Report

Project 1: Classification

NYCU Spr2022: AI Capstone [DCP1271]

Author: 0816066 官澔恩

Contents

I. Datasets

Public Image Dataset

Source: Kaggle: Satellite Image Classification

This is a dataset from Kaggle. It contains 4 classes of satellite images, which are cloudy, desert, green, and water areas. Except there being only 1,131 images of desert area, there are 1,500 images of other classes.

Public Non-image Dataset

Source: Kaggle: Stellar Classification Dataset - SDSS17

This is a dataset from Kaggle. It contains 100,000 stellar data with 18 attributes, one of which is the class of an object. Besides, there are 3 classes, which are galaxy, quasar, and star. What's worth noticing is that the number of each class is unbalanced. There are 59,445 galaxies, 21,594 stars, and 18,961 quasars. Thus, data augmentation is required.

Self-made Dataset

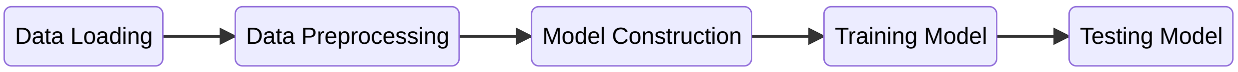
Source: BBC News

I made a web scraper with Selenium and BeautifulSoup that scrapes news articles from BBC News. I chose "business", "entertainment & arts", "science & environment", and "technology" as the categories to be classified. Each of them has 50 articles, that is, 200 articles in total.

First, I collected links to the page of each category. Then, I automated the following procedures with my program. For each page, it looks for and collects links that end with the same category name of current page, but if a link directs to a video news, it will be skipped. After that, it obtains a dictionary of lists containing links to news articles of each category. Subsequently, it creates directories for each category, and fetches each article and saves it under corresponding directory.

II. Procedures

Outline



Satellite Image Classification

1. Load images from each directory, and save the image data and its label (the directory's name) in two lists.
2. Scale each image into a 64×64 square.
3. Reshape each image data into a one-dimensional array.

4. Split the lists into a training set and a testing set.
5. Create models with different parameters and apply 5-fold cross validation with the training set.
6. Choose the best model and apply validation with the testing set.
7. Try different models and follow step 4 and 5.

Stellar Classification

1. Load the CSV file into a data frame of Pandas.
2. Label encode the target feature.
3. Find out the correlation coefficient between each feature with the target feature, and drop those with the absolute value of the coefficient no more than 0.02.
4. Split the data frame into a part without target feature (`data_X`) and the target feature (`data_y`).
5. One-hot encode all remaining categorical features in `data_X` .
6. Split the `data_X` and `data_y` into a training set and a testing set.
7. Randomly undersample the data of major classes in the training set to make the number of each class equal.
8. Apply PCA on the training set, and apply the transformation on both training set and testing set.
9. Scale the value of each feature in the training set into $[0, 1]$, and use the same criterion on the testing set.
10. Follow step 4 to 6 in Satellite Image Classification section.

BBC News Classification

1. Load articles from each directory, and save the texts and its label (the directory's name) in two lists.
2. Split the lists into a training set and a testing set.
3. Tokenize the article, remove stop words from it, and vectorize it with TF-IDF of each remaining word.
4. Follow step 4 to 6 in Satellite Image Classification section.

In all the train-test-spliting steps above, I tried 2 different testing set ratio, which are 0.2 and 0.3.

III. Algorithms

1. K-Nearest Neighbors

I tried KNN model with 3 different k, which is the number of neighbors to use.

- `n_neighbors` : $5, 10, 15$

2. Random Forest

I tried random forest model with 3 different minimum number of samples to be a leaf node.

- `min_samples_leaf` : $1, 5, 10$

The number of decision trees I used is 100, which is chosen by default.

3. Support Vector Machine

I tried SVM model with 3 different C, which is a regularization parameter. The strength of the regularization is inversely proportional to it.

- `C` : $1, 5, 10$

The kernel I used is radial basis function, which is chosen by default.

4. Multilayer Perceptron

I tried MLP model with 3 different sizes of the hidden layer.

- `hidden_layer_size` : $256, 512, 1024$

The model has only one hidden layer.

IV. Analysis

Note: “pre.”, “rec.”, and “acc.” are abbreviations of “precision”, “recall”, and “accuracy”, and “F1” means “F1-score”.

1. Satellite Image Classification

Label	0	1	2	3
Name	Cloudy Area	Desert Area	Green Area	Water Area

Testing Performance of KNN

With Test Size: 0.2

n_neighbors	Avg Pre.	Avg Rec.	Avg F1	Acc.
5	0.90	0.88	0.88	0.87
10	0.90	0.89	0.89	0.88
15	0.90	0.88	0.88	0.88

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.95	0.96	0.96
1	0.99	0.94	0.96
2	0.94	0.69	0.79
3	0.73	0.95	0.83

With Test Size: 0.3

n_neighbors	Avg Pre.	Avg Rec.	Avg F1	Acc.
5	0.90	0.87	0.87	0.87
10	0.90	0.88	0.89	0.88
15	0.90	0.88	0.88	0.88

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.94	0.96	0.95
1	0.99	0.93	0.96
2	0.93	0.68	0.78
3	0.74	0.94	0.83

Other Observation

It had poor recall ($\sim 70\% - 75\%$) on class 2 (green area), and poor precision ($\sim 65\% - 70\%$) on class 3 (water area).

Testing Performance of Random Forest

With Test Size: 0.2

min_samples_leaf	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.94	0.94	0.94	0.94
5	0.94	0.93	0.93	0.93
10	0.93	0.93	0.93	0.92

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.96	0.95	0.96
1	0.97	0.97	0.97
2	0.89	0.94	0.91
3	0.92	0.88	0.90

With Test Size: 0.3

min_samples_leaf	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.94	0.93	0.94	0.93
5	0.93	0.93	0.93	0.93
10	0.93	0.93	0.93	0.92

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.94	0.94	0.94
1	0.96	0.94	0.95
2	0.91	0.94	0.92
3	0.92	0.91	0.91

Other Observation

On each class it had high recall and precision, which ranged around 90% to 95%.

Testing Performance of SVM

With Test Size: 0.2

C	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.91	0.91	0.91	0.90
5	0.92	0.92	0.92	0.91
10	0.92	0.92	0.92	0.92

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.96	0.94	0.95
1	0.95	0.96	0.96
2	0.84	0.94	0.89
3	0.92	0.81	0.86

With Test Size: 0.3

C	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.91	0.90	0.90	0.90
5	0.92	0.92	0.92	0.92
10	0.92	0.92	0.92	0.91

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.95	0.93	0.94
1	0.94	0.96	0.95
2	0.84	0.95	0.89
3	0.93	0.82	0.87

Other Observation

It had poor precision on class 2 (green area), and poor recall on class 3 (water area). Both were ranged around 80% to 85%.

Testing Performance of MLP

With Test Size: 0.2

hidden_layer_size	Avg Pre.	Avg Rec.	Avg F1	Acc.
256	0.87	0.87	0.86	0.86
512	0.84	0.82	0.82	0.82
1024	0.86	0.86	0.86	0.86

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.92	0.87	0.89
1	0.88	0.90	0.89
2	0.77	0.95	0.85
3	0.87	0.69	0.77

With Test Size: 0.3

hidden_layer_size	Avg Pre.	Avg Rec.	Avg F1	Acc.
256	0.87	0.88	0.87	0.87
512	0.87	0.87	0.86	0.86
1024	0.86	0.86	0.86	0.85

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.92	0.89	0.90
1	0.88	0.93	0.91
2	0.80	0.88	0.84
3	0.86	0.76	0.80

Other Observation

It had varied precision ($\sim 75\% - 85\%$) on class 2 (green area), and vaired recall ($\sim 65\% - 85\%$) on class 3 (water area).

i. Test Size Comparison

There's no big difference between the performance of these 2 setups. From my perspective, it's because 0.2 is close to 0.3. I should have tried 2 setups with a larger gap.

ii. Classifier Comparison

According to the result, on average, RF > SVM > KNN > MLP. To my surprise, MLP performed the worst, even KNN performed a little better than it. I guessed it's due to the number of hidden layers. If it was increased, perhaps MLP would perform the best.

Although, in terms of accuracy, KNN performed better than MLP, it's the other way around, in terms of recall and precision on class 2 and 3.

iii. Hyper-parameter Comparison

For KNN, according to the accuracy and F1-score of it, it performed the best when $K = 10$. An optimal K could be found near it.

For random forest, as the minimum required number of data to form a leaf node increased, the accuracy of random forest decreased. Therefore, in this classification task, it should not be set too high.

For SVM, it's observed that the larger C was, the better performance SVM had. Thus, further experiments can be conducted to find the optimal C.

For MLP, when test size is 0.2, it had the worst performance when `hidden_layer_size` is 512. In contrast, when test size is 0.3, the accuracy decreased as the `hidden_layer_size` increased.

It's interesting that though the accuracy decreased as the `hidden_layer_size` increased, the precision on class 2 and recall on class 3 increased.

Thus, a high performance and stable MLP model can be constructed with `hidden_layer_size` at least 1024 and perhaps more than one hidden layer.

---

2. Stellar Classification

Label	0	1	2
Name	Galaxy	Star	Quasar

Testing Performance of KNN

With Test Size: 0.2

n_neighbors	Avg Pre.	Avg Rec.	Avg F1	Acc.
5	0.70	0.73	0.71	0.74
10	0.70	0.73	0.71	0.74
15	0.69	0.72	0.70	0.73

class	Avg Pre.	Avg Rec.	Avg F1
0	0.84	0.76	0.80
1	0.67	0.76	0.71
2	0.58	0.67	0.61

With Test Size: 0.3

n_neighbors	Avg Pre.	Avg Rec.	Avg F1	Acc.
5	0.69	0.72	0.70	0.73
10	0.68	0.72	0.70	0.73
15	0.68	0.72	0.70	0.72

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.84	0.74	0.78
1	0.66	0.75	0.70
2	0.56	0.67	0.61

Other Observation

It had high f1-score on class 0, then middle f1-score on class 1, and finally low f1-score on class 2. This is positively related to the original number of data in that class.

Testing Performance of Random Forest

With Test Size: 0.2

min_samples_leaf	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.87	0.90	0.88	0.89
5	0.87	0.89	0.88	0.88
10	0.86	0.88	0.87	0.88

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.94	0.87	0.90
1	0.91	0.91	0.91
2	0.75	0.90	0.82

With Test Size: 0.3

min_samples_leaf	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.86	0.89	0.87	0.88
5	0.86	0.88	0.87	0.87
10	0.85	0.88	0.86	0.87

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.93	0.86	0.89
1	0.91	0.89	0.90
2	0.73	0.90	0.81

Other Observation

It had poor precision ($\sim 72\% - 76\%$) on class 2 (quasar), no matter the value of `min_samples_leaf` was.

Testing Performance of SVM

With Test Size: 0.2

C	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.85	0.88	0.86	0.86
5	0.88	0.91	0.89	0.90
10	0.89	0.92	0.90	0.91

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.95	0.86	0.90
1	0.91	0.88	0.89
2	0.75	0.98	0.85

With Test Size: 0.3

C	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.83	0.87	0.84	0.85
5	0.86	0.90	0.88	0.88
10	0.88	0.91	0.89	0.90

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.95	0.84	0.89
1	0.90	0.87	0.88
2	0.73	0.98	0.83

Other Observation

It had poor precision on class 2 (quasar), and it was related to the value of C. As it got larger, the precision got larger, too. It ranged from 67% to 80%.

Testing Performance of MLP

With Test Size: 0.2

hidden_layer_size	Avg Pre.	Avg Rec.	Avg F1	Acc.
256	0.93	0.93	0.93	0.94
512	0.87	0.92	0.89	0.89
1024	0.92	0.94	0.93	0.93

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.96	0.91	0.93
1	0.88	0.89	0.89
2	0.88	0.98	0.93

With Test Size: 0.3

hidden_layer_size	Avg Pre.	Avg Rec.	Avg F1	Acc.
256	0.89	0.93	0.91	0.92
512	0.90	0.93	0.91	0.92
1024	0.92	0.93	0.92	0.93

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.96	0.91	0.93
1	0.88	0.89	0.89
2	0.87	0.98	0.92

Other Observation

It had high recall and precision on each class compared with other models, each was about 85% to 95%.

i. Test Size Comparison

On this dataset, it's obvious that each model had worse performance on testing set with test size equal to 0.3 than the one with test size equal to 0.2. It's because a larger testing set means a smaller training set, and the models were not trained with enough information.

ii. Classifier Comparison

On this dataset, comparing the best model of each type, MLP > SVM > RF > KNN.

Although I have done undersampling on the major classes to make each class has equal size, the precision and recall on each class is still related to the original class number, and this phenomenon can be seen on each model. Thus, this is probably related to the sampled data instead of the model selection.

Take a look at MLP, SVM, and RF. They all have high precision and recall except on class 3, so I think they can be tuned further or trained with more data with class 3 to become more robust.

iii. Hyper-parameter Comparison

For KNN, as the K increased, the performance decreased. However, the effect was not obvious.

For random forest, similar to KNN, as `min_samples_leaf` increased, the performance slightly decreased.

For SVM, as the C increased, the performance increased. Therefore, we can know that the gap between each class in testing data is also small, and other unseen data in the training process also lay at the same side as the outlier.

For MLP, it's hard to tell the relationship between `hidden_layer_size` and the performance, since when the test size was 0.2, it dropped and bounced back, but when the test size was 0.3, it increased with the `hidden_layer_size` .

iv. PCA Comparison

After categorical features were encoded with one-hot encoding, there became a large number of features. Before applying PCA transformation, it took too long to train and eventually run out of memory. Therefore, it's necessary to apply PCA transformation.

3. BBC News Classification

Label	0	1	2	3
Name	Business	Entertainment & Art	Science & Environment	Technology

Testing Performance of KNN

With Test Size: 0.2

n_neighbors	Avg Pre.	Avg Rec.	Avg F1	Acc.
5	0.82	0.83	0.83	0.82
10	0.77	0.76	0.76	0.75
15	0.85	0.83	0.83	0.82

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.64	0.77	0.70
1	0.89	0.76	0.82
2	0.87	1.00	0.93
3	0.85	0.70	0.76

With Test Size: 0.3

n_neighbors	Avg Pre.	Avg Rec.	Avg F1	Acc.
5	0.85	0.85	0.85	0.85
10	0.88	0.89	0.88	0.88
15	0.86	0.87	0.86	0.87

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.84	0.86	0.85
1	0.98	1.00	0.99
2	0.87	1.00	0.93
3	0.75	0.62	0.68

Other Observation

With test size being 0.2, on average, it had low precsion on class 0, and high recall on class 2, while with test size being 0.3, on average, it had both low precsion and recall on class 3.

Testing Performance of Random Forest

With Test Size: 0.2

min_samples_leaf	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.88	0.88	0.87	0.88
5	0.78	0.78	0.76	0.78
10	0.82	0.75	0.74	0.75

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.90	0.50	0.63
1	0.75	0.94	0.83
2	0.97	1.00	0.98
3	0.69	0.77	0.72

With Test Size: 0.3

min_samples_leaf	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.82	0.82	0.78	0.77
5	0.75	0.78	0.73	0.73
10	0.36	0.59	0.44	0.48

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.67	0.36	0.47
1	0.82	0.81	0.80
2	0.62	1.00	0.76
3	0.48	0.74	0.58

Other Observation

With test size being 0.2, on average, it had low precsion on class 1 and class 3, and low recall on class 0 and class 3, while with test size being 0.3, on average, it only had high precsion on class 1 and high recall on class 1 and class 2.

Testing Performance of SVM

With Test Size: 0.2

C	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.81	0.77	0.75	0.78
5	0.91	0.90	0.90	0.90
10	0.91	0.90	0.90	0.90

Class	Avg Pre.	Avg Rec.	Avg F1
0	1.00	0.60	0.74
1	0.88	1.00	0.93
2	0.87	1.00	0.93
3	0.76	0.83	0.80

With Test Size: 0.3

C	Avg Pre.	Avg Rec.	Avg F1	Acc.
1	0.73	0.75	0.68	0.68
5	0.82	0.85	0.82	0.82
10	0.82	0.85	0.82	0.82

Class	Avg Pre.	Avg Rec.	Avg F1
0	1.00	0.55	0.70
1	0.89	0.98	0.93
2	0.68	1.00	0.80
3	0.58	0.74	0.65

Other Observation

With test size being 0.2, on average, it had low precision on class 3 and low recall on class 0, while with test size being 0.3, on average, it had low precision on class 2 and class 3 and low recall on class 0 and class 3.

Testing Performance of MLP

With Test Size: 0.2

hidden_layer_size	Avg Pre.	Avg Rec.	Avg F1	Acc.
256	0.90	0.90	0.90	0.90
512	0.83	0.83	0.82	0.82
1024	0.83	0.83	0.82	0.82

Class	Avg Pre.	Avg Rec.	Avg F1
0	0.88	0.70	0.78
1	0.83	1.00	0.91
2	0.93	1.00	0.97
3	0.77	0.70	0.73

With Test Size: 0.3

hidden_layer_size	Avg Pre.	Avg Rec.	Avg F1	Acc.
256	0.84	0.87	0.85	0.85
512	0.86	0.89	0.87	0.87
1024	0.86	0.89	0.87	0.87

Class	Avg Pre.	Avg Rec.	Avg F1
0	1.00	0.73	0.84
1	0.89	1.00	0.94
2	0.80	1.00	0.89
3	0.72	0.82	0.77

Other Observation

With test size being 0.2, on average, it had low precision on class 3 and low recall on class 0 and class 3, while with test size being 0.3, on average, it had low precision on class 3 and low recall on class 0.

i. Test Size Comparison

KNN and MLP have higher accuracy on larger testing set, while random forest and SVM have lower accuracy on larger testing set. This means the former models need less information when training, whereas the latter models need more.

ii. Classifier Comparison

The performance of each model: MLP ~ SVM > KNN ~ RF. All of them were good at classifying class 1 (entertainment & arts) and class 2 (science & environment), since they all had relative high f1-score on these classes.

Take a look at MLP and SVM, they both proned to predict news of other categories as technology news, and predict business news as news of other categories. Perhaps it's because those news contained technological terms and business news often related to other topics.

iii. Hyper-parameter Comparison

For KNN, the relation between K and performance was not consistent with different test size. In my opinion, it's due to the number of the whole dataset is small. Therefore, these 2 datasets have different groups of data when randomly sampled. Consequently, the optimal K for good performance was different.

For random forest, in this case, it's obvious that the larger `min_samples_leaf` was, the worse the performance was. This is also due to small dataset. Thus, if the minimum required number to be a leaf node is large, each decision tree was not fully expanded, which caused low performance.

For SVM, on this dataset, its performance was also positively related to the value of C.

For MLP, with a larger training set (smaller testing set), its accuracy dropped when `hidden_layer_size` grew, while with a smaller training set (larger testing set), it got higher when the value grew.

V. Discussion

Are the Results What I Expected?

- I guessed the setup with the larger testing set would perform worse; however, it depended.
- Initially, I guessed MLP would perform better on image data; however, in my experiments, it did better job on other type of data.
- I didn't expect KNN to have high accuracy, but the outcome showed that It could have accuracy at least 70% and even higher.

Factors Affecting the Performance

- How well the data are processed
- Training set vs. testing set ratio
- Model selection
- Hyper-parameter of each model

Further Experiments if Time Available

- Try larger testing sets.
- Try more parameter combinations.
- Try other methods to balance data.

What I have learned from the Project

- Skills of dynamic web scraping
- Preprocessing of image and text data
- Undersampling for imbalanced data
- Data analysis with the training / testing outcome