What’s Cooking

Eka Renardi

Kaggle Competition

Problem

Classify ‘cuisine’ based on ‘ingredients’

Data

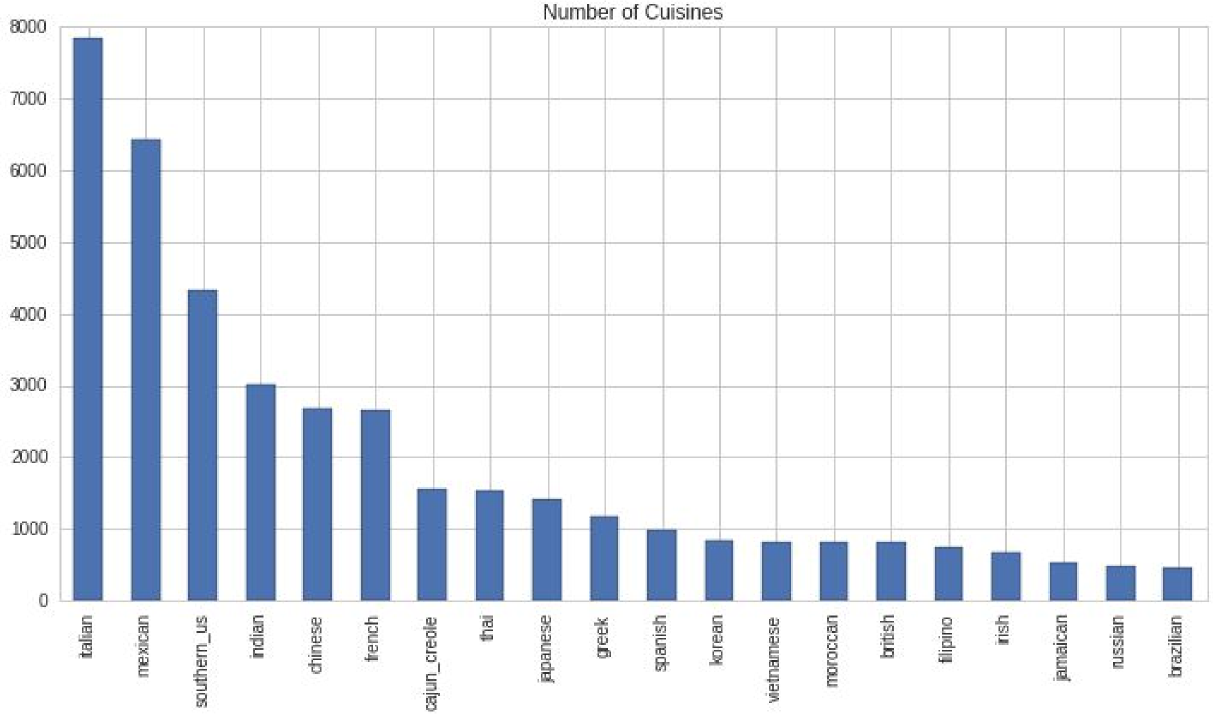
- ➔ Provided by Yummly
- ➔ Text data
- ➔ 39,000+ observations
- ➔ 3 Columns:
  - ◆ id: numeric, unique identifier
  - ◆ cuisine: text, 20 classes
  - ◆ ingredients: list of text

Sample Data


```json
{
    "id": 25693,
    "cuisine": "southern_us",
    "ingredients": [
      "plain flour",
      "ground pepper",
      "salt",
      "tomatoes",
      "ground black pepper",
      "thyme",
      "eggs",
      "green tomatoes",
      "yellow corn meal",
      "milk",
      "vegetable oil"
    ]
}
```

Cuisine

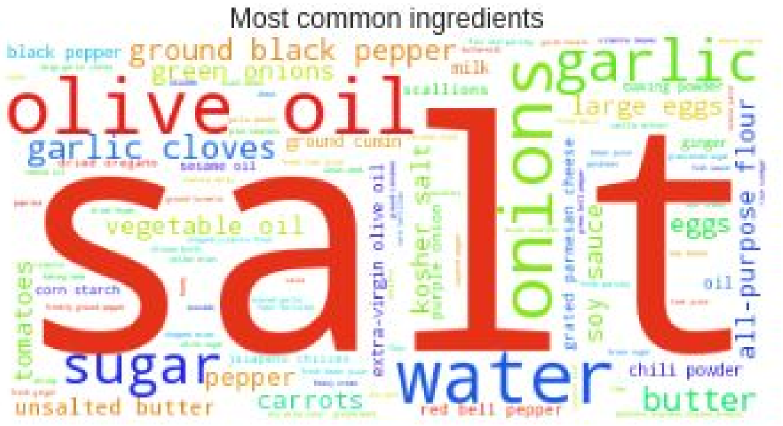
```
('shape:', (39774, 3))
('unique cuisine count:', 20)
```

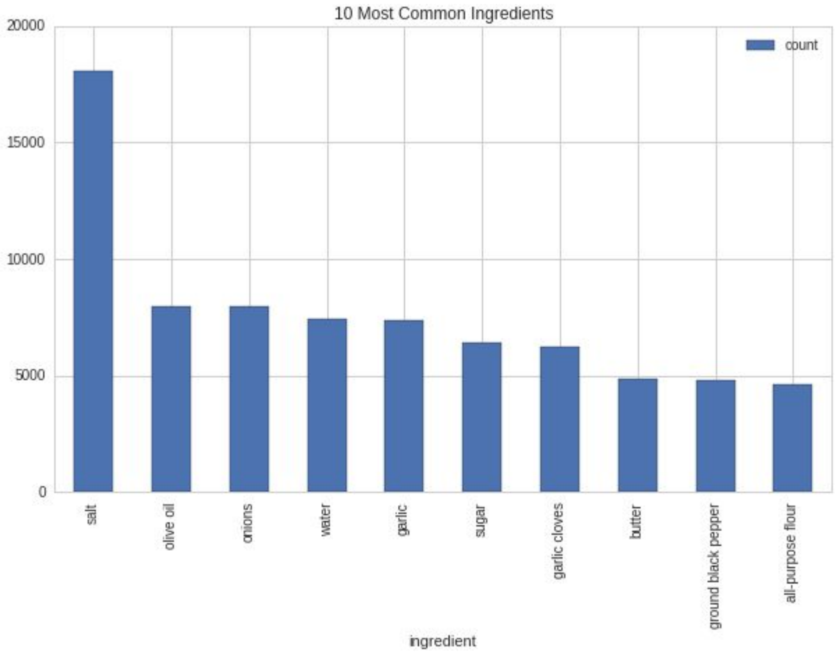


1. brazilian
2. british
3. cajun_creole
4. chinese
5. filipino
6. french
7. greek
8. indian
9. irish
10. italian
11. jamaican
12. japanese
13. korean
14. mexican
15. moroccan
16. russian
17. southern_us
18. spanish
19. thai
20. vietnamese

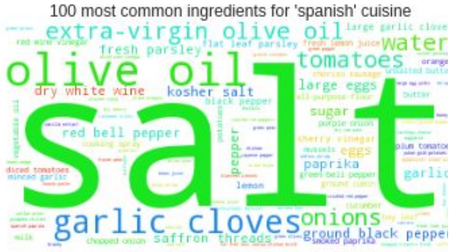
Ingredients

What's the most common ingredients





Ingredients (Europe)

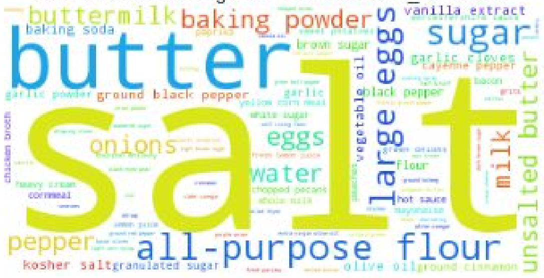


Ingredients (Americas)

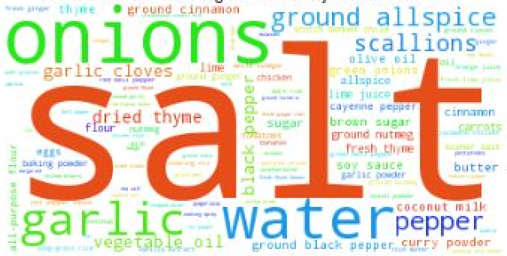
100 most common ingredients for 'brazilian' cuisine

100 most common ingredients for 'cajun_creole' cuisine

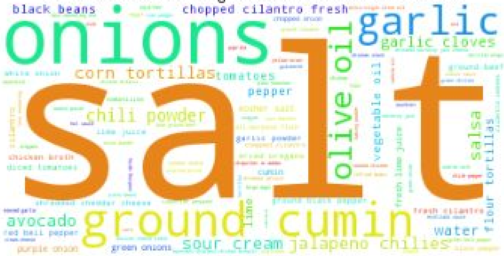
100 most common ingredients for 'southern_us' cuisine



100 most common ingredients for 'jamaican' cuisine



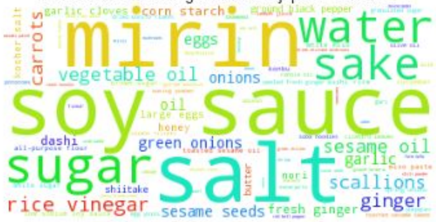
100 most common ingredients for 'mexican' cuisine



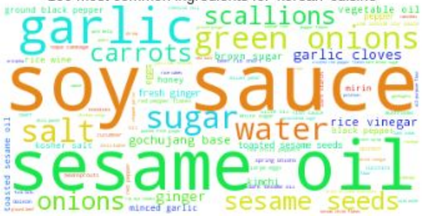
Ingredients (Asia)

100 most common ingredients for 'chinese' cuisine

100 most common ingredients for 'japanese' cuisine



100 most common ingredients for 'korean' cuisine



100 most common ingredients for 'thai' cuisine

100 most common ingredients for 'filipino' cuisine

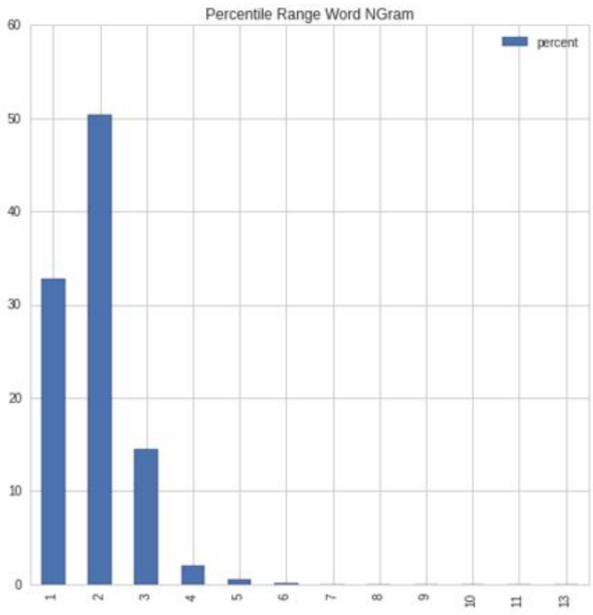
100 most common ingredients for 'vietnamese' cuisine

100 most common ingredients for 'indian' cuisine

Ingredients <-> Cuisine Relationship

Ingredients (Word Ngram)

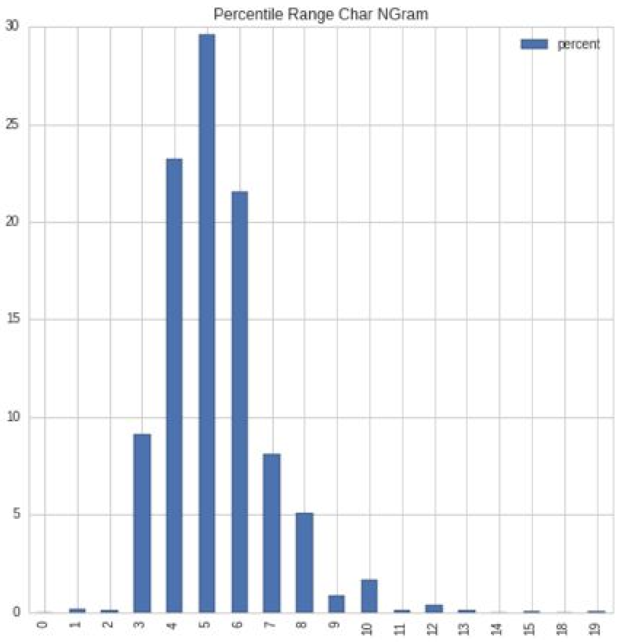
words in an ingredient (e.g. Salt = 1 word, Black Pepper = 2 words)



	percent
1	32.715556
2	50.361037
3	14.443825
4	1.970376
5	0.437395
6	0.047330
7	0.011191
8	0.006062
9	0.001632
10	0.000466
11	0.004197
13	0.000933

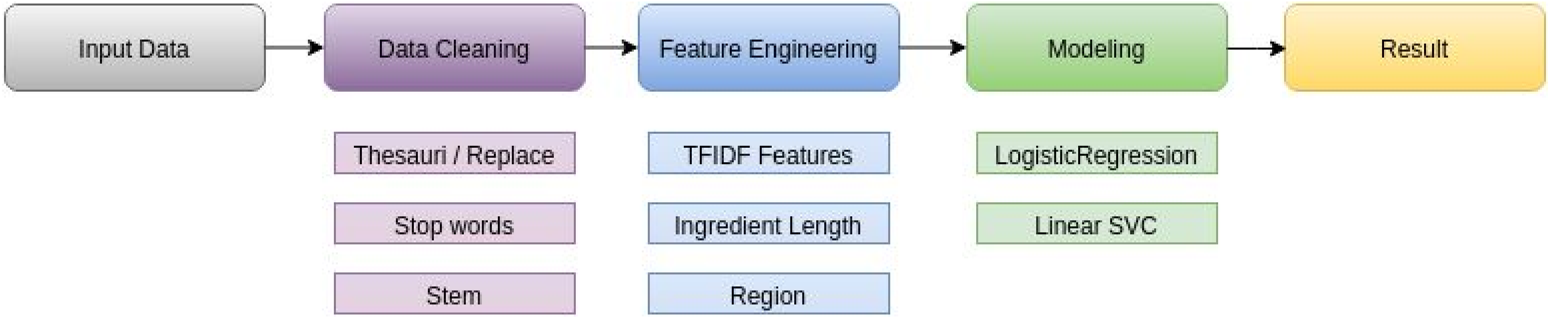
Ingredients (Char NGram)

characters per word (e.g. Salt = 4 chars)



	percent
0	0.001247
1	0.160310
2	0.083645
3	9.139650
4	23.213375
5	29.595723
6	21.562160
7	8.104493
8	5.108222
9	0.843558
10	1.628777
11	0.091997
12	0.355898
13	0.093618
14	0.004363
15	0.004488
18	0.000374
19	0.008103

Model Building



Data Cleaning

1. Word replacement (e.g. Philadelphia Cream Cheese -> cream cheese)
    a. Read from a text file (thesauri.txt)
2. Stop words (e.g. and, &)
    a. Read from a text file (stopwords.txt)
3. Stemming (e.g. apples -> apple)

Data Cleaning Files

stopwords.txt | thesauri.txt

```
a
a's
able
about
above
according
accordingly
across
actually
after
afterwards
again
against
ain't
all
allow
allows
```

stopwords.txt | thesauri.txt

```
Bisquick Baking Mix,baking mix
Bertolli® Classico Olive Oil,classico olive oil
Bertolli® Alfredo Sauce,alfredo sauce
Jonshonville® Cajun Style Chicken Sausage,cajun style chicken sausage
Old El Paso Enchilada Sauce,enchilada sauce
Old El Paso Green Chiles,green chiles
Old El Paso™ refried beans,refried beans
Old El Paso™ chopped green chiles,chopped green chiles
Old El Paso™ taco seasoning mix,taco seasonig mix
Old El Paso™ Thick 'n Chunky salsa,thick chunky salsa
Old El Paso Flour Tortillas,flour tortillas
Old El Paso™ mild red enchilada sauce",mild red enchilada sauce
Mexican cheese blend,cheese blend
Pillsbury™ Refrigerated Crescent Dinner Rolls,refrigerated crescent dinn
```

Feature Engineering

1. TFIDF features: ngram, max_features
2. Ingredient length
3. Region encoding (work in progres…)

Modeling

1. Logistic Regression. Classic classification algorithm.
2. Linear SVC. Large dimensions.

Model Building

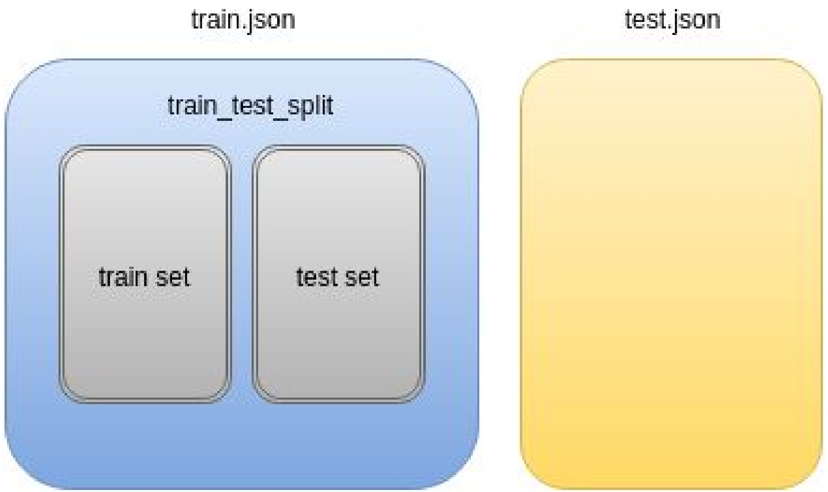
Use cross validation, and GridSearchCV with 5 folds.

Uses ‘Pipeline’ heavily

Scoring on ‘Accuracy’

Test on entire train.json

Default train test split



First Model

```
# base models
models = [
    ('nb',
        Pipeline([('vect', CountVectorizer(strip_accents='unicode')),
                  ('clf', MultinomialNB())
                 ])
    ),
    ('logistic',
        Pipeline([('vect', TfidfVectorizer(strip_accents='unicode', tokenizer=Tokenizer())),
                  ('clf', LogisticRegression(C=1e9))
                 ])
    ),
]
X = df['ingredients_all']
predLst = cross_val_models(models, X, y, 5)
predDf = pd.DataFrame.from_dict(predLst)
predDf
```

```
Cross_val nb...0.723
Cross_val logistic...0.653
```

	name	score	sem
0	nb	0.723085	0.002444
1	logistic	0.652690	0.004202

Data Cleaning Effect on Models

- With no data cleaning, accuracy is 0.734
- With data cleaning, accuracy is 0.743

TFIDF Tuning (Word Ngram)

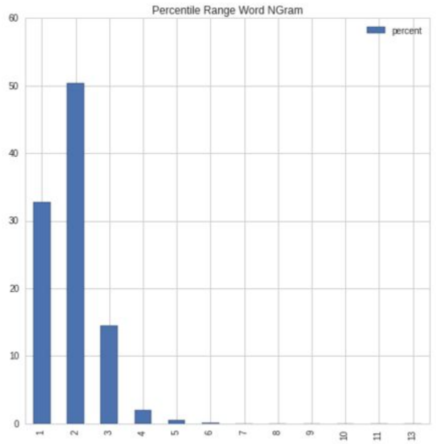
```
# the word ngram range
word_ngram_range = [(1,3), (1, 5)]
# create the model
model = Pipeline([('tfidf', TfidfVectorizer(strip_accents='unicode', analyzer='word')),
                  ('clf', LogisticRegression(C=1e9))
        ])
param_grid = {
    'tfidf__ngram_range': word_ngram_range,
}
# cross validate using grid search
X = df['ingredients_string']
word_ngram_results = grid_search_models('word_ngram', model, param_grid, X, y, 5)
word_ngram_results
```

```
Grid_search word_ngram...0.783

{'best_params': Pipeline(steps=[('tfidf', TfidfVectorizer(analyzer='word', binary=False, dec
ode_error=u'strict',
         dtype=<type 'numpy.int64'>, encoding=u'utf-8', input=u'content',
         lowercase=True, max_df=1.0, max_features=None, min_df=1,
         ngram_range=(1, 3), norm=u'l2', preprocessor=None, smooth_idf=True...class='ovr', p
enalty='l2', random_state=None,
           solver='liblinear', tol=0.0001, verbose=0))]),
 'name': 'word_ngram',
 'score': 0.78335093277015133,
 'scores': [mean: 0.78335, std: 0.00359, params: {'tfidf__ngram_range': (1, 3)},
  mean: 0.78071, std: 0.00509, params: {'tfidf__ngram_range': (1, 5)}]}
```

```
wpd = get_grid_scores_pd(word_ngram_results)
wpd
```

	name	mean score	scores
0	{u'tfidf__ngram_range': (1, 3)}	0.783351	[0.782089927154, 0.787132445338, 0.78192559074...
1	{u'tfidf__ngram_range': (1, 5)}	0.780711	[0.779201205727, 0.787760743906, 0.77790346908...



TFIDF Tuning

- char ngram: accuracy 0.772, no improvement
- max_features: accuracy: 0.471, no improvement.

Adding Ingredient Length

No improvement

```
# model additional feature (ingredient length) and logistic regression
model = Pipeline([
            ('ingredients', IngredientExtractor()),

            ('union', FeatureUnion(
                [
                    # adding ingredient length feature
                    ('ingredient_length', Pipeline([
                      ('extract', ItemSelector(key='ingredient_length')),
                      ('tfidf', TfidfVectorizer()),
                    ])),

                    # adding ingredient text feature
                    ('txt', Pipeline([
                      ('extract', ItemSelector(key='txt')),
                      ('tfidf', TfidfVectorizer(strip_accents='unicode',
                                                 analyzer='word', ngram_range=(1,3))),
                    ])),
                ],

            )),

            # using logistic classifier
            ('clf', LogisticRegression(C=1e9))
        ])
param_grid = {}
# cross validate using grid search
X = df['ingredients_all']
ful_results = grid_search_models('feature_union', model, param_grid, X, y, 5)
ful_results
```

```
Grid_search feature_union...0.782
```

Linear SVC (base)

Best model so far, with accuracy of 0.786

```
# let's try simple model with no feature union
model = Pipeline([
            # using the best parameters
            ('tfidf', TfidfVectorizer(strip_accents='unicode',
                                      analyzer='word', ngram_range=(1,3)
                                     )),
            # using linear svc
            ('clf', OneVsRestClassifier(LinearSVC(random_state=1)))
        ])
param_grid = {}
# cross validate using grid search
X = df['ingredients_all']
simple_svc_results = grid_search_models('simple_svc', model, param_grid, X, y, 5)
simple_svc_results
```

```
Grid_search simple_svc...0.786
```

Prediction Result

Accuracy on training set:
0.998759637948
Accuracy on testing set:
0.784794851167
Classification Report:

	precision	recall	f1-score	support
0	0.74	0.52	0.61	102
1	0.56	0.39	0.46	197
2	0.73	0.72	0.72	356
3	0.78	0.87	0.82	684
4	0.74	0.52	0.61	181
5	0.59	0.62	0.60	656
6	0.79	0.70	0.74	305
7	0.86	0.92	0.89	740
8	0.66	0.42	0.51	173
9	0.80	0.89	0.84	1994
10	0.78	0.70	0.74	132
11	0.86	0.66	0.75	340
12	0.87	0.75	0.81	206
13	0.90	0.93	0.91	1599
14	0.79	0.73	0.76	203
15	0.68	0.41	0.51	120
16	0.72	0.79	0.75	1115
17	0.68	0.46	0.55	239
18	0.82	0.80	0.81	419
19	0.71	0.57	0.63	183
avg / total	0.78	0.78	0.78	9944

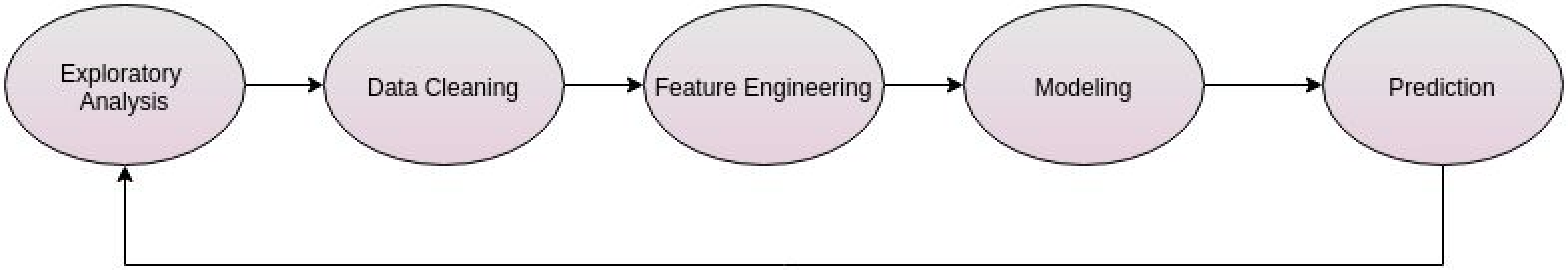
	0					5					10					15				
0	53	0	4	0	3	4	0	1	1	8	1	0	0	11	0	2	9	3	2	0
	0	76	1	1	1	29	1	5	8	18	2	0	0	3	0	3	47	2	0	0
	0	1	255	0	0	10	0	0	0	23	1	0	0	12	2	0	49	2	0	1
	0	2	6	595	8	0	0	1	0	9	0	14	10	6	1	0	8	0	16	8
	4	1	2	27	95	4	0	6	2	11	0	3	1	3	0	0	10	1	9	2
5	0	8	5	3	3	404	6	2	7	139	1	0	0	8	4	7	53	6	0	0
	0	1	0	1	0	8	213	2	0	67	1	1	0	3	2	2	1	3	0	0
	1	2	2	3	0	5	6	683	0	5	0	0	0	7	13	0	10	0	3	0
	0	12	0	3	0	24	1	3	73	14	3	0	0	2	1	2	33	2	0	0
	1	4	9	3	0	95	16	6	5	1770	4	0	1	21	4	1	43	11	0	0
10	1	0	2	1	1	5	0	9	0	1	93	0	0	5	1	0	8	1	4	0
	0	5	1	45	1	6	1	30	1	6	1	224	7	1	0	1	6	0	3	1
	0	0	1	27	1	0	0	0	0	5	0	9	155	3	1	0	2	1	0	1
	4	1	6	4	2	11	4	7	0	27	2	4	0	1482	0	2	33	8	1	1
	0	0	0	0	0	6	7	11	0	13	0	0	0	8	149	0	4	4	1	0
15	0	5	1	0	2	14	3	1	7	12	1	1	0	4	2	49	18	0	0	0
	0	16	53	6	3	31	7	8	7	38	5	0	0	44	3	2	885	6	1	0
	3	1	2	0	2	25	5	1	0	49	1	0	0	24	5	1	10	109	0	1
	2	0	0	21	3	1	1	13	0	3	3	2	1	2	0	0	3	0	337	27
	3	0	0	20	3	1	0	2	0	3	0	2	3	6	0	0	2	2	32	104

true label

predicted label

{u'brazilian': 0,
u'british': 1,
u'cajun_creole': 2,
u'chinese': 3,
u'filipino': 4,
u'french': 5,
u'greek': 6,
u'indian': 7,
u'irish': 8,
u'italian': 9,
u'jamaican': 10,
u'japanese': 11,
u'korean': 12,
u'mexican': 13,
u'moroccan': 14,
u'russian': 15,
u'southern_us': 16,
u'spanish': 17,
u'thai': 18,
u'vietnamese': 19}

Conclusion



Iterative approach to problem solving

Feature engineering is time consuming, but may improve the score.

Attempt different algorithms.

What's Next

- More feature engineering
- Ensembling