EE0005 Mini Project

Dataset

What's Cooking?

https://www.kaggle.com/c/whats-cooking/

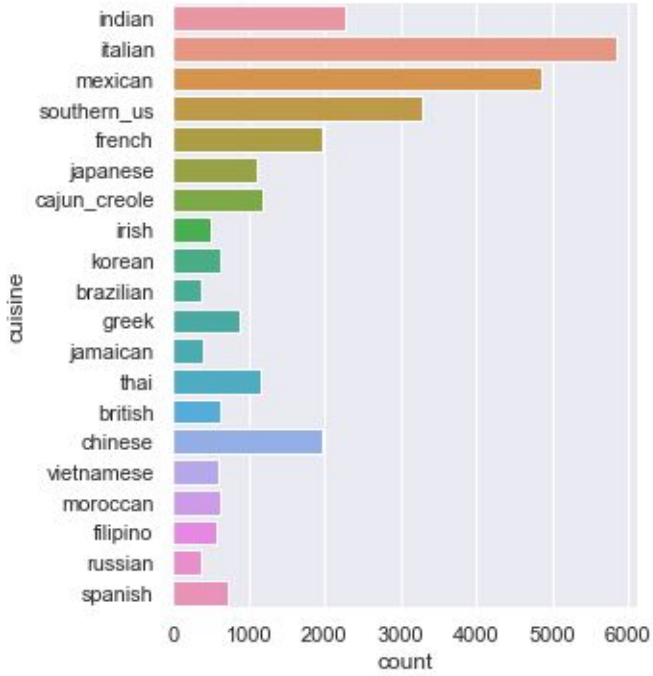
- Classification
- Find 'cuisine' given 'ingredients'
- Only 'train' dataset used

Data Exploration

	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...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

Data Exploration



Preparing the dataset

```
from sklearn.model_selection import train_test_split
trainData , testData = train_test_split(data , train_size = 0.75)
```

Preparing the dataset

	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...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

Preparing the dataset

```
for i in trainData.index:
    trainData.at[i, "ingredients"] = '#'.join(trainData.at[i, "ingredients"])
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace(' ' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace('-' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace('(' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace(')' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace('.' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace(',' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace('%' , '')
    trainData.at[i, "ingredients"] = trainData.at[i, "ingredients"].replace('#' , ' ')
```

Preparing the dataset

ingredients
[milk, butter, garlic cloves, pork, dried thym...
[ground black pepper, salt, lemon juice, pinen...
[pepper, rice, green beans, soy sauce, garlic,...
[red potato, basil leaves, salt, olive oil, ga...
[eggs, butter, onions, vegetable oil, oil, lig...
...
[cinnamon rolls, sugar]
[dried porcini mushrooms, fat free milk, crush...
[dried apricot, vegetable oil, unsweetened coc...
[white rice, black pepper, poultry seasoning, ...
[sugar, flour, soybean paste, dark soy sauce, ...

ingredients
milk butter garlicccloves pork driedthyme allpu...
groundblackpepper salt lemonjuice pinenuts bon...
pepper rice greenbeans soysauce garlic carrots...
redpotato basilleaves salt oliveoil garlic bla...
eggs butter onions vegetableoil oil lightsoysa...
...
cinnamonrolls sugar
driedporcinimushrooms fatfreemilk crushedredpe...
driedapricot vegetableoil unsweetenedcoconutmi...
whiterice blackpepper poultryseasoning salt bu...
sugar flour soybeanpaste darksoysauce porkrind...

Count Vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
matrix = vectorizer.fit_transform(trainData.ingredients)
trainCounts = pd.DataFrame(matrix.toarray(),
                           index=trainData.index,
                           columns=vectorizer.get_feature_names())
```

```
trainCounts['cuisine'] = trainData['cuisine']
trainCounts['id'] = trainData['id']
trainCounts[['id' , 'cuisine']]
```

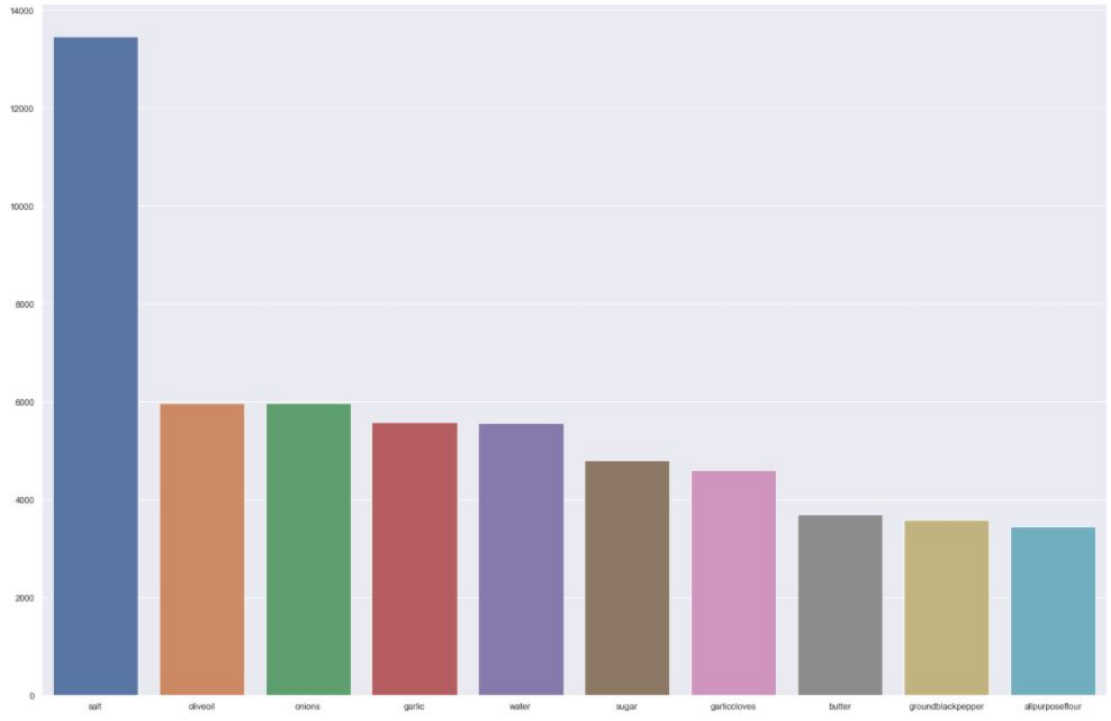
Count Vectorizer

	10ozfrozenchoppedspinach	10ozfrozenchoppedspinachthawedandsqueezeddry	145ozdicedtomatoes	14ozsweetenedcondensedmilk	15ozrefriedbeans	1lowf
33451	0	0	0	0	0	
33839	0	0	0	0	0	
35330	0	0	0	0	0	
4889	0	0	0	0	0	
9657	0	0	0	0	0	
...	...	...	...	...	...	
27021	0	0	0	0	0	
17361	0	0	0	0	0	
29999	0	0	0	0	0	
20316	0	0	0	0	0	
19310	0	0	0	0	0	

29830 rows × 6233 columns

Data Exploration

salt	13472
oliveoil	5980
onions	5973
garlic	5585
water	5572
sugar	4804
garliccloves	4604
butter	3691
groundblackpepper	3577
allpurposeflour	3450



Data Exploration

	indian
salt	1441
onions	905
garammasala	652
water	612
garlic	545
groundturmeric	541
cuminseed	525
groundcumin	507
vegetableoil	454
oil	415

	spanish
salt	339
oliveoil	293
garliccloves	213
onions	192
extravirginoliveoil	178
water	117
tomatoes	109
groundblackpepper	108
redbellpepper	100
pepper	98

	chinese
soysauce	982
cornstarch	683
sesameoil	667
salt	666
sugar	612
garlic	576
water	554
greenonions	474
vegetableoil	461
scallions	447

Preparing the dataset

```
trainX = trainCounts[top1000.index]
trainY = trainCounts['cuisine']
testX = testCounts[top1000.index]
testY = testCounts['cuisine']
```

```
top1000 = sum.sort_values(by=0 , ascending = False).head(1000)
top1000.index
```

Multinomial Naive Baye's

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(trainX , trainY)
```

sklearn.naive_bayes.MultinomialNB

class sklearn.naive_bayes.**MultinomialNB**(*, *alpha=1.0*, *fit_prior=True*, *class_prior=None*) [source]

Naive Bayes classifier for multinomial models

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Results

```
from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(trainY, predict))
```

```
Accuracy: 0.7469661414683205
```

```
predict = clf.predict(testX)
print("Accuracy:",metrics.accuracy_score(testY, predict))
```

```
Accuracy: 0.7202333065164923
```

Thank You