Stellar Classification Based on Supervised Learning

ECS784P

Abstract—Classification between galaxies, quasars, and stars is a fundamental aspect in astronomy. In this project, we performed two supervised learning algorithms on a dataset of 100,000 observation data labeled by either galaxies, quasars, or stars. We discussed the difference between the algorithms, possible optimizations and extensions.

Keywords—Stellar Classification, Supervised Learning, K-Nearest Neighbours, Support Vector Classification

Introduction

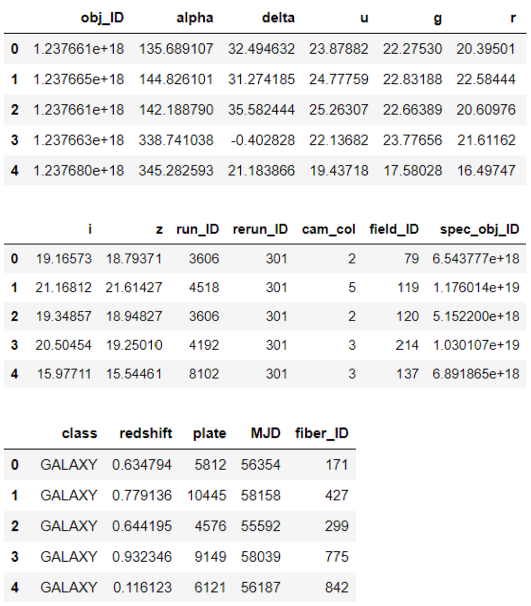
In astronomy, stellar classification is the classification of stars based on their spectral characteristics. The classification scheme of galaxies, quasars, and stars is one of the most fundamental in astronomy. The early cataloguing of stars and their distribution in the sky has led to the understanding that they make up our own galaxy and, following the distinction that Andromeda was a separate galaxy to our own, numerous galaxies began to be surveyed as more powerful telescopes were built.

The major work after obtaining a series of observation data is image processing and classification. Since our dataset is preprocessed from the images, we can skip the first step. In our project, we only discuss the rough classification among galaxies, quasars, and stars, the main difference between which is the luminosity and shape.

Data Description

This dataset is from Kaggle, and the link is provided in references.

The dataset aims to classification stars, galaxies, and quasars based on their spectral characteristics. It consists of 100,000 observations of space taken by the SDSS (Sloan Digital Sky Survey). Every observation is described by 17 feature columns and 1 class column which identifies it to be either a star, galaxy or quasar. Hence the shape is 100,000*18.



	obj_ID	alpha	delta	u	g	r
0	1.237661e+18	135.689107	32.494632	23.87882	22.27530	20.39501
1	1.237665e+18	144.826101	31.274185	24.77759	22.83188	22.58444
2	1.237661e+18	142.188790	35.582444	25.26307	22.66389	20.60976
3	1.237663e+18	338.741038	-0.402828	22.13682	23.77656	21.61162
4	1.237680e+18	345.282593	21.183866	19.43718	17.58028	16.49747

	i	z	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID
0	19.16573	18.79371	3606	301	2	79	6.543777e+18
1	21.16812	21.61427	4518	301	5	119	1.176014e+19
2	19.34857	18.94827	3606	301	2	120	5.152200e+18
3	20.50454	19.25010	4192	301	3	214	1.030107e+19
4	15.97711	15.54461	8102	301	3	137	6.891865e+18

	class	redshift	plate	MJD	fiber_ID
0	GALAXY	0.634794	5812	56354	171
1	GALAXY	0.779136	10445	58158	427
2	GALAXY	0.644195	4576	55592	299
3	GALAXY	0.932346	9149	58039	775
4	GALAXY	0.116123	6121	56187	842

Table 1 Data Sample

The meanings of the columns are:

1. obj_ID = Object Identifier, the unique value that identifies the object in the image catalog used by the CAS
2. alpha = Right Ascension angle (at J2000 epoch)
3. delta = Declination angle (at J2000 epoch)

4. u = Ultraviolet filter in the photometric system
5. g = Green filter in the photometric system
6. r = Red filter in the photometric system
7. i = Near Infrared filter in the photometric system
8. z = Infrared filter in the photometric system
9. run_ID = Run Number used to identify the specific scan
10. rerun_ID = Rerun Number to specify how the image was processed
11. cam_col = Camera column to identify the scanline within the run
12. field_ID = Field number to identify each field
13. spec_obj_ID = Unique ID used for optical spectroscopic objects (this means that 2 different observations with the same spec_obj_ID must share the output class)
14. class = object class (galaxy, star or quasar object)
15. redshift = redshift value based on the increase in wavelength
16. plate = plate ID, identifies each plate in SDSS
17. MJD = Modified Julian Date, used to indicate when a given piece of SDSS data was taken
18. fiber_ID = fiber ID that identifies the fiber that pointed the light at the focal plane in each observation

By running value_counts(), we learn that the dataset contains 59,445 items that is classified as galaxy, 21,594 items as star and 18,961 items as quasars. A visualized distribution is Fig.1.

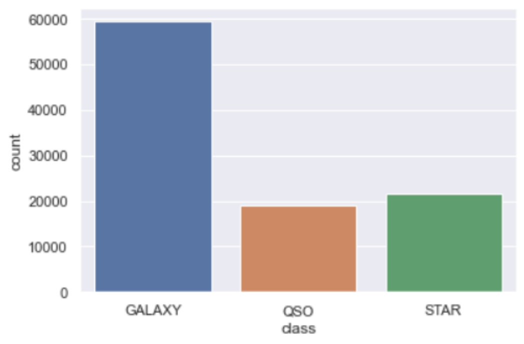


Fig 1 Data Distribution

Data Preprocessing

The dataset is not perfectly appropriate for modelling initially, hence we need some preprocessing.

First, based on the dataset description and the data sample, we can easily find that rerun_ID only shows how the original image was processed and provides no useful information in our modelling, so we choose to discard this feature.

Then, since we would choose the last 1/10 of the data as the test set and the rest as the training set, we should make sure that the data of the 3 labels are uniformly distributed. So we shuffled the data before distributing it to the two sets.

Some of the features contain big values and their differences provide difficulty in our modelling, hence we make the data points closer by data scaling.

The dataset does not contain null features, so we assume that no more preprocessing is needed.

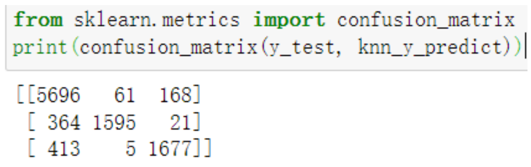
Modelling

In this dataset, the inputs together with expected labels are given, hence we choose to perform supervised learning on it. The training models we choose are the k-nearest neighbours algorithm and the support-vector clustering algorithm. These two methods are performed respectively to the data set and thus the results can be compared and evaluated.

K Nearest Neighbours

The k-nearest neighbours algorithm (KNN) is a supervised learning method that can be used in both classification and regression. It focus on an object's k closest neighbours to determine its classification result.

We apply KNN from scikit-learn on our training set and then use the resulted classifier to label the test set. The outcome is as follows:



```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, knn_y_predict))
```

```
[[5696   61  168]
 [ 364 1595   21]
 [ 413    5 1677]]
```

Fig 2 Confusion of KNN

This confusion matrix shows the number of each label given to data with expected labels, which is visualized in Fig 3:

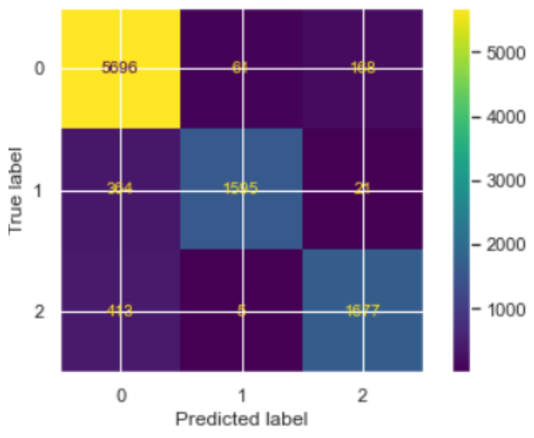
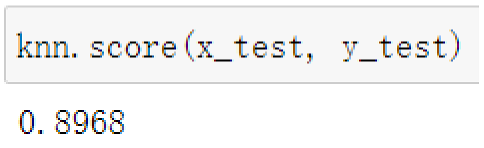


Fig 3 Visualized Confusion Matrix of KNN

And we can count the accuracy of KNN:



```
knn.score(x_test, y_test)
```

```
0.8968
```

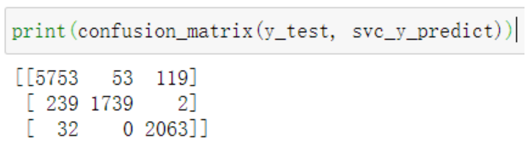
Fig 4 KNN Accuracy

The results show that our model make about 9 true classifications in every 10, which is not so high; but KNN is rapid enough in computing time.

Support Vector Classification

The support vector machine(SVM) is another kind of supervised learning method for both classification and regression with great robustness. The support-vector clustering algorithm(SVC), applies the statistics of support vectors in categorizing data. Unlike the original algorithm, SVC based on multiclass SVM can also be used in classification problems of more than 2 classes.

Similarly, we apply SVC from scikit-learn and get a slightly different result. In this process, the difference in computing time between the two algorithms is obvious. The outcome is as follows:



```
print(confusion_matrix(y_test, svc_y_predict))
```

```
[[5753   53  119]
 [ 239 1739    2]
 [  32    0 2063]]
```

Fig 5 Confusion of SVC

Then we also visualize the results. We can see clearly more data are given the true labels.

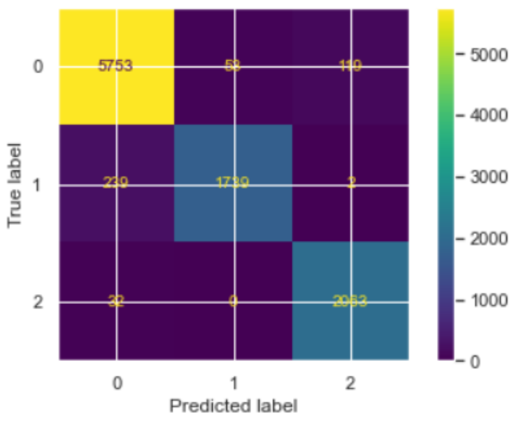


Fig 6 Visualized Confusion Matrix of SVC

Thus the accuracy of SVC is better than KNN.

```
print(np.mean(score))
```

```
0.9555
```

Fig 7 SVC Accuracy

Conclusion

Use the classification report, we can easily compare the outcomes of the two models in Table 2 and 3.

```
              precision    recall  f1-score   support

      GALAXY       0.88      0.96      0.92      5925
         QSO       0.96      0.81      0.88      1980
        STAR       0.90      0.80      0.85      2095

    accuracy                           0.90     10000
   macro avg       0.91      0.86      0.88     10000
weighted avg       0.90      0.90      0.90     10000
```

Table 2 Outcomes of KNN

```
              precision    recall  f1-score   support

      GALAXY       0.96      0.97      0.96      5925
         QSO       0.97      0.88      0.92      1980
        STAR       0.94      0.98      0.96      2095

    accuracy                           0.96     10000
   macro avg       0.96      0.94      0.95     10000
weighted avg       0.96      0.96      0.96     10000
```

Table 3 Outcomes of SVC

The outcomes show that in our training, the SVC algorithm shows better performance compared to the KNN algorithm in accuracy, but SVC shows a disadvantage in training time, because of its time complexity. We can then get the conclusion that SVC works better for small datasets but is harder to cope with a larger data scale.

However, both models do not provide a very satisfying accuracy, possible reasons are that we have not applied Principal Component Analysis to reduce the dimensions and we have not dealt with the imbalanced dataset.

The biggest trouble with the processing is the number of labels. The data should be classified to 3 classes, thus it is not a binary classification problem and we need to choose the proper classification algorithms. Since KNN and SVC can be used while logistic regression is hard to apply.

It is obvious that astronomical objects can be further subdivided into more precise classifications, for example, the Harvard system use letters O, B, A, F, G, K, M to classify stars with different colours (or surface temperature). So one of the extensions is applying the similar training and testing methods to a dataset with more specific classification labels, then carry out a model that can give better classification to those astronomical objects.

References

[1] Python Documents URL: https://www.python.org/doc/
[2] Python Data Analysis Library URL: https://pandas.pydata.org/
[3] API Reference -- scikit-learn documentation URL: https://scikit-learn.org/stable/modules/classes.html#
[4] Stellar Classification Dataset - SDSS17 | Kaggle URL: https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17

[5] Nearest Neighbors -- scikit-learn documentation URL:
https://scikit-learn.org/stable/modules/neighbors.html#neighbors
[6] Support Vector Machines -- scikit-learn documentation URL:
https://scikit-learn.org/stable/modules/neighbors.html#neighbors
[7] Stellar Classification | Kaggle URL:
https://www.kaggle.com/fahmisajid/stellar-classification#Model-Building
[8] Stellar Classification - 98.4% Acc 100% AUC | Kaggle URL:
https://www.kaggle.com/beyzanks/stellar-classification-98-4-acc-100-auc
[9] How to use Data Scaling Improve Deep Learning Model Stability and Performance URL:
https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/
[10] Stellar classification – Wikipedia URL:
https://en.wikipedia.org/wiki/Stellar_classification