

Beginner Competition for UESTC DM Lab 2023 Report

Yuchen Wu

June 23, 2023

Abstract

This article mainly documents the process of finishing this competition, including data visualization and analysis, sklearn algorithms and MLP construction by torch. For all submissions on [Kaggle](#), LogisticRegression finally gets the best Public score while SVM gets the second. I may choose bagging and LightGBM ensemble models for Private Scores. I've already upload my codes in [my repository](#).

1 Data visualization and analysis


In this section, I primarily introduces the steps that I did to visualize and analyze data. This part of code is [here](#).

1.1 Step 1: Read out Data

In Step 1, I mostly made use of the demo code for reading out data. In addition, I took analyzing unique data into consideration, where I got the below result showing that both training and testing data only contain values of 0 and 1. The function used is posted below.

```
def unique_value(df: pd.DataFrame) -> np.array:
    df_specific = df.drop(list(df.columns[:2]), axis=1, inplace=False)
    temp = df_specific.to_numpy()
    return np.unique(temp)
```

Then we can get the results in [Figure 1](#).



```
# 查看数据集内容unique情况（理论上test_df应该去除第一列即可）
unique_value(train_df), unique_value(test_df)
```

✓ 0.5s

```
(array([0, 1], dtype=int64), array([0, 1], dtype=int64))
```

Figure 1: Results from unique value function

1.2 Step 2: Ingredients Analysis

Because not all ingredients are contained in dishes, we have to figure out what kind of ingredients are contained in different types of cuisines. In an attempt, I built three dictionaries which called "food_index"(key: cuisine, value: ingredients(dict)), "food_count"(key: cuisine, value: occurrences), "loc_index"(key: cuisine, value: where this kind of cuisine located). The Parentheses after the names is the key and corresponding value of the dictionary. Then, we can have the overall results of different cuisines in [Figure 2](#).

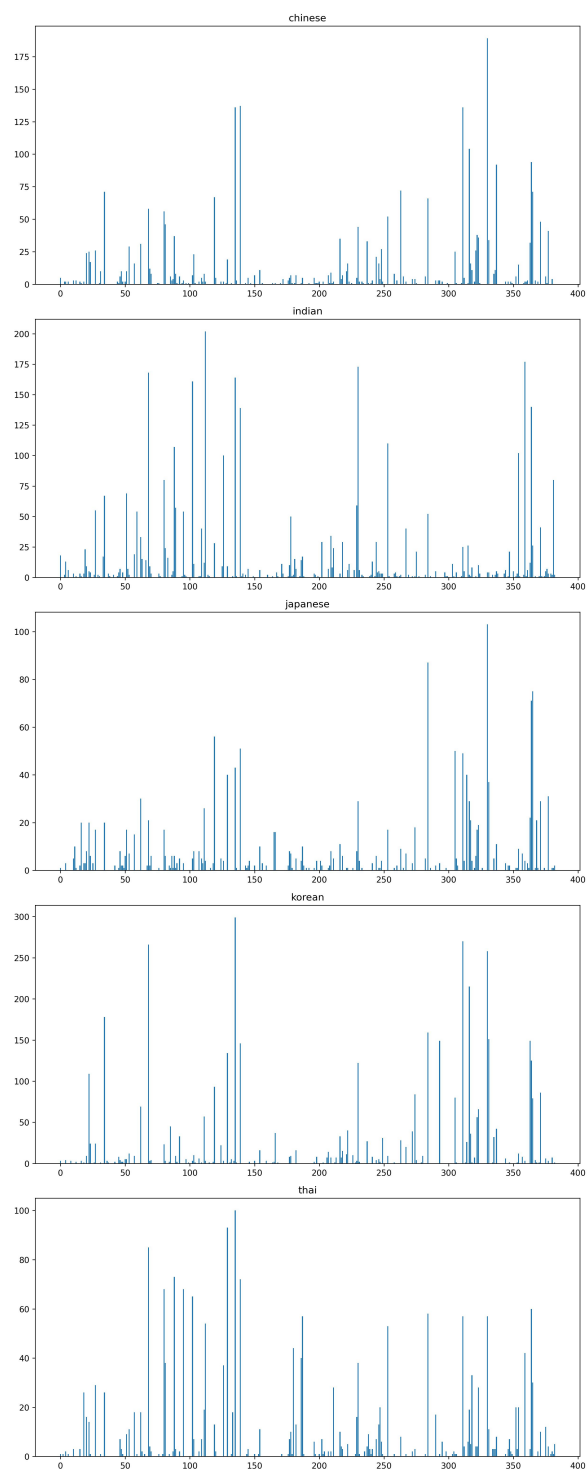


Figure 2: Ingredients of different cuisines

It's easy for us to find out that not all ingredients are included in different cuisines, which means that some ingredients are not used in some specific cuisines. This finding inspires me to do the following "choose feature" step.

After that, I specially picked out the top 3 most frequented cuisines("korean", "indian", "chinese") to gain a clearer insight of the ingredients' analysis which are showed in Figure 3.

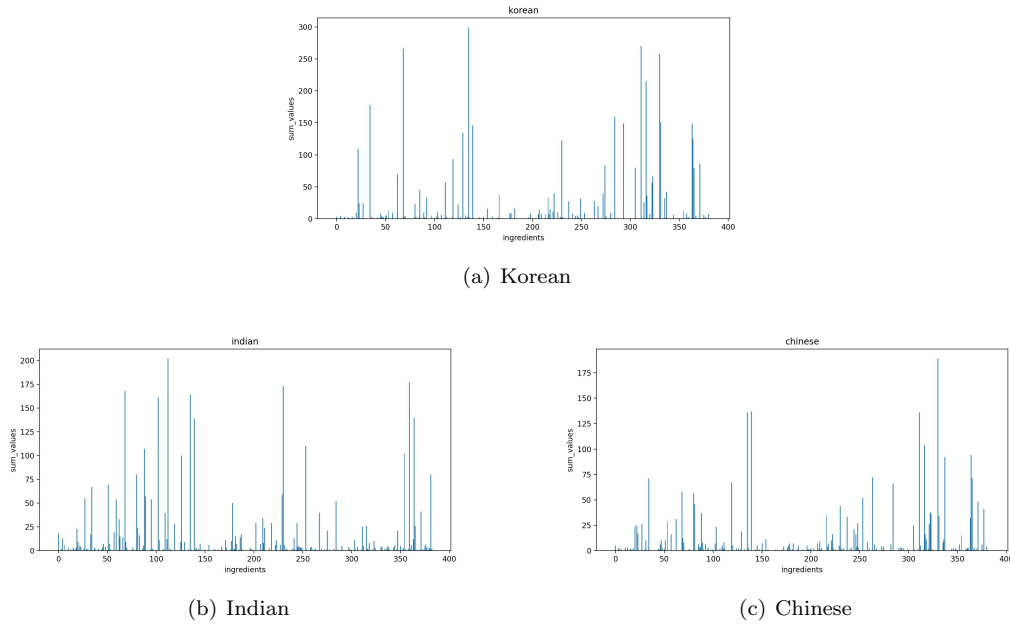


Figure 3: Top 3 frequented cuisines

1.3 Step 3: Feature Engineering

1.3.1 Choosing high correlation features

As showed above, the ingredients used vary, so I tried to pick out some important features to do the feature engineering. Specific steps are as below:

1. Using food_index dictionary to pick out the added ingredients of a kind of cuisine
2. Calculate the correlation matrix of the picked ingredients
3. Only choose the correlation that is over 0.8 features

The Feature Engineering code is as below:

```
cuisine_count = list(food_index.keys())
result_indices = np.array([])
for cuisine in cuisine_count:
    cur_features_names = [item[0] for item in food_index[cuisine].items() \
                          if item[1] != 0]
    cuisine_features = train_df.loc[loc_index[cuisine], cur_features_names]
    corr_matrix = cuisine_features.corr()
    # correlation > 0.8
    high_corr_pairs = np.where(np.abs(corr_matrix) > 0.8)
    selected_feature_indices = np.unique(np.concatenate(high_corr_pairs))
    # Combine different kinds of cuisine features that are picked out
    result_indices = np.union1d(result_indices, \
                                np.array(cur_features_names)[selected_feature_indices])
```

1.3.2 Draw Correlation Map

In this part, I drew different kinds of cuisines features(ingredients) that appear in corresponding cuisine in Figure 4.

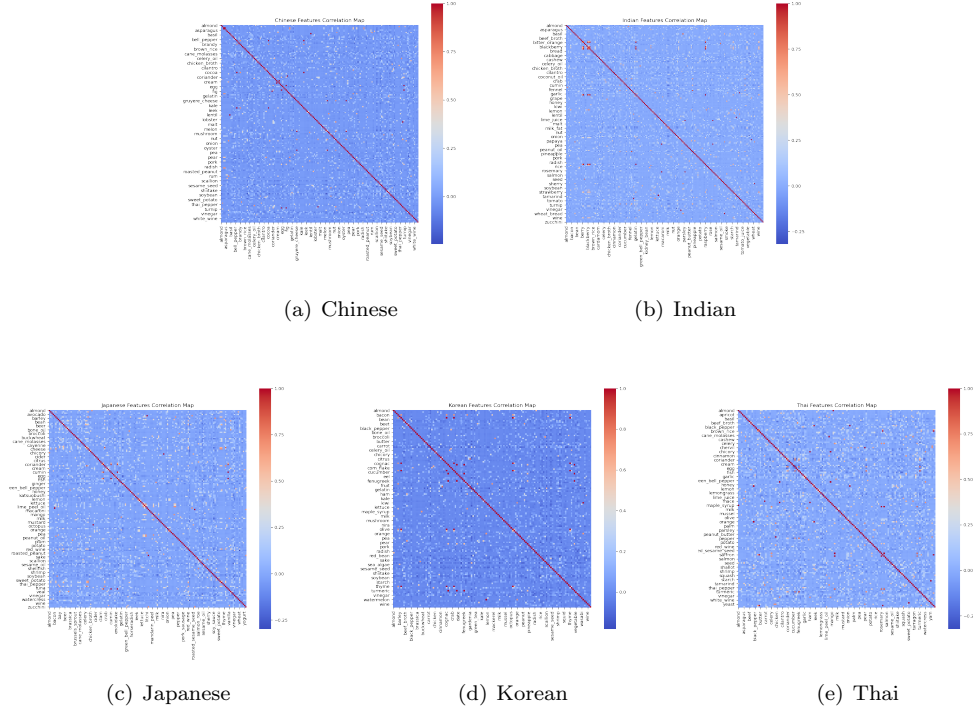


Figure 4: Correlation Maps of different kinds of cuisines

1.4 Step 4: Principal Components Analysis

Principal components analysis(PCA) is a multivariate statistical method to investigate the correlation between multiple variables, and it studies how to reveal the internal structure among multiple variables through a few principal components, that is, derive a few principal components from the original variables so that they can be as large as possible. Preserve the information of the original variables and are not correlated with each other. Usually, we can use PCA by `sklearn.decomposition`, which gets a parameter called `n_component`. Here, in order to visualize the PCA results, I set the parameter to 2, so we got Figure 5.

After that, we can also paint the correlation between the 2 principal components and the original features. Because the space limits, I only painted the correlation between the 2 principal components and the first 100 original features in Figure 6.

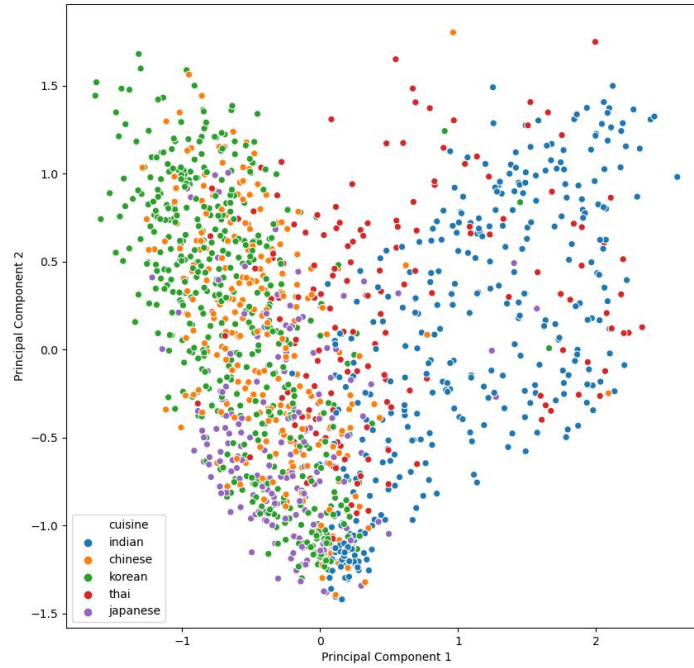


Figure 5: 2 Principal Components result Visualization

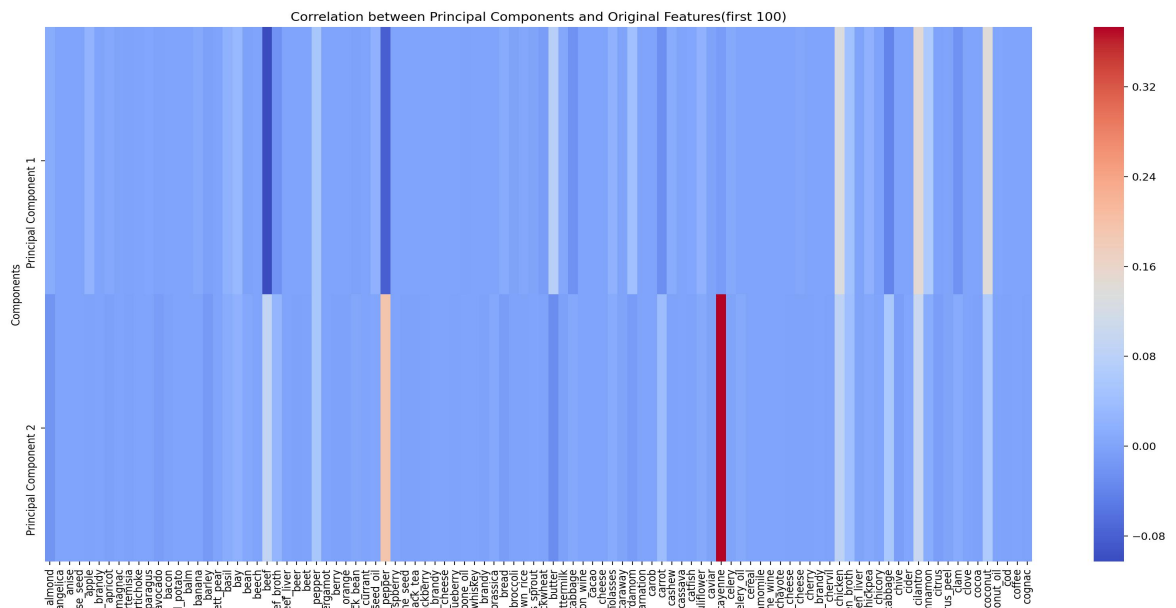


Figure 6: Correlation between 2 principal components and original features

2 Scikit-learn to build model

In this section, I simply show the models that I have tried to finish the task. Not only do I post the validation scores that I get, but also I will post the Public scores that I get in Kaggle. As Kaggle has the limit of submissions, so I will only post the best Public score for each kind of model.

The most important technique that I use is **GridSearch**, which can search the given parameter range through cross validation to get the most fitted parameter. The validation scores below are all from GridSearch.

As demo code uses Logistic Regression model to finish the task, I started with Logistic Regression model too. For high dimension features, it's usually linearly separable, which is good for Logistic Regression. Then, I tried some common classification models vary from **SVM**, **DecisionTree**, **BernoulliNB**. What's more, for ensemble models, I coded bagging classifier myself and tried LightGBM, Stacking in sklearn. The base model for these models are in the Parentheses. The results are shown in Table 1

Table 1: GridSearch Results among all models

Model Type	Model Name	Validation Score(Acc)	Public Score(Acc)
Single	Logistic Regression	0.8027	0.8328
Single	Support Vector Machine	0.8129	0.8123
Single	Decision Tree	0.7642	
Single	BernoulliNB	0.7993	0.8020
Ensemble	Bagging(SVM)	0.7823	0.7577
Ensemble	Bagging(Logistic Regression)	0.8061	0.7884
Ensemble	LightGBM	0.8095	0.8089
Ensemble	Stacking	0.5792	

As shown in Table 1, most single models are better than ensemble models in Public scores and Validation scores. The reason why this happened maybe the dataset isn't big enough (especially for Stacking). Also, bagging and LightGBM mainly focuses on robustness, so they can get good results and may have better Private Scores.

For single models, Logistic Regression gets the highest Public Score. The reason for this mainly lies on the high dimension features. However, SVM model gets the highest Validation Score, and the second Public Score, which shows that this kind of model using 'linear' kernel can study pretty good for high dimension tasks.

In fact, I've done some Dimension Reduction using both my Feature Engineering methods and PCA. Nonetheless, both methods were quite unsatisfactory, sometimes even got Scores less than 0.60. I've written these two methods in [my github repository](#).

3 Pytorch to build model

In this section, I only built Fully-connected network to finish the task. Here, I summarize the process of using Pytorch to finish a specific task. The codes for this section are [here](#).

1. Transform data to DataSet for building Dataloader

- **TensorDataSet**: only fits training process, needs to specify data and labels.
- **DataSet**: `__init__`, `__len__`, and `__getitem__` must be included, we can specify different return pattern to fit both training and predicting process. (the label must fit the loss function: cross-entropy needs "int64" label)

2. Define Loss, Batch_size, Num_epochs, Learning_rate, Optimizer and other hyper-parameters.

3. Build a Network

- Define `__init__` function, which receives required inputs and defines the layers used in *forward*.
- Define *forward* function, which is forward propagation process.

4. Training Function

- Using the parameters given to get corresponding **DataLoader** for training and validation.
- Using the parameters given to initialize.
- Epoch training and validation process
 - Each epoch starts with *net.train()*, which means turning net mode into training.
 - Get a batch data from DataLoader.
 - *optimizer.zero_grad()* to turn grad to zero!
 - Get the output from net.
 - Calculate the Loss, then *Loss.backward()*, *optimizer.step()*.
 - Calculate the metrics that you need.
 - Do the validation similar to steps before, make sure to turn net to *eval* mode and start with *torch.no_grad()*.

5. Predict Function: turn to *eval* mode first, then get the output without grad(e.g. *pred.detach().cpu()*).

I only try a small MLP with 3 hidden layers here. During the process of training, I discovered that the high dimension features that fits Logistic Regression well can't fit MLP model. The MLP model couldn't study the original features, the accuracy score only reached **0.3**. I'm wondering whether it's because there's too many 0 values in some features that can be deleted. According to my guess, I used my Feature Engineering method to reduce dimension. The results then turned normal. The final best Validation Score is **0.7737**, Puclic Score is **0.7270**. Obviously, they're the best result for the small model I tried, but not the best result for other MLP models. Later, I may try some other network structures to reach a higher score.

4 Conclusion

In the process of completing this competition, I became familiar with the visualization and analysis of data and the methods of using sklearn and torch to build models. Specially, I came to realize the importance of analysis of data, which can give rise to results sometime.

What's more, I tried to use git to control version and latex to finish this report, which is a great improvement for me. The git screenshots are in the Section 5 Appendix.

All in all, I learned a lot of content that is helpful for my future study, helping me to review and consolidate my understanding of machine learning and neural networks. Well worth the trip!

5 Appendix

5.1 Branch main

Due to space constraints, only snippets are truncated.

```
C-DM-Lab (main)
$ git log
commit 615c1237858248c40826b3c7163d8a6a3f30f69b (HEAD -> main, origin/main, origin/HEAD)
Author: yclggsddu <1694650401@qq.com>
Date:   Fri Jun 23 10:53:38 2023 +0800

    update Readme

commit 6c95cf3df4716007cddbdc721e412b1d5886dd761
Author: yclggsddu <1694650401@qq.com>
Date:   Wed Jun 21 17:31:31 2023 +0800

    Readme

commit d6e0c1bc6275dda5160caed7edd415b92797689d
Author: yclggsddu <1694650401@qq.com>
Date:   Mon Jun 19 22:13:34 2023 +0800

    Readme

commit b57ece116272024e754ea697819a5b5225037257
Author: yclggsddu <1694650401@qq.com>
Date:   Sun Jun 18 17:04:40 2023 +0800

...skipping...
commit 615c1237858248c40826b3c7163d8a6a3f30f69b (HEAD -> main, origin/main, origin/HEAD)
Author: yclggsddu <1694650401@qq.com>
Date:   Fri Jun 23 10:53:38 2023 +0800

    update Readme

commit 6c95cf3df4716007cddbdc721e412b1d5886dd761
Author: yclggsddu <1694650401@qq.com>
Date:   Wed Jun 21 17:31:31 2023 +0800

    Readme

commit d6e0c1bc6275dda5160caed7edd415b92797689d
Author: yclggsddu <1694650401@qq.com>
Date:   Mon Jun 19 22:13:34 2023 +0800

    Readme

commit b57ece116272024e754ea697819a5b5225037257
Author: yclggsddu <1694650401@qq.com>
Date:   Sun Jun 18 17:04:40 2023 +0800

    demo

commit d30e18dcb77652246a920c2f223a20110bb94644
Author: yclggsddu <1694650401@qq.com>
Date:   Sat Jun 17 12:32:54 2023 +0800
```

Figure 7: git logs of branch main

5.2 Branch visualize

```
C-DM-Lab/visualize (visualize)
$ git log
commit 1d56b130268f618d16bcaac905caf6f70199c2b5 (HEAD -> visualize)
Merge: 4927da8 ac87343
Author: yclggsddu <1694650401@qq.com>
Date: Fri Jun 23 11:01:33 2023 +0800

    Merge branch 'visualize' of https://github.com/yclggsddu/Beginner-Competition-for-UESTC-DM-Lab into visualize

commit 4927da8e9fd496721de4bb5d1feca174df6c9f10
Author: yclggsddu <1694650401@qq.com>
Date: Fri Jun 23 11:00:38 2023 +0800

    add pic used in report

commit ac87343fa6d7473209e0ef3f3741f17c433c65f0
Author: yclggsddu <98590971+yclggsddu@users.noreply.github.com>
Date: Fri Jun 23 10:56:01 2023 +0800

    Create README.md

commit c7ae8b9fb6e3ebd570440aee60c8371f8ecd3b0f
Author: yclggsddu <1694650401@qq.com>
Date: Mon Jun 19 22:07:28 2023 +0800

    ...skipping...
Merge: 4927da8 ac87343
Author: yclggsddu <1694650401@qq.com>
Date: Fri Jun 23 11:01:33 2023 +0800

    Merge branch 'visualize' of https://github.com/yclggsddu/Beginner-Competition-for-UESTC-DM-Lab into visualize

commit 4927da8e9fd496721de4bb5d1feca174df6c9f10
Author: yclggsddu <1694650401@qq.com>
Date: Fri Jun 23 11:00:38 2023 +0800

    add pic used in report

commit ac87343fa6d7473209e0ef3f3741f17c433c65f0
Author: yclggsddu <98590971+yclggsddu@users.noreply.github.com>
Date: Fri Jun 23 10:56:01 2023 +0800

    Create README.md

commit c7ae8b9fb6e3ebd570440aee60c8371f8ecd3b0f
Author: yclggsddu <1694650401@qq.com>
Date: Mon Jun 19 22:07:28 2023 +0800

visualize
```

Figure 8: git logs of branch visualize

5.3 Branch sklearn

```
C-DM-Lab/sklearn (sklearn)
$ git log
commit 53572b899a31e1540ba6185b045aa72819d6042b (HEAD -> sklearn)
Author: yclggsddu <1694650401@qq.com>
Date: Fri Jun 23 10:48:39 2023 +0800

    sklearn

commit e65bc2285a64c725413f8bc6e8a85c1900e01a1d
Author: yclggsddu <1694650401@qq.com>
Date: Wed Jun 21 17:50:27 2023 +0800

    sklearn

commit 0ca6aa15c43aa9e602662d3311f314e8ee1be74
Author: yclggsddu <1694650401@qq.com>
Date: Wed Jun 21 17:49:54 2023 +0800

    GridSearch
```

Figure 9: git logs of branch sklearn

5.4 Branch torch

```
C-DM-Lab/torch (torch)
$ git log
commit 49cd727133cc39bd01262678fa0bff9ffa84a04 (HEAD -> torch)
Author: yclggsddu <1694650401@qq.com>
Date:   Wed Jun 21 17:34:17 2023 +0800

    torch
```

Figure 10: git logs of branch torch