Introduction

Flavours, cooking styles and ingredients vary widely across cultures. The "What's Cookin'" Kaggle project asks competitors to submit machine learning models to predict the cultural origin of a cuisine based on its ingredients. Potential application areas for these classification models include websites and apps designed for sharing recipes. Automated classification of recipes would be very useful for a service where submitters and the cuisines they share can come from such diverse origins.

This problem has a number of challenging properties which allows us to demonstrate techniques learned in class. It is a multi-class problem with imbalanced classes and the input variables require some natural language processing. There are also a variety of methods that can be used to encode the data. All of this allows for much experimentation with representation and preprocessing. In the following pages we will outline the data and problem objectives, describe our methodology for preprocessing and data transformation. We will then discuss our model construction and assessment and conclude with a discussion of results and future work.

Problem and Data Description

The data for this project was provided by Yummly and consists of a JSON file containing the recipes. There are 39,774 recipes and there are approximately 6000 unique ingredients in the training dataset. There are 20 different type of cuisine appearing at different frequencies throughout. They consist of British, French, Indian, Thai, Vietnamese, Mexican, Brazilian, Russian, Italian, Greek, Chinese, Japanese, Korean, Filipino, Southern United States, Cajun/Creole, Moroccan, Spanish and Irish (Appendix A, Figure 1). Each recipe contains an identifying number, a cuisine label, and a list of ingredients (Appendix A, Figure 2). While the ingredients are provided as a list of terms, many ingredients contain descriptor words that refer to the state or preparation of the core ingredient. For example there is "dried basil" and "shredded cheddar cheese". While this data does not contain syntactic information such as that found in prose text samples, it does still pose a number of natural language processing challenges including how to manage the adjectives and ingredients with similar meanings.

Preprocessing and Data Transformation

Early in our project we decided to represent the ingredients by breaking them into individual words. It is challenging to find a good representation for ingredients because the variations suggest different interpretations. For example, given both "diced tomatoes" and "sliced tomatoes" it is difficult to decide *a priori* whether a slight difference in preparation is important enough to warrant keeping both ingredients separate, rather than combining them because they have the same core ingredient. By breaking each ingredients into an individual word, we allowed tomatoes to be represented as a single variable thus aggregating "tomato-ness" into a single dimension. However, this also produces somewhat awkward orphaned descriptor words such as "sliced" or "diced" from the above example or "black" from "black pepper" or "black beans". We decided that the single word representation would capture the core ingredients such as "tomato" or "cheese" most clearly, furthermore some orphaned descriptors could also contain valuable information useful in classification. Breaking ingredients into individual words produces 3157 unique words.

Once the ingredient words are separated we proceed with preprocessing the words in two steps. The first step is to clean up the original words so that only useful information is kept. The second step is to assign weights to words according to a weighting scheme. In this project, we considered Lemmatization and Stemming to parse the original ingredient words and TF-IDF weighting scheme to generate the instance matrix.

Stemming and Lemmatization

In natural language processing, both Lemmatization (L) and Stemming (S) are used to group words with different inflectional and derivationally related forms into a single base form [1]. However, the two methods deal with words in a completely different manner. S chops off the endings of words according to rules that differ across implementations and returns the stem of words. L removes inflectional endings only, but it sometimes also changes the entire word according to morphological

analysis [2]. In this project, both methods will transform “tomatoes” into “tomato”, but S transforms “noodles” into “noodl” while L returns “noodle”.

S has been explored and improved for years. The two famous S implementations, Porter and Lancaster, are both simpler and faster than lemmatizers. Nevertheless, they often lead to bad results that are not human-readable. For tasks that rely on efficiency or does not require interpretability of the results, S is usually good enough. L is a fairly new player in the field, and we only found one implementation by WordNet. L is designed to produce linguistically valid results, which suits our needs in this project since we want some insights on ingredients. However, L relies on the dictionary that is supposed to cover all vocabularies and complete morphological analysis of them [2]. Creating or maintaining such dictionary is essentially impossible, and particularly, the implementation we used in this project is not optimized for food.

TF-IDF

TF-IDF is a weighting scheme for text classification. TF stands for term frequency and IDF stands for inverse document frequency. This weighting scheme is widely used in information extraction and text classification. TF is calculated by dividing the number of occurrence of an ingredient in a recipe by the total number of words in that recipe. It takes into consideration that the length of ingredients is different across recipes, and so TF does not assign high values to repeated words in long lists. IDF is calculated by dividing the total number of recipes with the number of recipes containing a certain word. It measures the importance of a word. If a word tends to appear everywhere, then it carries little useful information for classification [3].

There is, however, a few technical glitches when applying TF-IDF in this project. Since most words appear in a recipe once only, TF in a recipe is essentially just 1/length for all words. Moreover, IDF does not incorporate the class labels. If a word appears everywhere in only one cuisine, its IDF value will be low, but this word might be very useful in classification. The effect of TF-IDF in this project is even more complicated when considering the imbalanced data, and so it is certainly not the best weighting scheme for this project. Despite our observations, we decided to apply TF-IDF for its good performance on other text classification tasks.

Experiment on Data

To assess the performance of the three text parsing methods, namely L on nouns, L on nouns and adjectives, and S (Porter), we trained linear SVM’s on the data processed by those three methods separately, and used them to classify test set. As expected, the two L methods produce similar results and have nearly identical performance. L on nouns keeps 2798 words with 78% accuracy, and L on nouns and adjectives keeps 2795 words with 78% accuracy. The L on adjectives does not seem to affect the results mainly because most adjectives are kept in original form, and so it is not necessary to parse adjectives in this project. S further reduces the number of words down to 2588. This is consistent with the fact that totally different words may have the same stem and end up merging into that stem. The accuracy of S, about 77%, is slightly worse than the other two methods.

All three methods manage to reduce the dimension of the data and together with TF-IDF, significantly improve the accuracy. Due to time constraint and the scope of this project, it is not practical to compare all three methods in all models. We then decided to proceed using L on nouns since it maintains the vocabulary form of words and it does not group totally different words into the some combination of characters.

PCA

Principal Component Analysis (PCA) provides a computationally efficient method for transforming data to reduce dimensionality and explore its variance structure. PCA produces a linear transformation of the data where the new dimensions are orthogonal and capture the greatest variance. The new dimensions are referred to as principal components and are ranked according to the amount of

variance within them. Furthermore, taking a subset of the top ranked components provides an approximation to the full dataset [4].

PCA was applied for two purposes. Firstly, we wished to better understand how variation occurred amongst recipes when represented using TF-IDF. Each of the principal components is produced from a linear weighting of ingredients. By investigating the weights assigned to ingredients in the most significant components, it is possible to better understand how ingredients contribute to variance across the cuisines. Furthermore, creating features that combine ingredients may produce a better representation of the data in the sense that a single dimension may be able to capture more complex traits such as flavours rather than having these traits dispersed over several ingredients.

Secondly, PCA can be used for dimensionality reduction. While the dataset is small enough that it is feasible to train most models on, we wished to understand how the reduction in dimensionality influenced computational efficiency and accuracy. Fewer dimensions would allow for faster training of models. Also, it could result in better generalizability of the classifier by avoiding the 'curse of dimensionality' and having fewer parameters necessary.

To generate visualizations based on the PCA we used the R implementation and the ggplot2 package. Below, a subset of the recipes and cuisines have their first two principal components visualized. As can be seen along the x-axis, the first principal component does poorly in separating cuisines.

The two wordclouds show the variables with the 30 largest magnitude weightings in the linear transformation for the first component. Large and positively correlated terms are shown in red and large negatively correlated terms are shown in blue. Size corresponds to the size of the magnitude of the coefficient. The correlated variables suggests that the largest dimension of variability in the data is linked to ingredients that separate baked goods and desserts from those recipes pertaining to cooked meals. As seen in the figure, European-influenced cuisines such as Italian and Mexican tend to have a longer "tail" in the first principal dimension which suggests that baked desserts may be more common in these types of cuisines. The second principal component correlates positively with ingredients common in Asian cuisines such as "soy", "sesame", "ginger" and "rice". It negatively correlated with ingredients more common in European-style cuisines, such as "cheese" and "olives". This explains the better separation between cuisines along the second dimension (Appendix B, Figure 1).

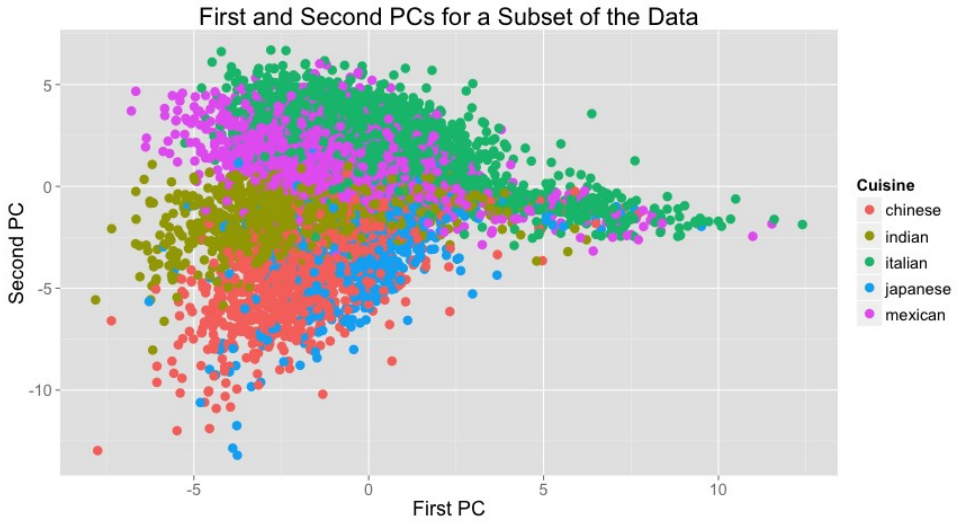

Figure 1: The above figure depicts a subset of the cuisines and data visualized using the first and second principal component from regular PCA.

Figure 2:The wordclouds depict the ingredient loading from the PCA transformation for the first principal component, with positively correlated ingredients (in red) and negatively correlated ingredients (in blue). Size corresponds to the magnitude of the coefficient.

The third principal component is positively correlated with spicy ingredients such as “jalapeno”, “cayenne”, “chillies”, and “turmeric”. It is negatively correlated with less spicy ingredients such as “cheese”, “oil” and “vinegar” and provides good separation between cuisines with spicy ingredients, such as Indian and Mexican, from other cuisines (Appendix B, Figure 2 & 3).

We also investigated model performance after PCA dimensionality reduction. The PCA for model evaluation was performed using the RandomizedPCA implementation in the Scikit-learn package. This implementation is a more efficient, although approximate, computation of the transform when compared with the R implementation. We used the approximate form as we required many more PC’s than for the visualizations above and the model training was performed in Python. We ran a linear SVM model, one-versus-all, on the reduced datasets, splitting them 75/25 into training and test sets. Including 2000 PC’s achieved 79.1% overall accuracy, for 1500 PC’s; 79.1%, 1000 PC’s; 79.0%, 750 PC’s; 78.9%, 500 PC’s; 77.65% and 250 PC’s; 75.2%. These results suggest that performance does not begin to degrade significantly until there are less than 500 components included indicating that a 83% reduction in the number of dimensions was possible using PCA. However, no significant increase in accuracy is observed. The Kaggle score for 2000 PC’s is 78.7% which is comparable to the full dataset results.

SPCA

Analysis of the PCA results suggests that at least some of the variance, including that associated with the first principal component, is related to variation within cuisines more that between cuisines. However, the principal components themselves do seem to have meaningful interpretations with respect to the flavours and ingredients found in the food. This suggests that a linear weighting of variables may be suitable but could be improved by targeting variance between cuisines using Supervised PCA.

We used the supervised PCA implementation created by Barshan and Ghodsi [5]. We used it to extract 1000 principal components on a 70% training set and implemented the transformation on a hold out set of 30%. We chose 1000 based on the PCA results and computational limitations. We then trained a linear SVM, again using the Python Scikit-learn package, to evaluate the effectiveness of the new representation; training on the 70% partition and testing on the 30% hold-out. However, results are very modest - only achieving 48% overall accuracy.

Visualizing the first two principal components for SPCA, we saw strong separation between the largest two cuisines; Italian and Mexican. Unfortunately, better separation is not found in the second or higher principal components. In fact, it seems SPCA gives worse separation than regular PCA for this dataset, at least in the three main principal components, and classification results suggest this extends to higher components as well (Appendix B, Figure 4).

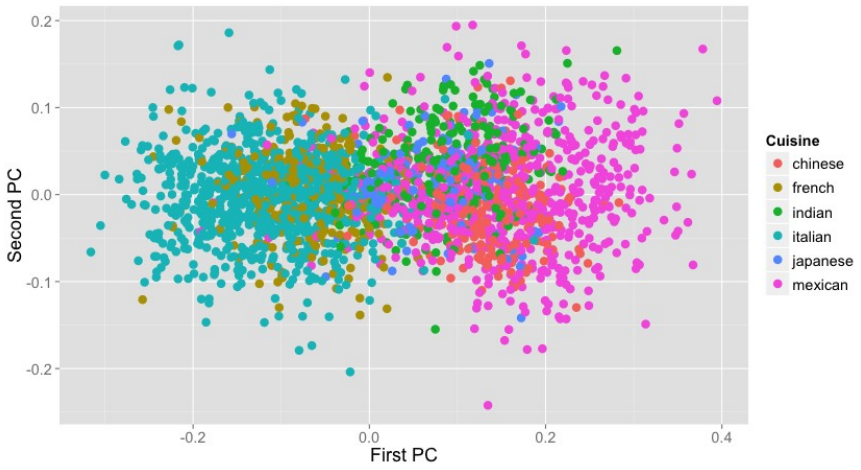

Figure 3: The above figure depicts a subset of the cuisines and data visualized using the first and second principal component from SPCA.

Visualizing the ingredient weightings in the transformation, we saw that the first component is positively correlated with ingredients typical of Mexican dishes and negatively correlated with those found in other cuisines. However, if we looked at the second component, the results are much less easy to interpret, with a variety of ingredients included that are not drawn from any single culinary tradition (Appendix B, Figure 5)

Figure 4:The wordclouds depict the ingredient loading from the SPCA transformation for the first principal component, with positively correlated ingredients (in red) and negatively correlated ingredients (in blue). Size corresponds to the magnitude of the coefficient.

Model Assessment

It is well established in literature that linear models such as Linear SVM's, Logistic Regression and Multinomial Naive Bayes perform better than more complex models on text classification. Thus, we wanted to try out these models on the raw data i.e. without implementing any sort of feature selection technique in order to have a qualitative approximation of how much work needs to be put into feature selection. Therefore, we tried a Linear SVM model with multiclass feature (one-vs-all), and obtained about 65% accuracy on Kaggle. We got similar accuracy with Multinomial Naive Bayes as well as Logistic Regression.

This mediocre performance motivated us to pursue feature selection strategies such as PCA, Lemmatization and Stemming. Finally, as discussed in detail in the previous section, we decided to use Lemmatization on the raw data and then use TF-IDF on the lemmatized data, and applied text classification models on this data.

Linear SVM

We used kernelized SVM with linear kernel and multiclass support which is handled according to the 'one-vs-the-all' principal. We experimented with nonlinear kernels such as RBF, Polynomial without much success. As per our literature survey, the choice of kernel functions do not affect text classification performance much [6], and from previous experience with the Ad Approval project, linear models worked well in text mining problems. Also the optimization problem involved in the linear SVM is far easier to solve compared to the polynomial or RBF kernel, and henceforth we went with the linear kernel. We used L2 regularization to reduce the impact of 'outliers' with the penalty parameter set to 1 and obtained classification success rate of 79% on 5 fold cross validation and 78.8% on Kaggle. This is a significant improvement as compared to mere 65% accuracy achieved by implementing SVM on raw data. Therefore, we concluded that models should be trained with lemmatized data, and rest of the models discussed below would follow that.

Multinomial Naive Bayes

Being a probabilistic learning method, Multinomial Naive Bayes assumes the feature probabilities P(Xi|C) are independent given the class C, and it has been established in literature that Naive Bayes classifiers can use any sort of features: URL, email address, dictionaries, network features [7]. The model training time is significantly shorter as compared to the SVM but it still achieved comparable accuracy, at about 77% on Kaggle. This motivated us to try simpler models like Decision Trees.

Ensemble Methods

As alluded to above, we wanted to try ensemble techniques to further improve the classification performance, and we tried Adaboost with about 5000 weak models without much success as we only achieved 60% accuracy. We then decided to use a gradient based boosting method [8] called Gradient Tree Boosting.We used xgboost package to implement the model using 6 decision stumps or tree depth of 6. We started with 20 weak models and achieved 70% accuracy, and we incremented the number of weak models to 300 and observed significant improvement in performance as evident in the table below.

Model Performance

Model	**Kaggle Score**
Linear SVM (Full Dataset)	78.9%
Multinomial Naive Bayes	77%
Adaboost	60%
Gradient Tree Boosting (20 weak models)	70%
Gradient Tree Boosting (300 weak models)	78.8%

Discussion and Conclusion

Overall the results of our models are comparable to the best scores found in the Kaggle competition with our highest score being 78.9% and the best Kaggle score being 83.2%. We placed a larger emphasis on experimenting with representation of the data and used mainly simple linear techniques as we found evidence that these were the most appropriate for the problem.

Supervised PCA results are found to be poorer than expected. In future work it would be ideal to explore these results further and perhaps experiment with various Kernel SPCA methods to better understand why the supervised technique did not match the results found with PCA.

In order to find out the error distribution on cuisines, we took a random sample consisting 25% of the recipes in the training set, and used the best model to predict their category. The error plot (Appendix C, Figure 1) indicates that cuisines with more recipes generally have lower error rate than the minorities.

However, the number of misclassified recipes in all cuisines are fairly close. The overall error rate is mainly determined by the major classes, e.g. Italian, Mexican and Southern US.

A potential way to improve our model is to improve the error rate for minority cuisines, and previous studies suggest threshold adjustment for SVM [9], term-frequency-feature-value (TFFV) [10], and some resampling methods [11, 12]. Nevertheless, due to time constraint, we are not able to fully implement those methods. In an attempt to apply TFFV, we found that the weights look more reasonable since it assigns lower weights to words appear across multiple cuisines, and higher weights to words appear often but in one cuisine. We are also unsure if the improvement in minorities will come at the cost of major cuisines - after all, the overall error rate is more dependent on the performance in major cuisines.

As far as the classification model is concerned, we would try to increase number of weak models in xgboost as we expect a further improvement in performance, also, we would like to improve linear SVM model as discussed in the previous sections. Computational and time limitations prevent us from investigating deep architecture neural networks for this problem, however this would also be a useful area for future work.

Overall, the results suggest that classification of recipes into cuisines based on ingredients alone is possible. This is especially surprising given that the data contains no information on the amount of the ingredients or the preparation methods used. We believe our accuracy to be very reasonable given the nature of the problem. The input data is noisy and required a large amount of preprocessing. The number of classes are large and their distribution is imbalanced further obscuring classification. PCA reveals interesting structure in the data and that many cuisines of similar geographic origin overlapped significantly. Rarer cuisines such as Brazilian or Russian and cuisines with close ethnic origin such as Vietnamese and Thai or Italian and Greek would be worthy of special attention in future work. To conclude, we believe that the methods demonstrated in this report address many of the topics discussed in the course material and could be applied to a variety of real world problems.

Works Cited

1. https://en.wikipedia.org/wiki/Lemmatisation
2. http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
3. Wu, Ho Chung, et al. "Interpreting tf-idf term weights as making relevance decisions." *ACM Transactions on Information Systems (TOIS)* 26.3 (2008): 13.
4. James, Gareth, et al. *An introduction to statistical learning*. New York: springer, 2013.
5. Barshan, Elnaz, et al. "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds." *Pattern Recognition* 44.7 (2011): 1357-1371.
6. Leopold, Edda, and Jörg Kindermann. "Text categorization with support vector machines. How to represent texts in input space?." *Machine Learning 46*.1-3 (2002): 423-444.
7. Dan Jurafsky, "Text Classification and Naïve Bayes", *Natural Language Processing Group, https://web.stanford.edu/class/cs124/lec/naivebayes.pdf*
8. Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of statistics* (2001): 1189-1232.
9. Sun, Aixin, Ee-Peng Lim, and Ying Liu. "On strategies for imbalanced text classification using SVM: A comparative study." *Decision Support Systems 48*.1 (2009): 191-201.
10. Liu, Ying, Han Tong Loh, and Aixin Sun. "Imbalanced text classification: A term weighting approach." *Expert systems with Applications* 36.1 (2009): 690-701.
11. Samanta, Mayukh, and A. H. Welsh. "Bootstrapping for highly unbalanced clustered data." *Computational Statistics & Data Analysis* 59 (2013): 70-81.
12. Yen, Show-Jane, and Yue-Shi Lee. "Cluster-based under-sampling approaches for imbalanced data distributions." *Expert Systems with Applications* 36.3 (2009): 5718-5727.
13. Ogura, Hiroshi, Hiromi Amano, and Masato Kondo. "Comparison of metrics for feature selection in imbalanced text classification." *Expert Systems with Applications* 38.5 (2011): 4978-4989.
14. Dupret, Georges, and Masato Koda. "Bootstrap re-sampling for unbalanced data in supervised learning." *European Journal of Operational Research 134*.1 (2001): 141-156.
15. Forman, George. "A pitfall and solution in multi-class feature selection for text classification." *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004.
16. Uysal, Alper Kursat, and Serkan Gunal. "A novel probabilistic feature selection method for text classification." *Knowledge-Based Systems* 36 (2012): 226-235.

Appendix A

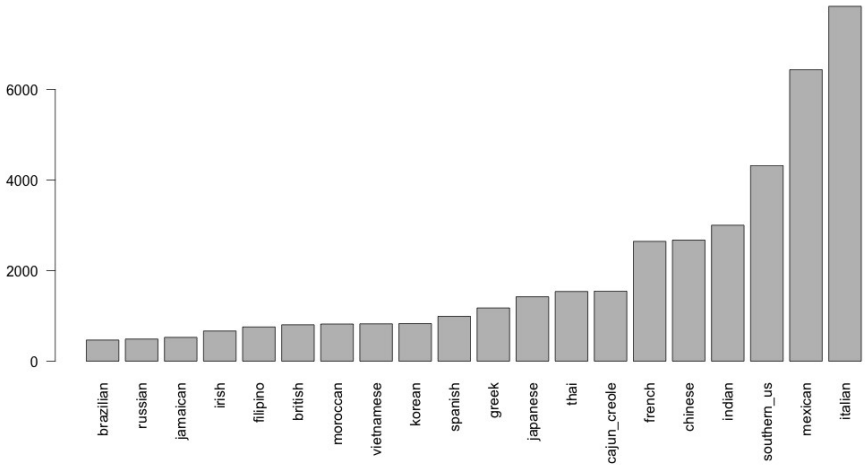

Figure 1. This histogram depicts the number of instances of each cuisine within the training dataset.


```
{
    "id": 39250,
    "cuisine": "italian",
    "ingredients": [
      "olive oil",
      "onions",
      "crushed garlic",
      "dried oregano",
      "green onions",
      "white sugar",
      "dried basil",
      "diced tomatoes"
    ]
}
```

```
{
    "id": 38128,
    "cuisine": "french",
    "ingredients": [
      "sliced tomatoes",
      "cream cheese",
      "pork sausages",
      "shredded cheddar cheese",
      "fresh parsley",
      "diced onions",
      "sour cream",
      "butter",
      "marjoram"
    ]
}
```


Figure 2: A sample of recipes found in the JSON training dataset.

Appendix B

Figure 1: Visualization of ingredients positively (red) and negatively (blue) correlated with the second principal component from regular PCA. Ingredients common in Asian cuisine are negatively correlated, more European-style ingredients are positively correlated.

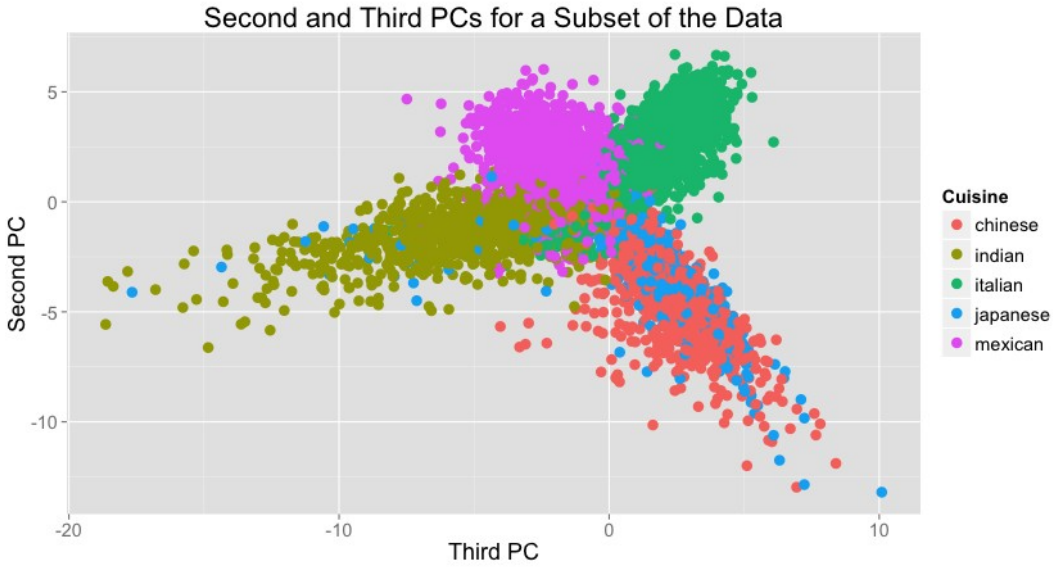

Figure 2: Visualization of a subset of the cuisines and data based on their second and third principal components from regular PCA. The second principal component separates European and Asian cuisines while the third separates Indian and Mexican cuisines from less spicy styles.

Figure 3: Visualization of ingredients positively (red) and negatively (blue) correlated with the third principal component from regular PCA. Spicy ingredients appear more common in the positively correlated terms and less so in the negatively correlated list.

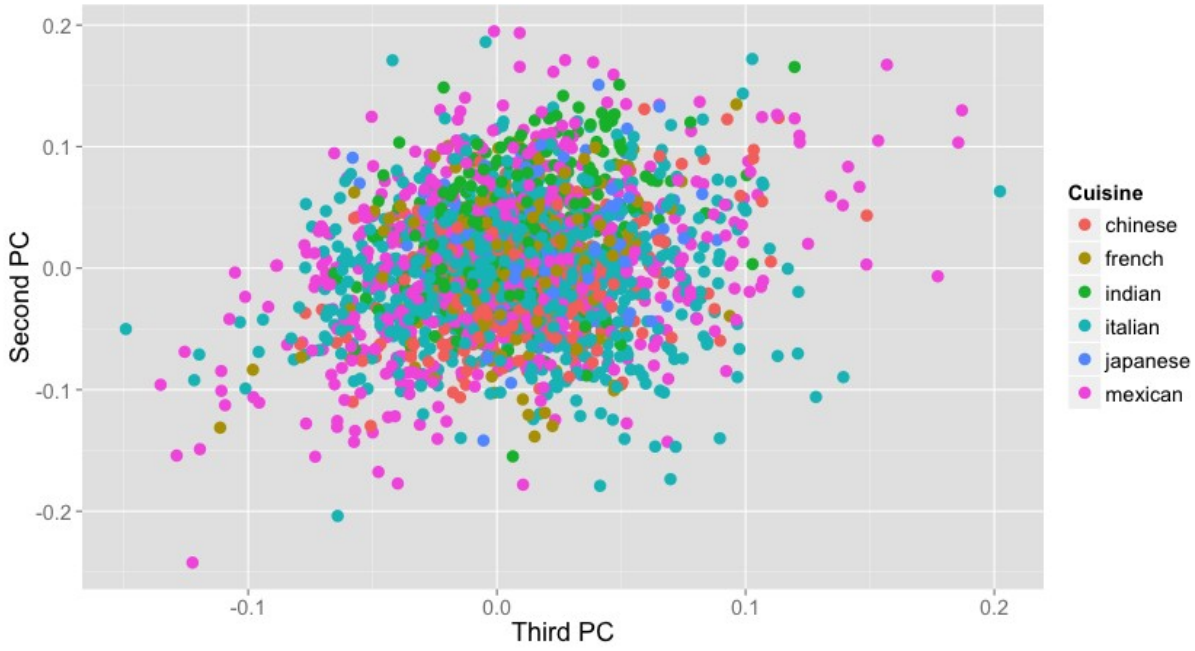

Figure 4: A visualization of a subset of the data mapped onto the second and third component produced by Supervised PCA. Little separation in the cuisines can be seen.

Figure 5: Visualization of ingredients positively (red) and negatively (blue) correlated with the second principal component from SPCA. No specific pattern appears obvious in these weightings.

Appendix C

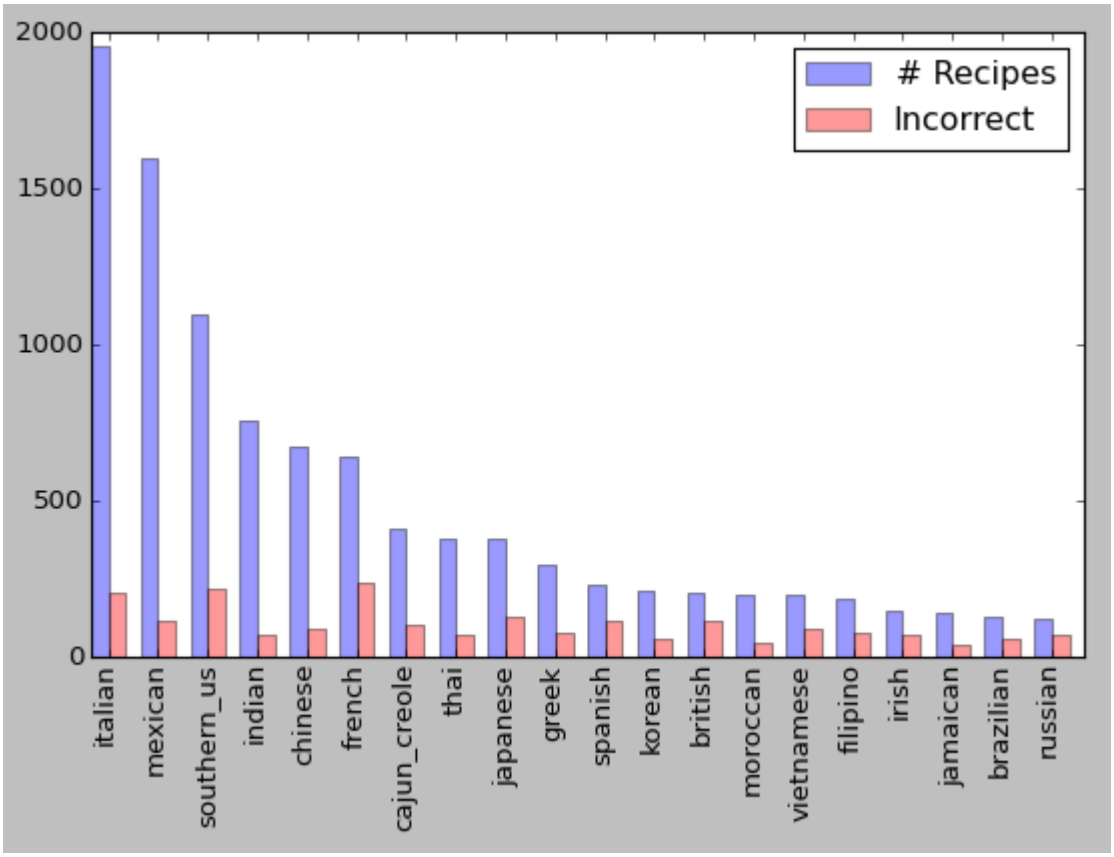

Figure 1: A bar plot showing the distribution of cuisine type within a training set for the full data Linear SVM model (in blue), along with the number of incorrect recipes within each cuisine (in red).