Machine Learning Engineer Nanodegree

Capstone Report : What's cooking

Mandeep Kaur
April 10, 2017

Definition

Project Overview

Cuisine is the style of cooking. Every culture in this world has its own cuisine. Every cuisine has its own unique taste. Everyone can agree that this uniqueness comes from the choice of ingredients. Nobody can deny that few flavours are strongly related to some certain parts of world. Many times people, including the food critics, try to define a cuisine entirely on the bases of its ingredients.

This project takes inspiration from a competition on Kaggle. I like trying new dishes, new flavours. There are many implementations and different ideas for this problem. For example 10 Most used ingredients [1] and data exploration using word2vec algorithm[2]

Problem Statement

- The goal is to create a model that can, given the ingredients of a recipe, find the cuisine of that recipe it belongs to. This is a supervised learning classification problem. Based on list of ingredients we are going to predict the cuisine of the recipe.
- Find diffusion of cuisines. How similar and different they are based on the ingredient used. This is an unsupervised learning problem. We are going to find relationships and groups between different cuisine using ingredients.

Evaluation Metrics

Logloss is the function that quantifies the accuracy of the classifier by penalizing false classifications. The classifier assigns the probability to each class rather than simply yielding the most likely class.

So we can simply define LogLoss in mathematical terms as:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

This classifier was used because this is a classification problem and the efficiency of the solution can be evaluated on the number of cuisines it can identify accurately. And because this is a small dataset using this metric will be more efficient.

Analysis

Data Exploration

For this project the dataset is publically available on Kaggle's competition "What's cooking?"[3] The dataset includes the recipe id, cuisine and list of ingredients.

An example is:


```
{
 "id": 24717,
 "cuisine": "indian",
 "ingredients": [
     "tumeric",
     "vegetable stock",
     "tomatoes",
     "garam masala",
     "naan",
     "red lentils",
     "red chili peppers",
     "onions",
     "spinach",
     "sweet potatoes" ]
 }
```


Total size of data is 1.76MB. This dataset contains a total of 6714 unique ingredients and 39774 sets of recipes' ingredient lists with their cuisines.

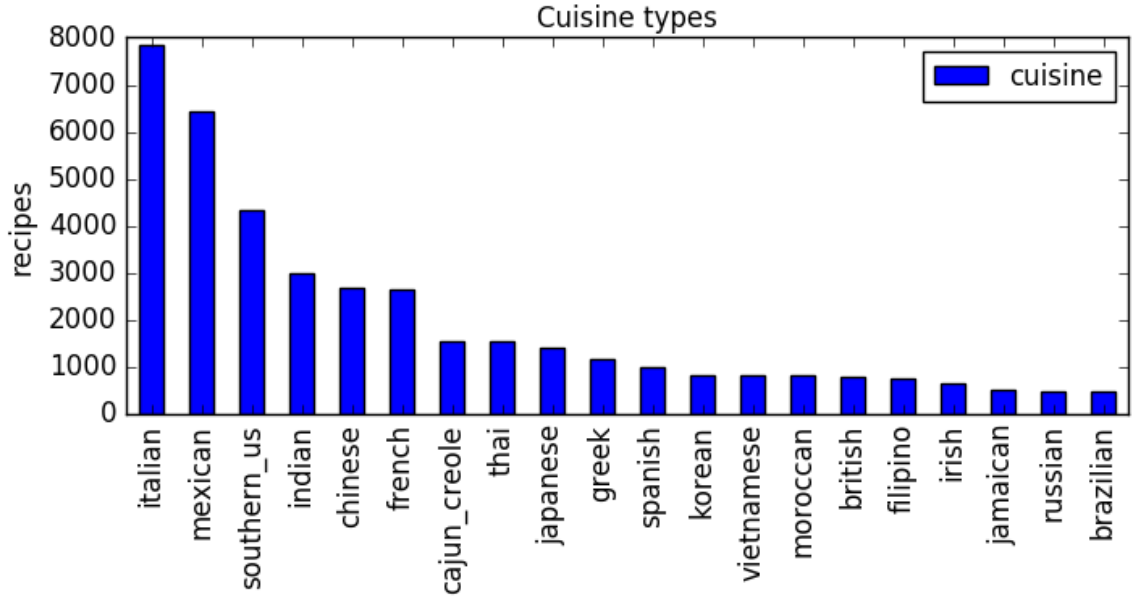
This dataset has 20 different cuisines.

```
['british',
 'french',
 'moroccan',
 'russian',
 'filipino',
 'japanese',
 'mexican',
 'brazilian',
 'indian',
 'chinese',
 'italian',
 'vietnamese',
 'spanish',
 'irish',
 'greek',
 'korean',
 'southern_us',
 'cajun_creole',
 'thai',
 'jamaican']
```

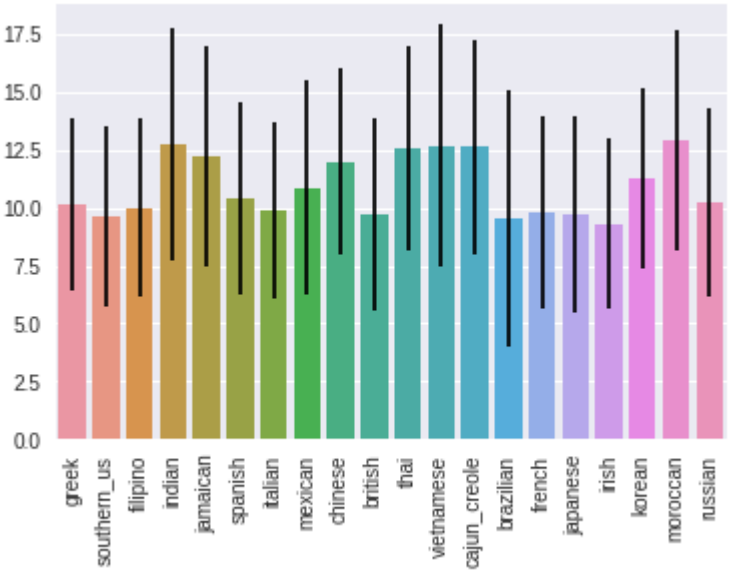
These cuisines include Asian, African, European and American.

Exploratory Visualization

The plot below shows the number of recipes present for each cuisine in dataset



As we can see Italian cuisine has maximum number of recipes and brazilian has least and the difference between max and min is very high. From cajun to brazilian the number of recipes are almost same.
The following chart shows the average size of list of ingredients in the recipe for each cuisine. As you can see there is not much change in average size of ingredient lists. The range lies between 9 to 13.



From the above two graphs it is clear that there is a bias toward a few cuisines in this database as they have more recipes than other, otherwise from the perspective of ingredients in each recipe data is not biased.

Algorithms and Techniques

For **data pre-processing** the main technique used is the TfIdf transformation:

TfIdf:
This is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It increases the proportionally to the number of times a word appears in the document, but is offset by the frequency of word in the corpus which helps to adjust the fact that some words appears more frequently in general.

The algorithms used for **Finding Cuisine** are :
Random Forest, Logistic Regression and Stochastic gradient descent

Logistic Regression:
Logistic Regression is a predictive analysis. It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. It is fast, give more importance to more relevant features. But, it requires the data points to be independent of one another, otherwise it will overweigh the significance of observations

Random Forest:
Random Forest is an ensemble learning method that operates by constructing a multitude od decision trees at training time and outputting a class that is mode of classes(for classification) of individual trees. This is one of most accurate learnings available and produces highly accurate classifier. It corrects the over fitting of decision trees to training set but has bias for attributes with more levels

Stochastic gradient descent:
This is also known as incremental gradient descent. This is the stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. It work well for the error manifolds that have lots of local maxima/minima. Setting the learning rate for this algorithm can be problematic in basic algorithm.

We trained our model on all the 3 algorithms and then compared based on speed and logloss score. After comparison we choose Logistic Regression as our final algorithm and find the best parameters for this algorithm.

For **Cuisine Diffusion** the algorithms and techniques used are:
PCA, K Means and jaccard similarity

PCA:
PCA is a method of dimensionality reduction. It uses single value decomposition to project it to lower dimensional space. It transforms the data to new coordinates such that the greatest varience by some projection of data comes to lie on the first coordinate and the second greatest on the second coordinate and so on. Dimensionality reduction loses information, in general PCA tends to minimize the information loss.

K Means:
This is one of the simplest unsupervised learning algorithms that solve clustering problems. This aims to partition n observations into k clusters in which each observation belongs to clusters with nearest mean

Jaccard similarity:
This compares the members for two sets to which members are shared and which are distinct. This is extremely senstive to small sample sizes and may give erroneous results.

Since our tfidf matrix has large number of features present we apply PCA to reduce the dimensionality. After that we create clusters and using jaccard similarity we define the diffusion of cuisines with one another.

Benchmark Model

For the benchmark model my choice is to use decision tree with the following parameters to find cuisine:
criterion : gini,
min_samples_split:2
min_samples_leaf=1
(This is the default configuration of the decision tree classifier in sklearn)

Decision tree classifier is a classic algorithm for this type of problem, so this is the best choice to create a benchmark model. The accuracy of predictions made on test set with this model is used as a result to compare with our model.

For cuisine diffusion, my benchmark model is predefine set of general rules, According to which all the cuisines will be clustered based on the region. The more geographically and continentally close the region the more close the clusters would be and vice versa. For example all the East Asian cuisines like Korean, Japanese will be closer to each other but far from Western ones like Mexicans or Spanish etc.

Analysis

Data pre-processing

Data pre-processing is done in file data_transformation.py for finding cuisine and in cuisine_diffusion.py for cuisine diffusion part:
For finding cuisine:
1. Load the data
2. Split the columns into two sets: X for ingredient list and Y for respective cuisines
3. Encode the ingredients list into tfidf matrix
For cuisine diffusion:
1. Load the data

2. Create a dictionary of cuisines with list of ingredients present in it
3. From this dictionary generate a count matrix with number of times each ingredient is present in the cuisine
4. Use this matrix to create the tfidf matrix

Above are the steps used to process the data to feed into the models. Both models ultimately use tfidf representation of data

Finding cuisine :

1. Divide the data into train and test.
2. Train the benchmark model and record the test results
3. Train this data on all the three algorithms.
4. Plot graphs for all algorithm performances.
5. Evaluate the best model.
6. Use this model to find the accuracy on test data.
7. Compare the results with benchmark model.

Cuisine diffusion:

8. Use the TfIdf matrix to find the count of ingredients for each cuisine.
9. Apply PCA
10. Compare the explained variance of variance dimensions
11. Choose the number of principal dimensions required to reduce dimensionality.
12. Generate reduced data using these dimensions
13. Apply k-means on this reduced data
14. Plot the resulting data model for visualization.
15. Check whether the result is in accordance with our predefined set of rules.

Implementation

Implementation of the project follows the below mentioned steps. The project has two parts so both parts has its own flow.

Finding cuisine :

1. Divide the data into train and test.
2. Train the benchmark model and record the test results
3. Train this data on all the three algorithms.
4. Plot graphs for all algorithm train time, predict time, train score and test score.
5. Evaluate the best model based on the graph.

Cuisine diffusion:

6. Generate TfIdf matrix from count matrix

7. Apply PCA
8. Compare the explained variance of variance dimensions
9. Choose the number of principal dimensions required to reduce dimensionality.
10. Generate reduced data using these dimensions
11. Apply k-means on this reduced data

The whole flow of implementation is successfully captured in ipynb with explaination.

Refinement

Finding cuisine:

1. Find the best parameters for the algorithm using Grid Search
2. Use this model to find the accuracy on test data.
3. Compare the results with benchmark model.

Now that we have selected a model from all the algorithm choices, we need to fine tune this by tweaking the parameters. Grid search is a good way to do this. I chose LogisicRegression for my model and then using Grid search Iterated over following parameters:

- C : This is the inverse of regularization strength.
Regularization is applying a penalty to increasing the magnitude of parameter values in order to reduce overfitting. The problem comes when you have a lot of parameters (a lot of independent variables) but not too much data. In this case, the model will often tailor the parameter values to idiosyncrasies in your data -- which means it fits your data almost perfectly. However because those idiosyncrasies don't appear in future data you see, your model predicts poorly.
To solve this, as well as minimizing the error, you add to what is minimized and also minimize a function that penalizes large values of the parameters. Most often the function is $\lambda\Sigma\theta j2$, which is some constant λ times the sum of the squared parameter values $\theta j2$. The larger λ is the less likely it is that the parameters will be increased in magnitude simply to adjust for small perturbations in the data, in our case $C=1/\lambda$.
I used C : [1.0, 10.0, 50.0, 100.0]

- **solver** :These are the algorithms used in optimization.
    - lbfgs : This is from the family of quasi-newton methods and it uses limited amount of memory. It only store a few vectors that represent the approximation implicitly. This is particularly well suited for optimization problems with a large number of parameters.
    -liblinear: When dataset is small this is a good choice of algorithm. It is a linear classifier. Main features include probability estimate and weights for unbalanced data.
    -newton-cg: It is useful in multiclass problem. It has multinomial loss and can handle L2 penalties.

So, when we iterate over the above mentioned parameters and Grid search gives us best algorithm fit of parameters. Due to some issues with processing I had to apply Grid search on each solver one at a time. So I divided my dataset into train and validation set. After the grid

search returned parameters for each solver on train data. I used logloss function on validation set and compared the results.

Cuisine Diffusion :

For this part I used PCA to reduce the dimensions of the data we have. After comparing the variance of components, only first two components were enough to represent the data and passed this reduced data to K means to form groups.

To perform the diffusion I used Jaccard similarity index, as discussed in algorithms and techniques section.

Results

Model evaluation and validation

when we iterate over the above mentioned parameters and Grid search gives us best algorithm fit of parameters. Due to some issues with processing I had to apply Grid search on each solver one at a time. So I divided my dataset into train and validation set. After the grid search returned parameters for each solver on train data. I used logloss function on validation set and compared the results.

Libliniear : 0.7423
lbfgs: 0.7637
newton-cg: 0.7656

Justification

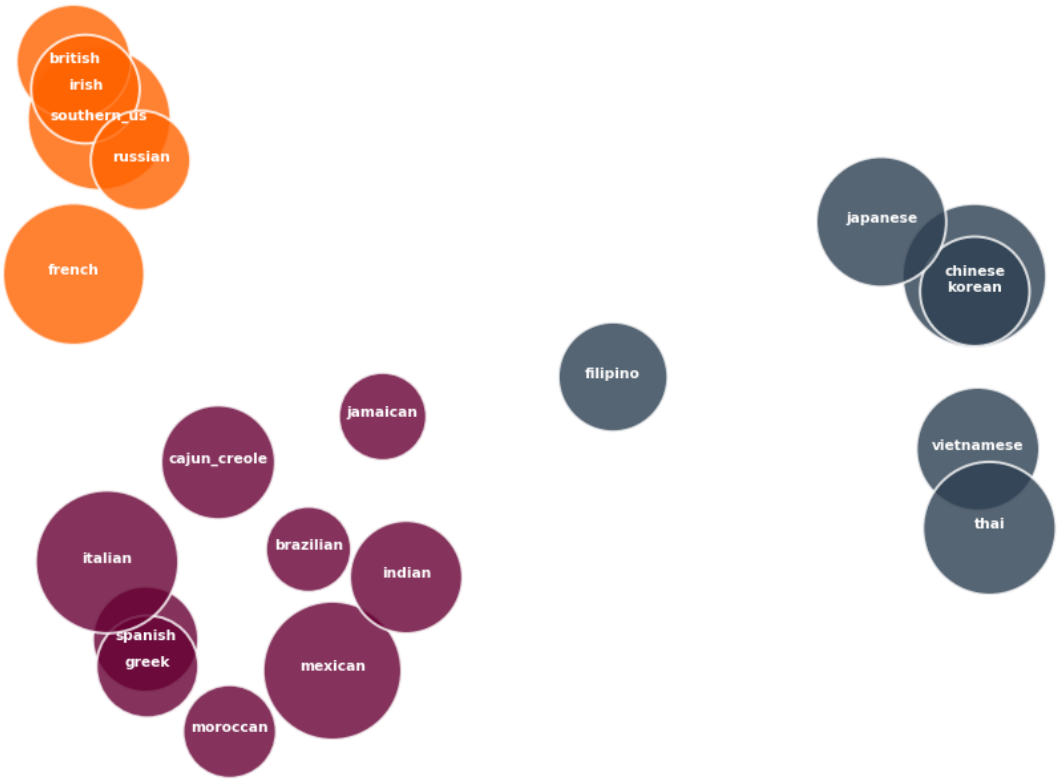
Finally the model's performance is checked on future data in our case test dataset that we kept in the beginning. On passing the test dataset through our model we get a score **of 0.7319**
The score of our benchmark is 13.06 which was too worse. Our model beats the benchmark model by a very large difference. With a score of 0.7319 I think we can say that we have a good model that can solve the classification of cuisine problem with very high efficiency.

For cuisine diffusion we stated earlier in our benchmark section that the cuisines that cuisines are diffused regionally. Our model clearly shows that relation. Asian cuisines like korean, chinese and japanies are closer while europian cuisines are closer to each other and far from Asian.

Conclusion

Free form visualization

For cuisine diffusion we get the following plot:



As you can see japanese, chinese and korean are close together, similarly thai and vietnamese. We know that even though filipino and indian are asian but they are different from east asia. Filipino is shown different from asian and Indian is with more similar cuisines.

Same applies to europian cuisines like Italian,sapnish and greek. Also british and irish are similar but different from continental.

Reflection

One of the biggest challenges in this project was tuning the model. Grid search was taking too much time. So I had to further break down the process but in the end I got an optimized model that beats the benchmark very easily.

Other problem was data representation. At first I tried using a binary vector for all the ingredients but processing that vector became very tough. Then I used TfIdf matrix and it worked, it took some time to find the proper representation of data.

Improvement

There is always a room for improvement. TfIdf is one type of representing the list of ingredients. There may be other representations that can improve the processing time of the algorithm and are more efficient to work with.

I used simple machine learning Logistic Regression technique to find cuisines. There are many other classification algorithms that can produce better results. Also, this problem can be solved using deep learning and that opens a whole new field to model that can be built to solve this problem.

Additionally for diffusion purpose, jaccard index is one of many techniques to find the similarities. If we change the number of PCA components we might find new relationships between cuisines.

---

citations:

[1] https://www.kaggle.com/manuelatadvice/whats-cooking/noname

[2] https://www.kaggle.com/ccorbi/whats-cooking/word2vec-with-ingredients

[3] https://www.kaggle.com/c/whats-cooking