Yummly Coding Challenge Report

Installation:

- *conda create -n yummlyenv python=3.6 anaconda*
- *source activate yummlyenv*
- conda install pip
- pip install -r requirement.txt

How to run:

- *python3 yummly_cuisine_classification.py cuisine.unlabeled.json.gz*

    1. It will ask for your choice of method:
       1 for logistic regression, or 2 for random forests, or 3 for Text CNN.

    2. Based on the choice, it will load the selected model.

    3. After it will perform data loading and prediction of cuisines.

    4. At the end, it will generate a CSV on your local machine.

Related Files:

1) Yummly_Data_Training.ipynb (Data cleaning and Model training code)
2) yummly_cuisine_classification.py (cuisine prediction)
3) vectorizer.pk (Vectorization of all training text)
4) yummly_lg.pkl (Logistic regression model file)
5) yummly_rf.pkl (Random forests model file)
6) labels.json (Labels JSON file for mapping)
7) ./trained_model_1517178549/ (Text CNN model folder)
    1. checkpoints (latest checkpoint)
    2. vocab.pickle (Vocab file)

Data:

The dataset consists of a training set, a dev set, and an unlabeled set. The data is in a simple JSON format. Each recipe includes an id, a list of ingredients, and an optional cuisine. We have 20 different types of cuisines. To ease things, we have directly put JSON data to Pandas. After putting, data pre-processing is performed i.e. lemmatization, removing special characters/digits. Before giving these data as an input to model, we have to vectorize all text input.

Model Building:

Now, we have our vectorized data ready to do some analysis. And we can use 'cuisine' column as our labels. Hence, we have 20 classes to be predicted. First, we are going to start with basic algorithm. So, we could visualize data and decide which algorithm should work for this problem. Here, three classification algorithms have been implemented:

1) Logistic Regression
2) Random Forests
3) Text CNN

For all algorithms, cross validations have been performed. Moreover, I have tried different hyper-parameter combinations and after training, I am storing model file. (So, while doing cuisine predictions, we don't need to train model every time. We can directly load model file and use it)

Algorithm Performance on Dev set:

1) *Logistic Regression:*

** Accuracy for Dev Set **0.6691389063482087**... **

Report created for Logistic Regression model:

	precision	recall	f1-score	support
brazilian	0.75	0.55	0.63	97
british	0.62	0.47	0.54	159
cajun_creole	0.77	0.70	0.73	304
chinese	0.01	0.01	0.01	532
filipino	0.75	0.57	0.65	132
french	0.61	0.66	0.63	527
greek	0.78	0.72	0.74	226
indian	0.00	0.00	0.00	592
irish	0.64	0.44	0.52	133
italian	0.83	0.90	0.86	1575
jamaican	0.84	0.68	0.75	126
japanese	0.80	0.73	0.76	280
korean	0.82	0.81	0.82	154
mexican	0.89	0.92	0.91	1291
moroccan	0.76	0.73	0.75	146
russian	0.60	0.39	0.48	89
southern_us	0.72	0.82	0.77	871
spanish	0.66	0.51	0.58	222
thai	0.83	0.75	0.79	350
vietnamese	0.65	0.55	0.60	149
avg / total	0.67	0.67	0.67	7955

- Training Accuracy of Logistic regression is **0.868.**
- Validation Accuracy of Logistic regression is **0.789.**

2) ***Random Forests:***

** Accuracy for Dev Set 0.**613073538654934**... **

Report created for Random Forests model:

	precision	recall	f1-score	support
brazilian	0.82	0.43	0.57	97
british	0.75	0.28	0.40	159
cajun_creole	0.79	0.62	0.70	304
chinese	0.00	0.01	0.01	532
filipino	0.75	0.51	0.61	132
french	0.56	0.48	0.51	527
greek	0.83	0.53	0.65	226
indian	0.00	0.00	0.00	592
irish	0.68	0.29	0.40	133
italian	0.69	0.91	0.78	1575
jamaican	0.92	0.44	0.60	126
japanese	0.84	0.66	0.74	280
korean	0.86	0.63	0.73	154
mexican	0.84	0.92	0.88	1291
moroccan	0.89	0.62	0.73	146
russian	0.65	0.12	0.21	89
southern_us	0.63	0.77	0.69	871
spanish	0.82	0.27	0.40	222
thai	0.82	0.75	0.78	350
vietnamese	0.80	0.42	0.55	149
avg / total	0.64	0.61	0.61	7955

- Training Accuracy of Random Forests is **0.9996.**
- Validation Accuracy of Random Forests is **0.748.**

*3) **Text CNN:***

** Accuracy for Dev Set **0.6334380892520427**... **

Report created for Text CNN model:

	precision	recall	f1-score	support
brazilian	0.72	0.40	0.52	97
british	0.44	0.31	0.37	159
cajun_creole	0.79	0.61	0.69	304
chinese	0.01	0.01	0.01	532
filipino	0.61	0.43	0.50	132
french	0.53	0.62	0.57	527
greek	0.78	0.64	0.70	226
indian	0.00	0.01	0.00	592
irish	0.64	0.38	0.48	133
italian	0.80	0.89	0.84	1575
jamaican	0.87	0.60	0.71	126
japanese	0.84	0.67	0.74	280
korean	0.75	0.67	0.71	154
mexican	0.89	0.92	0.90	1291
moroccan	0.74	0.69	0.71	146
russian	0.72	0.15	0.24	89
southern_us	0.65	0.80	0.72	871
spanish	0.66	0.33	0.44	222
thai	0.77	0.78	0.78	350
vietnamese	0.58	0.44	0.50	149
avg / total	0.64	0.63	0.63	7955

- Training Accuracy of Random Forests is **0.777.**
- Validation Accuracy of Random Forests is **0.740.**