Cuisine Type Classifier

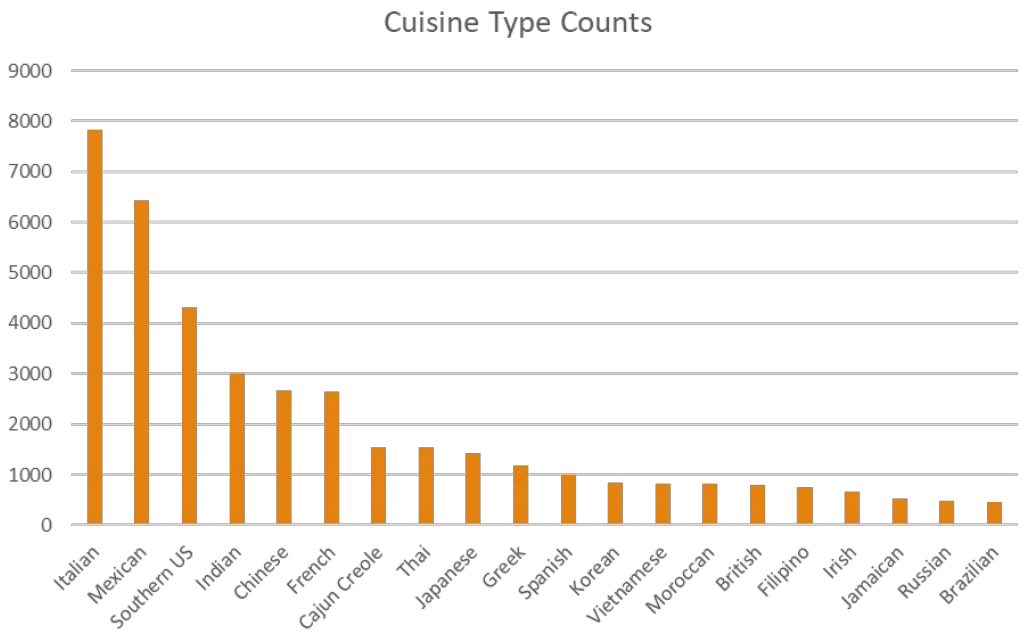
Team Hubble

Robert Gacki
Jinwoong Nam

Introduction

- Our customer runs a website and has hired us to develop code that will classify user submitted recipes into 20 different ethnic categories (Italian, Brazilian, Indian, etc.), with the primary metric being classification accuracy.
- When a user submits their recipe, our code will implement the best algorithm to take the ingredients as input and classify the recipe into one of these 20 categories.
- Users can then search for recipes and filter results based on these different categories.
- The data set contains:
  - 39,774 unique recipes
  - 428,275 ingredients (counting repeats)
  - 6,714 unique ingredients

Data Categories - Overview




```
In [8]: df['cuisine'].value_counts()

Out[8]: italian          7838
        mexican          6438
        southern_us      4320
        indian           3003
        chinese          2673
        french           2646
        cajun_creole     1546
        thai             1539
        japanese         1423
        greek            1175
        spanish           989
        korean            830
        vietnamese        825
        moroccan          821
        british           804
        filipino          755
        irish             667
        jamaican          526
        russian           489
        brazilian         467
```

Data Cleaning

```
# new ingredients list in lowercase, without punctuations, digits, parenthesis, brand names
ing = []
for i in df['ingredients']:
    i = ' '.join(i)
    ing.append(i)

df['ing'] = ing
```

```
import re  #regular expression package
l=[]
for i in df['ing']:

    #Remove punctuations
    i=re.sub(r'[^\w\s]','',i)

    #Remove Digits
    i=re.sub(r"(\d)", "", i)

    #Remove content inside parenthesis
    i=re.sub(r'\([^)]*\)', '', i)

    #Remove Brand Name
    i=re.sub(u'\w*\u2122', '', i)

    #Convert to lowercase
    i=i.lower()

    l.append(i)
df['ing_mod']=l
print(df.head(10))
```

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
5	6602	jamaican	[plain flour, sugar, butter, eggs, fresh ginge...
6	42779	spanish	[olive oil, salt, medium shrimp, pepper, garli...
7	3735	italian	[sugar, pistachio nuts, white almond bark, flo...
8	16903	mexican	[olive oil, purple onion, fresh pineapple, por...
9	12734	italian	[chopped tomatoes, fresh basil, garlic, extra-...

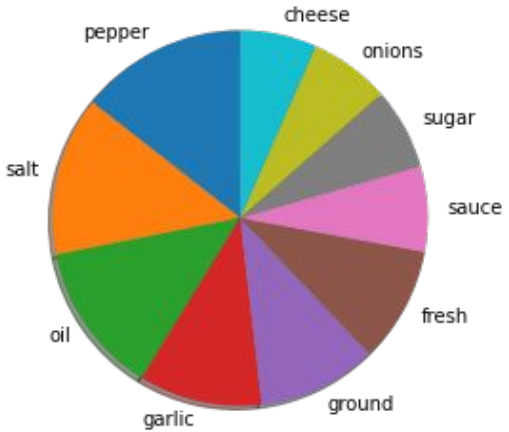
New list of ingredients in lowercase with punctuations, digits, parentheses, and brand names removed.

	ing_mod
0	romaine lettuce black olives grape tomatoes ga...
1	plain flour ground pepper salt tomatoes ground...
2	eggs pepper salt mayonaise cooking oil green c...
3	water vegetable oil wheat salt
4	black pepper shallots cornflour cayenne pepper...
5	plain flour sugar butter eggs fresh ginger roo...
6	olive oil salt medium shrimp pepper garlic cho...
7	sugar pistachio nuts white almond bark flour v...
8	olive oil purple onion fresh pineapple pork po...
9	chopped tomatoes fresh basil garlic extravirgi...

TF-IDF Vectorization

- TF-IDF stand for "Term Frequency – Inverse Document Frequency" and is commonly utilized for text mining and information retrieval.
- Terms are weighted, and these weights are used to evaluate how important each word is to a particular collection of words. In our case, these are recipes (words) and cuisine types (collections of words).
- The *term frequency* assigns a weight to a particular ingredient in a given recipe. It is calculated by dividing by the number of ingredients in the recipe of interest.
  - Example: if a recipe consists of 8 ingredients and carrots is one of them, then the *term frequency* for carrots is $\frac{1}{8} = 0.125$
- The *inverse document frequency* is a measure of how important a particular ingredient is to a given cuisine type. Ingredients that appear often are scaled down in terms of importance, whereas rare ingredients are scaled up and given more weight.
  - Example: if we looked at 1000 different recipes and carrots appeared in 90 of them, then the *inverse document frequency* for carrots would be $log_{10}\left(\frac{1000}{90}\right) = 1.046$
- So, the *TF-IDF* weight for carrots in this particular recipe is $(0.125) \times (1.046) = 0.131$

Top Ingredients by Cuisine Type



All cuisines

Mexican cuisine

Indian cuisine

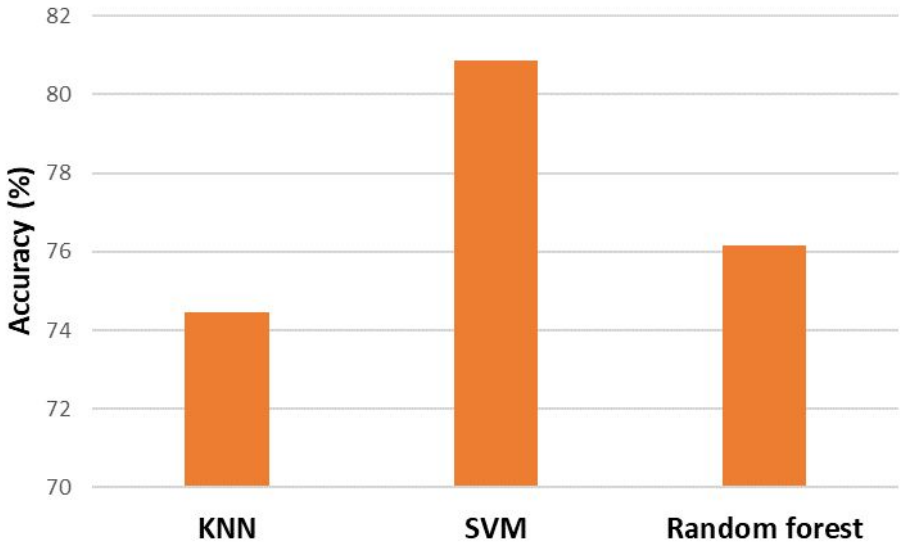
- Unique top ingredient used in each cuisine type is expected to have a higher weight factor

  e.g.) cilantro in Mexican cuisine, cumin in Indian cuisine

Methods and Results

- Method 1: K-Nearest Neighbors
  - Utilized the “KNeighborsClassifier” function in the sklearn.neighbors package
  - # of Neighbors = n in range(30), weights = uniform, algorithm = auto
  - Maximum accuracy achieved = 74.53% for n = 17
- Method 2: Support Vector Machine
  - Utilized “SVC” function in the sklearn.svm package
  - Optimized hyperparameters: C=10, gamma=1, kernel = ‘rbf’
  - Accuracy = 80.87%
- Method 3: Random Forest
  - Utilized “RandomForestClassifier” function in the sklearn.ensemble package
  - Hyperparameters used: bootstrap=False, max_features='sqrt', min_samples_split=3, n_estimators=300
  - Accuracy = 76.14%

Conclusions and Future work



- Conclusions
  - Support vector machine was the best algorithm based on the overall accuracy
- Future work
  - Rigorous work for data cleaning can be applied to remove non-ingredient word (e.g. fresh)
  - Other algorithms (e.g. gradient boost) can be tried

Thank You!