Machine Learning Engineer Nanodegree

Capstone Project Report

Sai Marasani
October 3, 2018

Project Overview

Picture yourself strolling through your local, open-air market... What do you see? What do you smell? What will you make for dinner tonight?

If you're in Northern California, you'll be walking past the inevitable bushels of leafy greens, spiked with dark purple kale and the bright pinks and yellows of chard. Across the world in South Korea, mounds of bright red kimchi greet you, while the smell of the sea draws your attention to squids squirming nearby. India's market is perhaps the most colorful, awash in the rich hues and aromas of dozens of spices: turmeric, star anise, poppy seeds, and garam masala as far as the eye can see. Some of our strongest geographic and cultural associations are tied to a region's local foods.

In this project, I have developed supervised machine learning model which will predict the cuisine type provided with ingredients as input.

Problem Statement:

This is Multi Class Classification problem where there are 20 possible responses from the features we pass as input. Final goal of this project is to predict cuisine type from the ingredients list provided. For out of sample performances I have considered Cross-validation scores. The ultimate goal is to accurately predict the cuisine type provided with ingredients as input. We will use accuracy measurement as a model performance metrics which gives us a percentage based on all correct and incorrect classifications of our model on the test data.

Tasks involved are:

1. Download and preprocess the JSON format data
2. Build and train the ML model to classify cuisine types
3. Based on accuracy scores on test data, improve the model

Metrics:

As the problem comes under multi classification problem, Accuracy will be my metric which takes true positives and true negatives with equal weight.

Accuracy measures how often the classifier makes the correct prediction. It's the ratio of the number of correct predictions to the total number of predictions (the number of test data points).

Accuracy = true positives + true negatives / total dataset size

Analysis

Data Exploration:

"Yummly" has provided the data set which consists of 39774 samples which consists of list of ingredients which are going to be our features and cuisine column which is going to be our response variable. Data looks like below picture
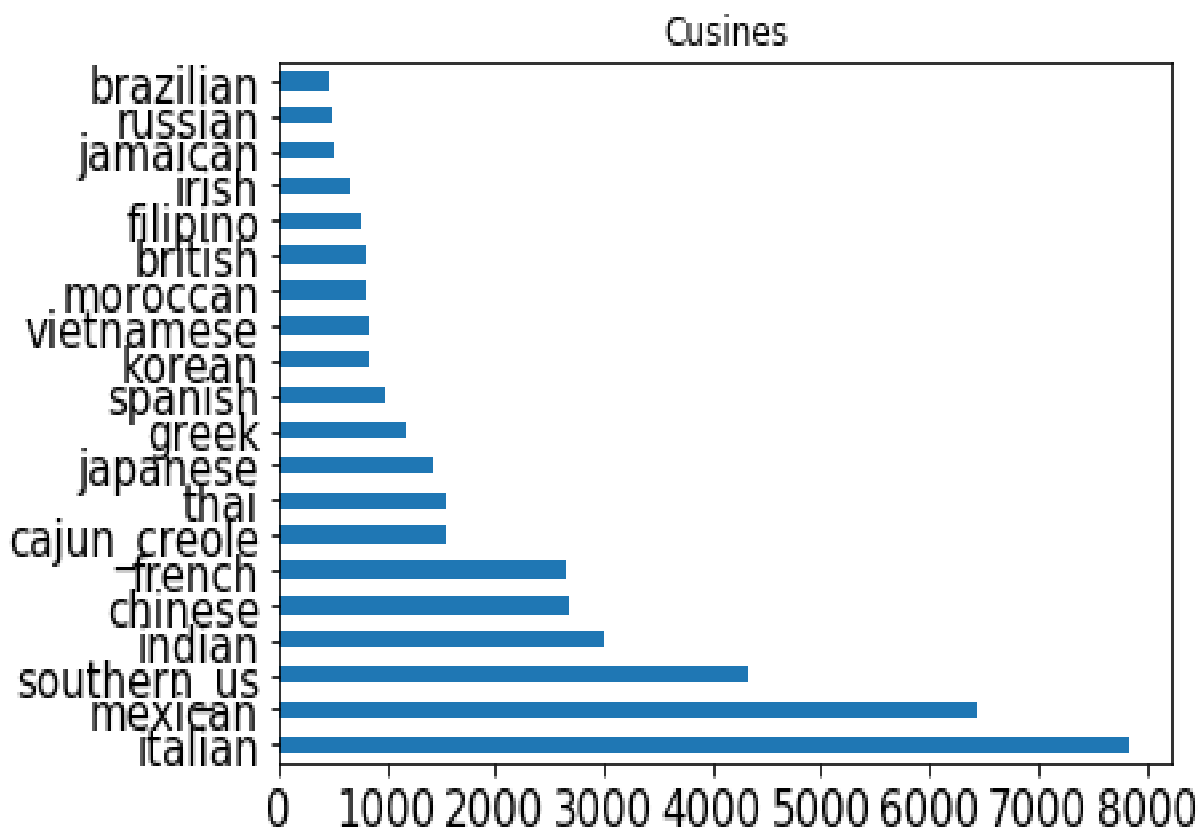
	cuisine	id	ingredients
0	greek	10259	[romaine lettuce, black olives, grape tomatoes...
1	southern_us	25693	[plain flour, ground pepper, salt, tomatoes, g...
2	filipino	20130	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	indian	22213	[water, vegetable oil, wheat, salt]
4	indian	13162	[black pepper, shallots, cornflour, cayenne pe...

We have 20 different kinds of cuisines in the data set with respective value counts as shown below:

```
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
```

Exploratory Visualization:

Using value_counts function I have obtained the count of sample data we have for each cuisine in our training data and following plot shows the distribution of cuisines

Inorder to check the variation between classes lets describe the number of ingredients by displaying count, mean, std, min, max.

```
train.groupby('cuisine').num_ingredients.describe()
```

Out[14]:

cuisine	count	mean	std	min	25%	50%	75%	max
brazilian	467.0	9.520343	5.555139	2.0	5.0	9.0	13.0	59.0
british	804.0	9.708955	4.165011	2.0	7.0	9.0	12.0	30.0
cajun_creole	1546.0	12.617076	4.611601	2.0	9.0	12.0	16.0	31.0
chinese	2673.0	11.982791	4.042125	2.0	9.0	12.0	14.0	38.0
filipino	755.0	10.000000	3.855135	2.0	7.0	10.0	12.0	38.0
french	2646.0	9.817838	4.144744	1.0	7.0	9.0	12.0	31.0
greek	1175.0	10.182128	3.729461	1.0	7.0	10.0	12.0	27.0
indian	3003.0	12.705961	5.016806	1.0	9.0	12.0	16.0	49.0
irish	667.0	9.299850	3.700505	2.0	7.0	9.0	12.0	27.0
italian	7838.0	9.909033	3.806708	1.0	7.0	10.0	12.0	65.0
jamaican	526.0	12.214829	4.763897	2.0	9.0	12.0	15.0	35.0
japanese	1423.0	9.735067	4.245882	1.0	7.0	9.0	12.0	34.0
korean	830.0	11.284337	3.878880	2.0	9.0	11.0	14.0	29.0
mexican	6438.0	10.877446	4.660183	1.0	7.0	10.0	14.0	52.0
moroccan	821.0	12.909866	4.799813	2.0	9.0	13.0	16.0	31.0
russian	489.0	10.224949	4.051223	2.0	7.0	10.0	13.0	25.0
southern_us	4320.0	9.634954	3.869404	1.0	7.0	9.0	12.0	40.0
spanish	989.0	10.423660	4.160919	1.0	7.0	10.0	13.0	35.0
thai	1539.0	12.545809	4.411794	1.0	9.0	12.0	15.0	40.0
vietnamese	825.0	12.675152	5.256173	1.0	9.0	12.0	16.0	31.0

**Algorithms and Techniques:**

Logistic Regression:

Logistic regression classifier is more like a linear classifier which uses the calculated logits (score) to predict the target class.

Definition from Wikipedia: "Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function"

The dependent variable is the target class variable we are going to predict. However, the independent variables are the features or attributes we are going to use to predict the target class.

Following picture is for a model which predicts how likely that penguin will be happy based on its daily activities.
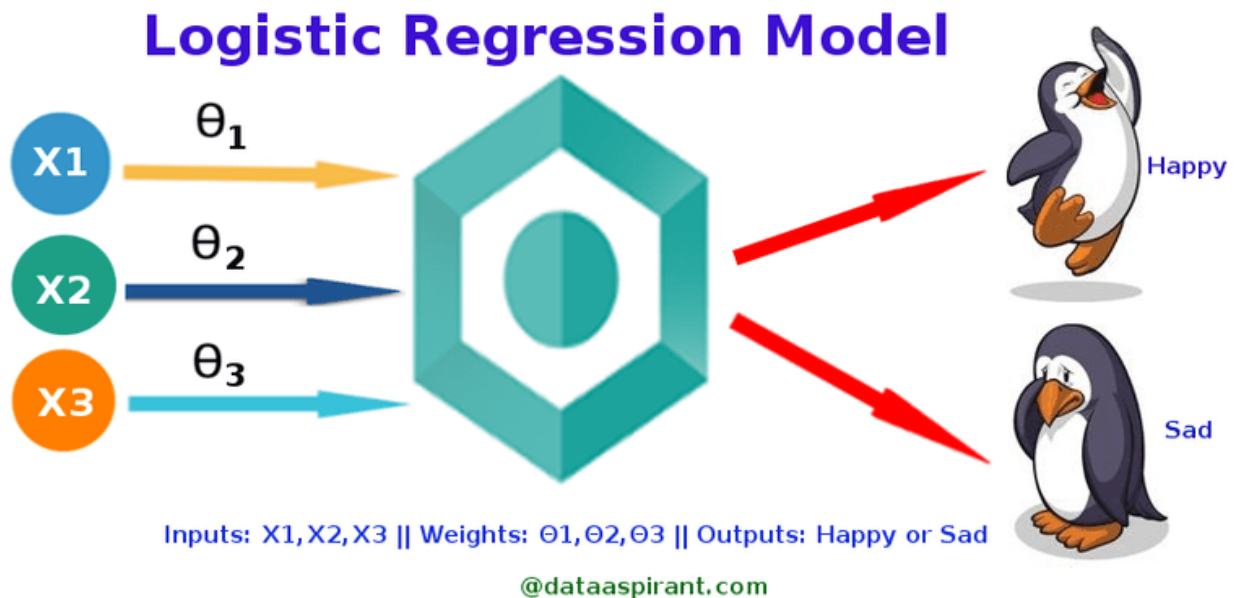
In our model we will import Logistic Regression from sklearn and we will tune hyperparameters for better accuracy. As our problem statement comes under multi class classification problem, I have chosen to use Logistic Regression for several reasons which will be discussed in later part of the report.

Benchmark:

As initial baseline model, I have started with Null model as the benchmark model; null model always predicts the most frequent class from the training data.

y_test.value_counts().head(1) / y_test.shape

```
italian    0.199216
```

So, our model accuracy should be way greater than the null model accuracy which is 0.199216.

After the review comments from my proposal submission I am going to use Naïve Bayes as my benchmark model. Naïve Bayes model has produced an accuracy score of 0.67115 on the test data set.

Methodology:

Data Preprocessing:

Preprocessing of data is carried out before the model is built and training process is executed:

Following are the steps carried out during preprocessing:

1. Dataset is divided into a training set and testing dataset.
2. Will apply feature engineering techniques to create new features like number of ingredients, mean ingredient length Ingredients are in the form of list of strings so I will convert them into a string
3. All the new features will be created for testing data set as well before we predict.

Implementation:

The first step in this process will be training a classifier with the following sub steps:

1. Reading the training data and look for null values, if we find minimal amount of null values then delete them from dataset.
2. Adding new features through Feature Engineering process
3. Adding new features to existing ones through FeatureUnion so that it will be compatible with sklearn terms once we vectorize.
4. Then use CountVectorizer to convert text into a matrix of token counts.
5. Use Pipeline for proper cross validation.
6. Using GrisSearchCV search(can be exhaustive search) and tune parameters
7. Final step would be predicting on test data and calculate accuracy score, if accuracy is not high then return to step 5 to tune model or for change of hyper parameters.

Refinement:

Initially I have started with Support Vector Machines and the accuracy achieved was not satisfactory (68%) so I have tried with Logistic Regression model which resulted in 70% accuracy. I have concentrated my energy more on Feature Engineering through which I came to know about Feature union Process. I have started with Random Forests with different hyper parameters which resulted in accuracy of 75% and then I went back to Logistic Regression which resulted in accuracy of 78.73%.

Results

Model Evaluation and Validation:

The final hyper parameters were chosen because they performed best among the tried combinations. During model creation and development, a validation set was used to evaluate the model.

Complete Description of Final Model:

1. After Reading the training data and deep diving into the ingredients feature provided, I have come to a conclusion to add more features.
2. Adding new features through Feature Engineering process
3. Now we have new features created through step2 and inorder to train a model on both(newly created and given), these features have to be combined to generate one feature matrix
4. Using FeatureUnion adding new features to the document term matrix was accomplished. Refer to the below picture on how it was achieved:

```
In [153]: from sklearn.preprocessing import FunctionTransformer

In [191]: new_features_ft = FunctionTransformer(new_features, validate=False)
```

5. Then use CountVectorizer to convert text into a matrix of token counts and then using Pipeline and grid search I have tuned and found best hyper parameters.

To verify the robustness of the model, I have used ingredients of food I prepare and few from internet and these are shown in free form visualization and the model performed pretty good.

Justification

Finally, the model is tried on the Kaggle's test data set which resulted in public score of 78.73% which is pretty higher than our null model and baseline model (Naïve Bayes). The public leaderboard score is 83.21% So our model has performed decently and this can be used for predictions.

Conclusion

Free-Form Visualization:

I have tried with some recipes' which I use to cook and then few from google to validate the model.

Prediction 1:

```
In [73]:   # test_data = pd.read_json('../data/test.json')
           test_data = number_of_ingredients(test_data)
           test_data = mean_ingredient_length(test_data)
           test_data = ingredients_as_string(test_data)

In [72]:   test_data = pd.DataFrame({'ingredients': ('balsamic vinegar, pasta, basil, mozzarella')}, index=[0])

In [74]:   test_data.head()
```

Out[74]:

	ingredients	num_of_ingredients	ingredient_length	ingredients_string
0	balsamic vinegar, pasta, basil, mozzarella	42	1.0	balsamic vinegar, pasta, basil, mozzarella

click to scroll output; double click to hide

```
In [32]:   grid.best_estimator_

Out[32]:   Pipeline(memory=None,
                steps=[('featureunion', FeatureUnion(n_jobs=1,
                    transformer_list=[('pipeline', Pipeline(memory=None,
                    steps=[('functiontransformer', FunctionTransformer(accept_sparse=False,
                        func=<function get_ingredients at 0x000000000C2DCDD8>,
                        inv_kw_args=None, inverse_func=None, kw_a...ty='l2', random_state=None, solver='liblinear', tol=0.0001,
                        verbose=0, warm_start=False))])

In [75]:   test_data_pred = grid.predict(test_data)
           test_data_pred

           <class 'pandas.core.series.Series'>
           <class 'pandas.core.frame.DataFrame'>

Out[75]:   array([u'italian'], dtype=object)
```

Prediction 2:

```python
In [69]:    # test_data = pd.read_json('../data/test.json')
            test_data = number_of_ingredients(test_data)
            test_data = mean_ingredient_length(test_data)
            test_data = ingredients_as_string(test_data)
```

```python
In [68]:    test_data = pd.DataFrame({'ingredients': ('basmati rice, chicken, garam masala, egg')}, index=[0])
```

```python
In [70]:    test_data.head()
```

Out[70]:

	ingredients	num_of_ingredients	ingredient_length	ingredients_string
0	basmati rice, chicken, garam masala, egg	40	1.0	basmati rice, chicken, garam masala, egg

```python
In [32]:    grid.best_estimator_
```

```
Out[32]:    Pipeline(memory=None,
                 steps=[('featureunion', FeatureUnion(n_jobs=1,
                   transformer_list=[('pipeline', Pipeline(memory=None,
                 steps=[('functiontransformer', FunctionTransformer(accept_sparse=False,
                     func=<function get_ingredients at 0x000000000C2DCDD8>,
                     inv_kw_args=None, inverse_func=None, kw_a...ty='l2', random_state=None, solver='liblinear', tol=0.0001,
                     verbose=0, warm_start=False))])
```

```python
In [71]:    test_data_pred = grid.predict(test_data)
            test_data_pred
```

```
            <class 'pandas.core.series.Series'>
            <class 'pandas.core.frame.DataFrame'>
```

```
Out[71]:    array([u'indian'], dtype=object)
```

Prediction 3:

```python
In [65]:    # test_data = pd.read_json('../data/test.json')
            test_data = number_of_ingredients(test_data)
            test_data = mean_ingredient_length(test_data)
            test_data = ingredients_as_string(test_data)
```

```python
In [64]:    test_data = pd.DataFrame({'ingredients': ('basil, tomatoes, oregano, garlic, parmesan cheese')}, index=[0])
```

```python
In [67]:    test_data.head()
```

Out[67]:

	ingredients	num_of_ingredients	ingredient_length	ingredients_string
0	basil, tomatoes, oregano, garlic, parmesan cheese	49	1.0	basil, tomatoes, oregano, garlic, parmesan cheese

```python
In [32]:    grid.best_estimator_
```

```
Out[32]:    Pipeline(memory=None,
                 steps=[('featureunion', FeatureUnion(n_jobs=1,
                   transformer_list=[('pipeline', Pipeline(memory=None,
                 steps=[('functiontransformer', FunctionTransformer(accept_sparse=False,
                     func=<function get_ingredients at 0x000000000C2DCDD8>,
                     inv_kw_args=None, inverse_func=None, kw_a...ty='l2', random_state=None, solver='liblinear', tol=0.0001,
                     verbose=0, warm_start=False))])
```

```python
In [66]:    test_data_pred = grid.predict(test_data)
            test_data_pred
```

```
            <class 'pandas.core.series.Series'>
            <class 'pandas.core.frame.DataFrame'>
```

```
Out[66]:    array([u'italian'], dtype=object)
```

Prediction 4:

```
In [77]:   # test_data = pd.read_json('../data/test.json')
           test_data = number_of_ingredients(test_data)
           test_data = mean_ingredient_length(test_data)
           test_data = ingredients_as_string(test_data)

In [76]:   test_data = pd.DataFrame({'ingredients': ('avocadoes, beans, chipotles, tomatos')}, index=[0])

In [78]:   test_data.head()

Out[78]:
              ingredients          num_of_ingredients  ingredient_length          ingredients_string

         0  avocadoes, beans, chipotles, tomatos          36                1.0  avocadoes, beans, chipotles, tomatos


In [32]:   grid.best_estimator_

Out[32]:   Pipeline(memory=None,
               steps=[('featureunion', FeatureUnion(n_jobs=1,
                   transformer_list=[('pipeline', Pipeline(memory=None,
                 steps=[('functiontransformer', FunctionTransformer(accept_sparse=False,
                     func=<function get_ingredients at 0x000000000C2DCDD8>,
                     inv_kw_args=None, inverse_func=None, kw_a...ty='l2', random_state=None, solver='liblinear', tol=0.0001,
                     verbose=0, warm_start=False))])

In [79]:   test_data_pred = grid.predict(test_data)
           test_data_pred

           <class 'pandas.core.series.Series'>
           <class 'pandas.core.frame.DataFrame'>

Out[79]:   array([u'filipino'], dtype=object)
```

Predictions were accurate for the first three ones but for the last one, model has predicted the cuisine as "Filipino" for ingredients which fall under "Mexican" cuisine.

Reflection:

The process used for this project can be summarized using the following steps:

1. An initial problem of my interest and relevant, public datasets were found in Kaggle.
2. The data was downloaded and preprocessed(creating new features, FeatureUnion)
3. Benchmark model was created (Naïve Bayes model)
4. The classifier was trained using the data multiple times by tuning the parameters until the good set of hyper parameters were found.
5. The classifier was tested using the test data set and compared with public leader board.

After creating new features, I have struggled to find feature importance of the features to be used. There were instances where basic model used to perform well than complicated models(Ensembling) with more features. Then with permutation importance I was able to get more insights into feature importance and then as initial step I have appended new features I have created to the ingredients string and then did vectorization but I have found that weight of new features has become low may be due to the reasons of adding to the rest of the strings. After some research I have found that sklearn FeatureUnion can be used to accomplish this.

This is the most interesting supervised learning project I have ever worked and the way FeatureUnion was very interesting (bit complex).

Improvement:

To achieve more accuracy I think we may need bigger dataset with more data regarding Russian, Brazilian, Jamaican cuisines and few more. Other area of improvement can be use of more robust models and may be ensembling two or three models. May be we can more cuisine types rather than restricting us to 20 types to serve better purpose.

References:

http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.FeatureUnion.html

http://michelleful.github.io/code-blog/2015/06/20/pipelines/

http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

https://codesachin.wordpress.com/2015/08/16/logistic-regression-for-dummies/

http://dataaspirant.com/2017/03/02/how-logistic-regression-model-works/

https://machinelearningmastery.com/logistic-regression-for-machine-learning/

https://en.wikipedia.org/wiki/Logistic_function