What's Cooking?

Machine Learning Engineer Nanodegree
Capstone Project

Albert Pan
November 23, 2018

1 Definition

1.1 Project Overview

Food plays a major part in any culture around the world. In fact, the geographical characteristics and cultural associations of a region directly influence their cuisines. By investigating the ingredients used in various cuisines, we can gain a better understanding of the geographical and cultural landscape of different regions.

Han Su et. al.[1] has worked on investigating if recipe cuisines could be identified by their ingredients, using data from food.com. They treated each ingredient as a feature and examined the common ingredients for each cuisine. Their study provides good insight on how to approach this project and what results we could expect to achieve.

For this project, we can use machine learning techniques to attain useful predictions with our data. This project will give me an opportunity to work with real-world datasets and to learn about the relation between ingredients and cuisines. This kind of research can be expanded to other fields of study involving text classification.

[1] Su, Han & Lin, Ting-Wei & Li, C.-T & Shan, Man-Kwan & Chang, Janet. (2014). Automatic recipe cuisine classification by ingredients. UbiComp 2014 - Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. 565-570. 10.1145/2638728.2641335.

1.2 Problem Statement

Using the data provided by Yummly[2], the challenge is to predict the cuisine of the dish from its list of ingredients. More specifically, this would be multi-class classfication problem, as we have 20 different cuisines we can predict. We can use a machine learning model that utilizes this data to predict the appropriate cuisine.

My strategy for solving this problem is as follows:

1. Download data from Kaggle
2. Explore data with visualizations
3. Preprocess data and extract features
4. Train and test model
5. Tune hyperparameters

1.3 Metrics

As previously mentioned, our training data seems to be unbalanced, and thus I will be using the F_1-score to evaluate the model. The F_1-score is the weighted average of the precision and recall, and can be expressed mathematically with the following form:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{1}$$

We can also take a look at the precision and recall metrics on their own. Precision can be expressed with the following equation, where $dish_x$ is a dish of a particular cuisine x:

$$precision = \frac{dish_x \, correct}{total \, dish_x} \tag{2}$$

where $dish_x \, correct$ is the number of dishes of cuisine x that are correctly classified as x, and $total \, dish_x$ is the the total number of dishes that were classified as x.

[2] https://www.kaggle.com/c/whats-cooking-kernels-only/data

Recall can expressed with:

$$recall = \frac{dish_x\ correct}{dish_x\ correct + dish_x\ incorrect} \tag{3}$$

where $dish_x\ correct$ is the number of dishes of cuisine x that are correctly classified as x, and $dish_x\ incorrect$ is the number of dishes of cuisine x that are incorrectly classified.

2 Analysis

2.1 Data Exploration

The dataset that I will be using is provided by Yummly, and it consists of two JSON files. The most important file is the *train.json* file, which has 39774 rows of data, and this is the data that we will be using to train our model. The *test.json* file has 9944 rows of data, and this is the data that we will be using to evaluate our model.

id	**cuisine**	**ingredients**
10259	greek	[romaine lettuce, black olives, grape tomatoes, ...]
22213	indian	[water, vegetable oil, wheat, salt]
12734	italian	[chopped tomatoes, fresh basil, garlic, ...]
41995	mexican	[ground cinnamon, fresh cilantro, chili powder, ...]
2941	thai	[sugar, hot chili, asian fish sauce, lime juice]

Table 1: A few examples from our training dataset.

The training data contains three fields, *id*, *cuisine*, and *ingredients*. The *id* field is a unique integer identifier for a particular dish, and is not needed for modeling or analysis. The other two fields, *cuisine* and *ingredients* are very important. *ingredients* consists of the list of ingredients for each dish, while *cuisine* denotes the cuisine. We will be extracting our features from *ingredients* and we will be trying to predict the target class *cuisine*.

2.2 Exploratory Visualization

To better understand the data, we can take a look at the distribution of cuisines in our training data:

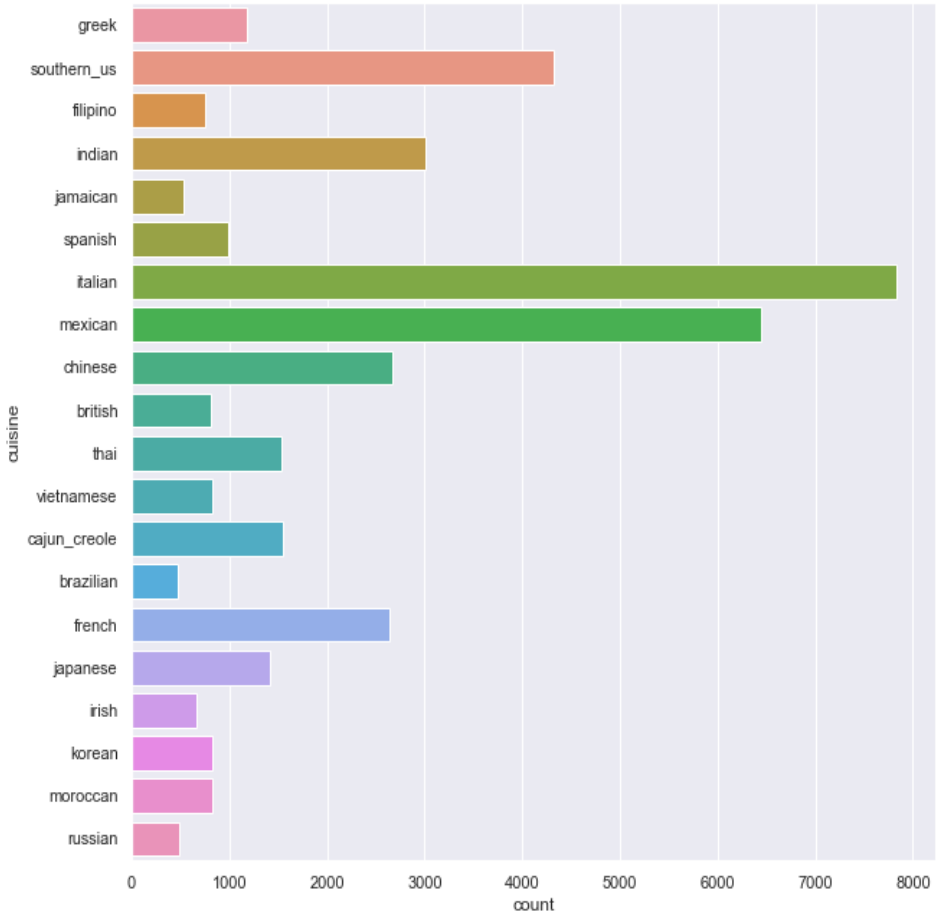

Figure 1: Cuisine distribution in training dataset

We see that our training data is unbalanced, as there are much more "italian"

and "mexican" cuisines that any of the other cuisines. This is why we decided to use the F_1-score as our evaluation metric, as the F_1-score will take into account both precision and recall.

We can also take a closer look at our ingredients. We have 6714 kinds of distinct ingredients, and a good number of them are common to different recipes and cuisines.

2.3 Algorithms and Techniques

This is a classification problem, and so I will be using algorithms and techniques that are suited for classification problems. In particular, I will be using the following models/algorithms:

- Support Vector Machines (LinearSVC[3]): Support vector machines, or SVMs, are a set of supervised learning models that uses decision boundaries for both classification and regressions problems. SVMs construct a hyperplane that allows us to linearly separate data by transforming the feature space. For this problem, I decided to use a linear kernel because it is less computationally expensive and allows us to use multi-class classification.

- Naive Bayes (MultinomialNB[4]): Naive Bayes is a supervised learning algorithm that is based on Bayes' Theorem[5], and it assumes that all of the features are independent. This particular algorithm has been historically used for SPAM detection for its speed and accuracy. For our problem, we will be using MultinomialNB (Multinomial Naive Bayes model), which is a multi-class classifier implementation of Naive Bayes.

- Decision Trees (DecisionTreeClassifier[6]): Decision Trees learn decision rules and generates trees that can be used for both classification and regression problems. Using the input features, it can decide which subtree to follow until it reaches the bottom of the tree, in which it will return its prediction. We will be using the DecisionTreeClassifier from the *scikit-learn* library to perform multi-class classification.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[5] https://en.wikipedia.org/wiki/Bayes%27_theorem
[6] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

- Random Forests (RandomForestClassifier[7]): Random forests are ensemble methods that can used to build models for classification and regression models[8]. They operate by constructing a series of decision trees and outputting a combined prediction from all of the trees. Random Forests aim to correct the decision tree's tendency to overfit to the training data. We will be using the RandomForestClassifier, which is capable of multi-class classification.

- Gradient Boosted Trees (XGBClassifier [9]): Gradient boosting is a technique used in both classification and regression problems. and it builds a prediction model by creating an ensemble of weak learners, usually with decision trees[10]. It builds the model by utilizing the optimization of an arbitrary differentiable loss function. We will be using XGBClassifier from the *xgboost* library.

2.4 Benchmark

For our baseline benchmark, we can use the metric obtained by predicting the most common cuisine in the training and testing datasets. The most common cuisine is *Italian*, and our benchmark model will predict Italian to all recipes. This would give us a benchmark F_1-score of 0.3292.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[8] https://en.wikipedia.org/wiki/Random_forest
[9] https://xgboost.readthedocs.io/en/latest/python/python_api.html
[10]https://xgboost.readthedocs.io/en/latest/tutorials/model.html