would hence contain either floating point values between 0 and 1 (denoting the significance of that word to the recipe) if the word corresponding to the index is contained by the recipe or zero if the word is absent. Hence, the key portion of the ingredient (in this example, mozzarella) could be recognized as a feature and receive an appropriate importance score, whereas the quirky modifiers would fall to the side.

C. Summary of pre-processing

The following operations are performed to clean data and reduce the feature dimensionality:

- Convert each recipe to a string of words it comprises of
- Convert each word to lowercase
- Remove numbers
- Remove symbols, punctuations, and special characters
- Remove redundant white spaces
- Convert accented letters using the unidecode module
- Remove units, stop-words, and certain adjectives
- Lemmatize words

- Convert training and test sets from collections of strings to sparse TF-IDF matrices

Finally, the resultant cuisine values from the training set are converted to a label-encoded form. We prefer this to a one-hot encoded form since the categories involved are mutually exclusive.

IV. Technical Approach

A. k-Nearest Neighbors

The k Nearest Neighbors algorithm is a powerful classification tool when used in the correct context, and doesnt require any training. Testing with k Nearest Neighbors involves determining the closest k training points to a point thats an instance of the test set. Then, the y value of the majority of the nearest k training points is selected as the predicted y value. Despite the sheer simplicity of the algorithm and the fact that no actual "training" is required, it can still be somewhat computationally intensive, as the k nearest points to the tested item must still be found. To find the nearest points, the algorithm calculates a "distance" between the tested item and training items, which can be expressed as numerical values across d dimensions, with larger values of d resulting in a more computationally complex algorithm.

According to the work done by Choi et al. on classifying music by their lyrics, k Nearest Neighbors performed the worst compared to other algorithms such as SVMs and Naive Bayes [3]. Despite this, there are plenty of differences between our datasets, namely that theirs is based on text classification which harbors many complexities in how and in what order words are present, while the order of ingredients doesn't matter and they arent as dependent on each other. In the first iteration of adopting K Nearest Neighbors to the problem, we performed primitive cleaning operations on the dataset and applied the Jaccard Similarity Index as a means of measuring distance between two recipes. Jaccard similarity measures how similar two sets are by the following formula:

$$Jaccard(A, B) = \frac{len(A \cup B)}{len(A \cap B)} \tag{3}$$

For a preliminary attempt, the Jaccard similarity index did an adequate job of measuring and representing the equivalent distance between two recipes that the k Nearest Neighbors algorithm could make proper use of, yielding an accuracy of 72.76%. Later, we implemented a more sophisticated means of cleaning up the data (discussed in Section III). The matrix obtained was then used alongside various k Nearest Neighbor algorithms from the sklearn library, which provided us with a much higher validation score. This large difference is mostly due to the better state of the training set and the TF-IDF matrix that added weights to the significance of different ingredients in their ability to differentiate between different cuisines.

B. Naive Bayes

The Naive Bayes algorithm is based on the Naive Bayes formula, which predicts the probability of different outcomes based on the input variables. Naive Bayes is a probabilistic model that makes classifications based on attributes of categories involved. Also, each sub-element must be relatively independent of each other. This is applied to our problem, but with the probability of each cuisine being calculated based on the occurrence (or lack thereof) of each ingredient. The algorithm follows the general form:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y), \tag{4}$$

where $\hat{y}$ is the predicted label and $(x_1, x_2, ...x_i)$ are the input features.

We tested a few variations of this, ultimately settling for a Complement Naive Bayes algorithm, which is supposed to outperform its cousin, the Multinomial Naive Bayes on text-classification tasks. Despite Naive Bayes appearing to be a good fit for this problem set, as each variable should theoretically increase the probability of the dish belonging to a particular type of cuisine and the cuisine with the greatest probability selected, it was overall not a very effective classifier.

C. Ensemble Learning Methods

The motive of using ensemble methods is to get a better generalization by combining the predictions of various base models and achieve a common optimal prediction. Decision tree models (which form the base of these strategies) operate by creating sets of decision parameters (usually threshold values for features) based on the training set feature values and make predictions based on these decision parameters. We implemented two ensemble techniques in the course of our project (described below), where various decision trees are built independent of each other, and their predictions are averaged to get the prediction of a single estimator. The act of averaging many different models reduces the overall variance and improves generalization by constraining overfitting.

Cross-validation gave us the best results on using an ensemble of 300 decision trees for each of the following.

1) Random Forests: An ensemble of decision trees classifiers is used by fitting them into various sub-samples of the dataset. Each tree is built from a random data point, and splits between nodes are computed based on best splits using random subsets of input data features, as opposed to the entire feature space. In our case, this would imply that decision tree nodes are built based on the TF-IDF scores of a fraction of all ingredient scores.

2) Extra (or Extremely Random) Trees: This algorithm brings an additional stage of randomness to the Random Forest Classifier; in this strategy, decision nodes are based on the best split amongst a set of random, uniform splits, instead of the best overall split. Hence, decisions are made less deterministically than those of the Random Forest Classifier.

D. Logistic Regression

Logistic Regression is a linear classification model and a statistical method of predicting the estimations of a trained

model. When the target variables are either binary or categorical, the sklearn module uses the "multi-class" hyperparameter to distinguish between them by accepting values of either 'OvR' or 'multinomial' respectively. The two regularization techniques L1 (Lasso) and L2 (Ridge) are supported by the OvR and multinomial versions, where OvR supports both L1 and L2, whereas multinomial supports only L2. These techniques are used to address overfitting and feature selection. L1 is efficient for feature selection whereas L2 is efficient in avoiding overfitting. For the purpose of predicting the cuisines, where the target is categorical, we used 'multinomial' parameter which will predict the cuisines given the list of ingredients. Cross-validation was used to select the regularization hyperparameter C, and a default of $C = 1$ seemed to yield optimal results.

E. Neural Network

Neural Networks are a powerful learning paradigm that employs an interconnected mesh of layers of 'neurons'. Each neuron accepts a signal (sum of weighted inputs from neurons in the previous layer), applies a non-linearity to it (called an activation function), and passes the resulting value to every neuron in the next layer. Our neural network solution uses the popular Rectified Linear Units (ReLU) as the activation function, described by the following equation.

$$f(x) = max(0, x) \tag{5}$$

Once the outputs of the final layer have been generated, these are compared with the desired output and an 'error' or 'loss' value is computed. This value is then used to adjust the weights of every synapse (i.e. the connection between two neurons) in the network through gradient descent. This entire process takes place over 1 'epoch'. Due to the nature of our problem, we use a variation of the categorical cross-entropy loss function (described in the equation below) called "sparse categorical cross entropy". The "sparse" variation is used since our expected output categories are mutually exclusive (i.e. a recipe can belong to only one cuisine).

$$loss = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{6}$$

where M = number of classes, y = 0 (if the class labels match) or 1 (if the labels don't match), and p = probability prediction that observation 'o' belongs to class 'c'.

Using validation, we found the best results on training a 2-hidden layer network with a dropout of 50% at every layer (including the inputs), meaning that half of all the neuron in a layer will randomly be disabled at random for every training epoch. This curbs overfitting, which neural networks are prone to doing. Table III further lists the details of the implementation.

F. Support Vector Classifier

Support vector machines are binary classifiers which are fairly well resistant to noise in the training data. Classifiers created from SVM usually give great results in real life problems. Support vector machines are binary classifier and cannot inherently be used for multi-class classification. Since the cuisine data had dishes from various cuisines, using a simple SVM would not work for classification. Fortunately, there are various techniques that unable us to perform multi-class classification using SVM's [5].

1) One versus One: : In this techniques a separate classifier is used to classify a pair of class. So, if the data set has k classes then this technique would create $k(k-1)/2$ classifiers. For a pair of class i and j the SVM optimization problem would be:

$$\min_{w_{ij}, b_{ij}, \xi_{ij}} \quad \frac{1}{2} {w_{ij}}^T w_{ij} + C \sum_{n=1}^{N} \xi_{nij} \tag{7}$$

$$subject\ to \quad y_n \left({w_{ij}}^T x_n + b_{ij} \right) \geq\ 1 - \xi_{nij}$$

$$\xi_{nij} \geq 0$$

2) One versus Many: : In this technique, one classifier is used to classify one class from all other classes. In total there are only k classifiers for k classes. For each class, the i^{th} optimization problem would be

$$\min_{w_i, b_i, \xi_i} \quad \frac{1}{2} {w_i}^T w_i + C \sum_{n=1}^{N} \xi_{ni} \tag{8}$$

$$subject\ to \quad y_n \left({w_i}^T x_n + b_i \right) \geq\ 1 - \xi_{ni}$$

$$for\ n = 1, 2, \ldots, N$$

$$\xi_{ni} \geq 0\ for\ n = 1, 2, \ldots, N$$

Each one would generate a classifier as follows:

$$w_1^T x + b_1 = 0$$

$$w_2^T x + b_2 = 0$$

$$.$$

$$w_k^T x + b_k = 0$$

To predict the class of a new point, the point is classified with all k classifiers. The classifier which predicts a class with the maximum amount of signal is chosen as the final class.

$$class\ of\ new\ point\ x = argmax_{i=1,\ldots,k}(w_i^T x + b_i) \tag{9}$$

To fine tune the hyperparameters (such as regularization parameter C, kernel and degree), we used validation and cross-validation. Validation was used to find a general range of values of hyperparameters, which showed that the best results were found by using RBF and polynomial kernels. The polynomial kernel with $degree = 2$, C in the range of 10000 and gamma set to "scale" gave us the best results. Cross-validation was then used to fine tune the value of C.

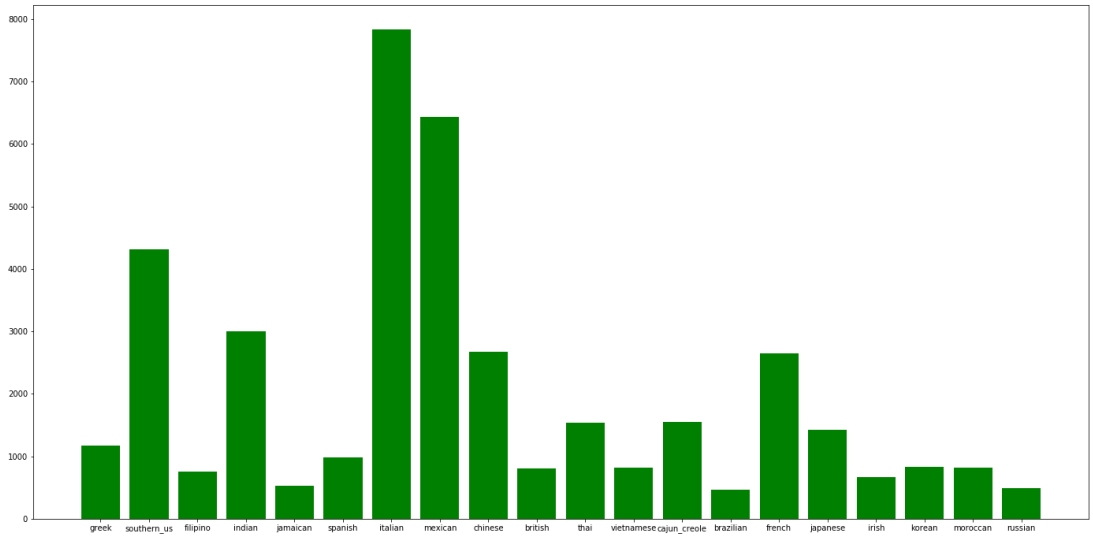

Fig. 3. Distribution of training set recipes according to cuisines.

TABLE I
MOST FREQUENT FEATURE BY CUISINE

Cuisine	Most Frequent Ingredient
Mexican	Pepper
Cajun Creole	
Brazilian	
Filipino	
Jamaican	
Spanish	
Southern US	Salt
Russian	
British	
French	
Irish	
Thai	Sauce
Vietnamese	
Japanese	
Chinese	
Indian	Ground
Moroccan	
Greek	Olive
Italian	Cheese
Korean	Sesame

V. EXPERIMENTAL METHODOLOGY

A. Features of the Dataset Used

The dataset provided in the Kaggle competition (found at `https://www.kaggle.com/c/whats-cooking/data`) consists of two parts:

1) Training Set: This is a labeled set of 39774 recipes spread across 20 different cuisines provided in JSON file format. The frequency distribution of cuisines is not uniform; the bias towards certain cuisines can be seen in the depicted bar graph shown in Fig. 3.

It can be seen that the cuisines Italian and Mexican have substantially more recipes than their peers (7838 for Italian, 6438 for Mexican), while Brazilian, Russian, and Jamaican have the least (467, 489, and 526 respectively).

A brief analysis reveals to us the most frequent word (i.e. feature) from each cuisine (following pre-processing operations). Table I lists them. The most glaring observation (besides the loose clustering of cuisines based on geographic and cultural similarities) is that multiple different cuisines have high frequencies of the same ingredients (e.g. salt). However, this is taken into account by the IDF portion of the TF-IDF transformation, as evident in the low TF-IDF scores assigned to common ingredients shown in Table II.

TABLE II
TF-IDF SCORES ASSIGNED TO COMMONLY OCCURING INGREDIENTS

Ingredient	TF-IDF Score
salt	0.0577
pepper	0.05215
oil	0.05344
butter	0.03442
garlic	0.04947
sauce	0.03647

After pre-processing operations have been performed, the input feature space has a dimensionality of 2822.

The resulting matrix of input features $X = [X_1, X_1, ...X_N]$ has the dimensionality: `(39774 x 2822)`.

Since we are using a label-encoded transformation in order to represent our cuisines, for each data point (X_n, y_n), y_n is an integer value between 1 and 20. The corresponding vector of input labels $y = [y_1, y_2, ...y_N]$ hence has the dimensionality: `(39774 x 1)`.

TABLE III
HIGHEST SCORES RECEIVED BY EACH ALGORITHM

Model	Best Prediction Accuracy Score	Comments
Support Vector Classifiers	81.858%	Polynomial Kernel[a] (One vs Rest) $C = 10250$
Neural Networks	81.566%	2 hidden layers[b] 2000 epochs trained[c] $Dropout = 0.5$
Logistic Regression	78.439%	Multinomial, $solver = sag$
Extra Trees	77.986%	Ensemble of 300 trees
Random Forests	75.693%	Ensemble of 300 trees
k-Nearest Neighbors	75.643%	$k = 21$
Naive Bayes	70.172%	Complement Naive Bayes

[a] $2^{nd} Degree$
[b] 1024 Neurons in the 1^{st} Layer, 128 Neurons in the 2^{nd} Layer
[c] Averaged over 10 runs

2) Test Set: This is an unlabelled set of 9944 recipes spread across 20 different cuisines in JSON file format. After undergoing the same pre-processing as the training set, the resulting matrix of test features $X_{test} = [X'_1, X'_2, ...X'_N]$ has the dimensionality: `(9944 x 2822)`.

Our test set predictions will, therefore, be a vector with dimensionality: `(9944 x 1)`, i.e. one integer value between 1 and 20 for each of the 9944 data points. This vector of integers will then undergo an inverse label encoded transform to yield a vector of 9944 cuisine names, each of which will finally be combined with the ID of the corresponding recipe (provided in the test data set) to create a submission file.

B. Results and Discussion

Table III summarizes the most accurate prediction results from each of the models we used. One-versus-All Support Vector Classifiers perform the best, with an accuracy of 81.85%, with Neural Networks coming a close second, with an 81.56% accuracy. However, the computational overhead for the neural network solution is much greater than that of the support vector classifier. This makes the Support Vector Classifier the clear candidate for the best classifier, all factors considered. Other configurations of neural networks and support vector classifiers had slightly worse performances (in the ballpark of 80% to 81%).

Of the remaining models, a Logistic Regression classifier with fairly basic hyperparameters yielded the best result (78.44%, with other configurations in the 77% range). This was followed by the Extra-Trees classifier (nearly 78%), with the Random Forests and 21-Nearest Neighbors classifiers trailing behind, each with an accuracy of around 75.6%. Finally, the Naive Bayes classifier was the worst performer, with a meager prediction accuracy of 70.17%.

Comparing our best results to those from the Kaggle leaderboard, our SVC candidate would have ranked at 17^{th} place, and our Neural Network candidate would have placed 39^{th} (out of 1388). The highest score on the leaderboard, however, is 83.21%, meaning there is scope for improvement.

VI. CONCLUSION AND FURTHER SCOPE

A possible avenue to get better results is to further refine our preprocessing pipeline. Fig. 1 shows the presence of brand names in some ingredient values, which could be removed. Kalajdziski et al. talk about the removal of vacuous adjectives (such as "chopped", "smoked", "fried", etc.) using Part-of-Speech tagging, which would likely improve our prediction accuracy [1]. Li and Wang discuss an alternate approach to cut down the feature space by calculating the Mutual Information of all ingredients and only considering to top 60% [2]. Either or both of these approaches, if integrated into our solution, would likely result in improved performance.

Another approach is to take the semantic clustering of cuisines into account. For example, Asian cuisines (Thai, Filipino, Japanese, Chinese, Vietnamese, Korean) are likely to be clustered together and have a higher probability of being misclassified as another. We get a peek at this behavior in Table I. Hence if recipes are first assigned cuisine groups, and then a second classification with stricter conditions is performed to identify individual cuisines, there is potential for more accurate results.

The above step also brings us into the territory of data mining. Insights drawn from both the training set (e.g. mutual occurrence of ingredients), as well as our solution (contribution factor of each ingredient to each cuisine) are a stepping stone to creating applications that generate new and innovative recipes from the desired cuisine based on a list of potential ingredients.

As discussed in the introduction, the ideas implemented and discussed so far can be generalized to other multiclass text-classification problems with similar feature-sets.

VII. INDIVIDUAL CONTRIBUTIONS OF TEAM MEMBERS

Aashay Mokadam	Data Preprocessing, Neural Network Compilation of Project Report
Krupa Vadher	Logistic Regression, Ensemble Methods
Pranav Dixit	Support Vector Classifiers
Vishnu Narayana	k Nearest Neighbors, Naive Bayes

REFERENCES

[1] S. Kalajdziski, G. Radevski, I. Ivanoska, K. Trivodaliev and B. R. Stojkoska, "Cuisine classification using recipe's ingredients," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2018, pp. 1074-1079. doi: 10.23919/MIPRO.2018.8400196

[2] Li, Boqi, and Mingyu Wang. "Cuisine Classification from Ingredients."

[3] Choi, K. , Lee, J. H., Hu, X. and Downie, J. S. (2016), Music subject classification based on lyrics and user interpretations. Proc. Assoc. Info. Sci. Tech., 53: 1-10. doi:10.1002/pra2.2016.14505301041

[4] Youngjoong Ko. 2012. A study of term weighting schemes using class information for text classification. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '12). ACM, New York, NY, USA, 1029-1030. DOI: https://doi.org/10.1145/2348283.2348453

[5] Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. Trans. Neur. Netw. 13, 2 (March 2002), 415-425. DOI: https://doi.org/10.1109/72.991427