

CS6220-Assignment3

Problem2. Hand on Experience with Supervised (Classification) Ensemble

Bin Xie

Overview

For this assignment, I will choose the following five different classification algorithms: Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Supported Vector Machine (SVM) and Multi-layer Perceptron (MLP). For the ensemble methods, I choose bagging technique on my best decision tree, which uses Bagging Classifier (BC) to explore how much that cluster ensemble can improve the learning accuracy.

Dataset

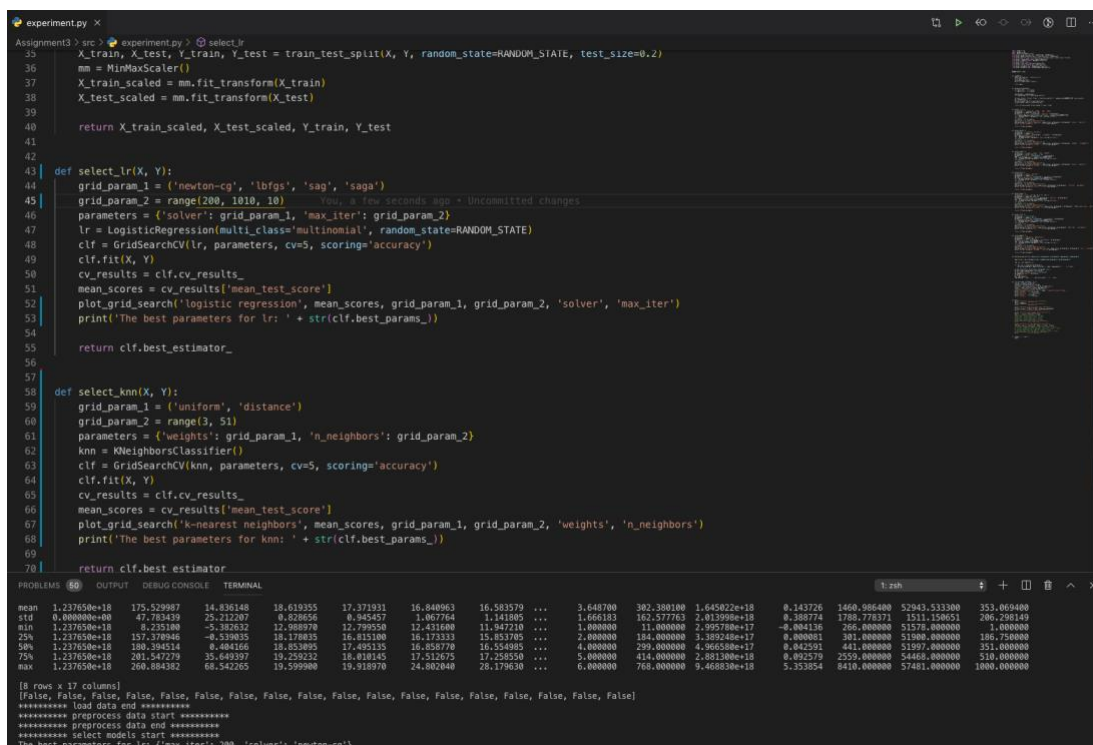
Sloan Digital Sky Survey DR14 dataset

This dataset is sourced from [kaggle.com](https://www.kaggle.com/dhansh/SDSS-DR14). It consists of 10,000 subjects of space taken by the SDSS. Each subject is described by 17 feature columns and 1 class column which identifies it to be either a star, galaxy or quasar. All the features are real value and class column is object. It's an inspiring and interesting multi-classification problem.

Experiment Environment

- Operating System: Mac OSX
- Language: Python3.5
- Packages: numpy, pandas, matplotlib, sklearn

The screenshot of the execution program:



```
experiment.py X
Assignment3 > src > experiment.py > select lr
25 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=RANDOM_STATE, test_size=0.2)
26 mm = MinMaxScaler()
27 X_train_scaled = mm.fit_transform(X_train)
28 X_test_scaled = mm.fit_transform(X_test)
29
30 return X_train_scaled, X_test_scaled, Y_train, Y_test
31
32
33 def select_lr(X, Y):
34     grid_param_1 = ('newton-cg', 'lbfgs', 'sag', 'saga')
35     grid_param_2 = range(200, 1010, 10)
36     parameters = {'solver': grid_param_1, 'max_iter': grid_param_2}
37     lr = LogisticRegression(multi_class='multinomial', random_state=RANDOM_STATE)
38     clf = GridSearchCV(lr, parameters, cv=5, scoring='accuracy')
39     clf.fit(X, Y)
40     cv_results = clf.cv_results_
41     mean_scores = cv_results['mean_test_score']
42     plot_grid_search('logistic regression', mean_scores, grid_param_1, grid_param_2, 'solver', 'max_iter')
43     print('The best parameters for lr: ' + str(clf.best_params_))
44
45     return clf.best_estimator_
46
47
48 def select_knn(X, Y):
49     grid_param_1 = ('uniform', 'distance')
50     grid_param_2 = range(3, 51)
51     parameters = {'weights': grid_param_1, 'n_neighbors': grid_param_2}
52     knn = KNeighborsClassifier()
53     clf = GridSearchCV(knn, parameters, cv=5, scoring='accuracy')
54     clf.fit(X, Y)
55     cv_results = clf.cv_results_
56     mean_scores = cv_results['mean_test_score']
57     plot_grid_search('k-nearest neighbors', mean_scores, grid_param_1, grid_param_2, 'weights', 'n_neighbors')
58     print('The best parameters for knn: ' + str(clf.best_params_))
59
60     return clf.best_estimator_
61
62
63 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
64
65 mean 1.237650e+18 175.529987 14.436148 18.619355 17.371931 16.840963 16.583579 ... 3.648700 302.380180 1.645822e+18 0.143726 1460.986400 52943.533300 353.069400
66 std 0.000000e+00 47.782429 25.112207 8.828656 8.845457 1.407764 1.141895 ... 1.666183 162.377763 2.491399e+18 0.380774 1760.778171 1511.150651 206.288149
67 min 1.237650e+18 8.235100 -5.382632 12.988978 12.799550 12.431600 11.947210 ... 1.000000 11.000000 2.995780e+17 -0.004136 266.000000 51578.000000 1.000000
68 25% 1.237650e+18 157.378946 -4.534035 18.178835 16.815100 16.173333 15.853785 ... 2.000000 184.000000 3.393246e+17 0.000001 301.000000 51900.000000 185.750000
69 50% 1.237650e+18 180.384514 0.404166 18.853895 17.459135 16.433770 16.545085 ... 4.000000 299.000000 4.865580e+17 0.842591 441.000000 51597.000000 351.000000
70 75% 1.237650e+18 201.547279 35.649397 19.259232 18.018145 17.512675 17.258550 ... 5.000000 414.000000 2.881380e+18 0.092579 2559.000000 54468.000000 510.000000
71 max 1.237650e+18 260.684382 68.542265 19.599980 19.918978 24.802840 26.179630 ... 6.000000 768.000000 9.468530e+18 5.353854 8410.000000 57481.000000 1000.000000
72
73 [8 rows x 17 columns]
74 [False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False]
75 ***** load data end *****
76 ***** preprocess data start *****
77 ***** preprocess data end *****
78 ***** select models start *****
79 The best parameters for lr: {'max_iter': 200, 'solver': 'newton-cg'}
```

Quality Metrics

For the quality metrics, I will use accuracy, precision, recall and F1-score.

Accuracy is the most intuitive quality metric. It is just a ratio of correctly predicted subjects to the total subjects.

Precision is the ratio of correctly predicted positive subjects to the total predicted positive subjects. Precision is how sure you are of the true positives in the dataset.

Recall (also sensitivity) is the ratio of correctly predicted positive subjects to the all subjects in actual class. Recall is how sure you are that you are not missing any positives.

F1 Score is the harmonic average of the precision and recall. It takes both false positives and false negatives into account. It works best if the class distribution is uneven.

In a word, accuracy is a great measure if the false positives and false negatives have similar cost. However, if the dataset has the uneven class distribution, it's better to look at the precision and recall. F1 Score takes the balance of precision and recall.

Here, since my problem is a multiclass classification task, I need to calculate the average of the precision, recall and f1-score for three classes. Also, since the distribution of these three classes are almost even, I use the macro precision, macro recall and macro f1-score.

Data Preprocessing

At first, I check the missing value in the dataset. The dataset is clean and good. No missing value or abnormal value exist.

Then, for the label, since the original label is in the string object type, I use the LabelEncoder function to encode the three labels: [star, galaxy, quasar] to value [0, 1, 2].

After that, I find that the values of different features vary in range. For example, the range of the values for feature "mjd" is 0~57481. However, the range of the values for feature "camcol" is 0~6. Therefore, to avoid that some features would outweigh the rest of the features, I used min-max scalar to scale the data into the range 0~1. Also, the min-max normalization can make the models converge quickly.

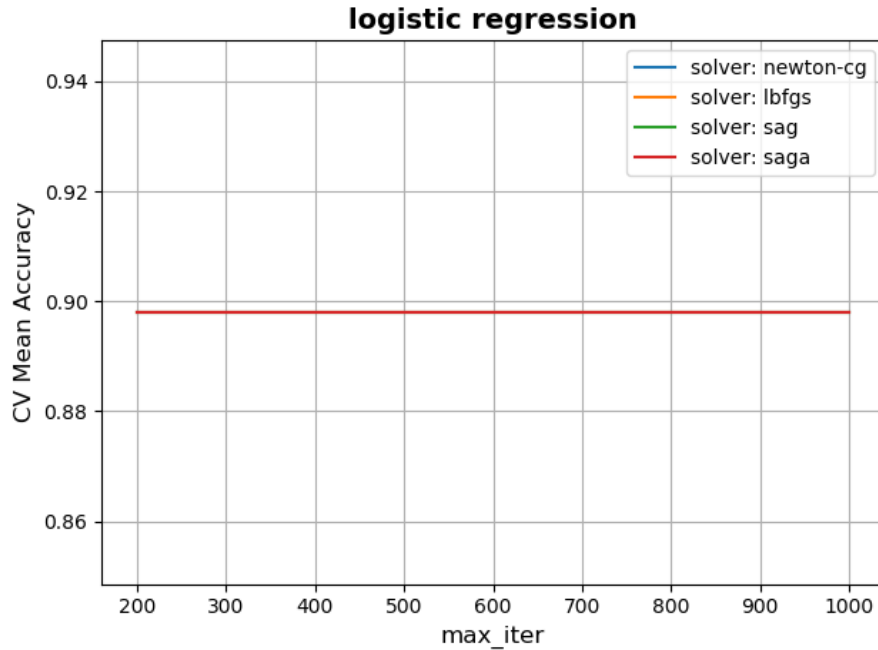
Finally, when splitting the data, the ratio of training data and test data is 8:2.

Models and Parameters

When training the models and selecting the best parameters, in order to make full use of our dataset and keep the generalization ability of our models, I use the cross validation and grid search to train the model and choose the best parameter combinations. For model selection, I only use the accuracy as the evaluation metric.

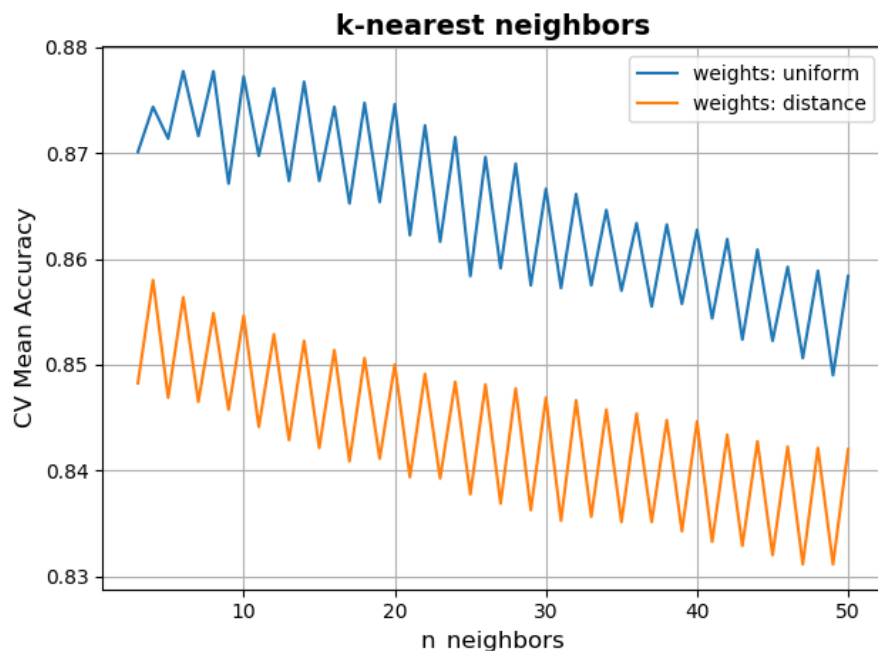
For the Logistic Regression classifier, I prune the parameters of "solver" in {'newton-cg', 'lbfgs', 'sag', 'saga'} and "max_iter" from 200 to 1000. Here, "solver" is the optimization algorithm for loss function and "max_iter" is the maximum number of iterations taken for the solvers to converge.

From the learning curve, I can find that the logistic model is already converged when the number of iterations reaches 200 and it looks like that the "solver" parameter has no influence on the model so that all of four lines are almost overlapped. From the grid search, I can get the best parameters are: {'max_iter': 200, 'solver': 'newton-cg'} and the best average accuracy is 0.898.



For the K-Nearest Neighbor classifier, I prune the parameters of “weights” in {‘uniform’, ‘distance’} and “n_neighbors” from 3 to 50. Here, “weights” is the weight function used in prediction and “n_neighbors” is the number of neighbors to use for queries.

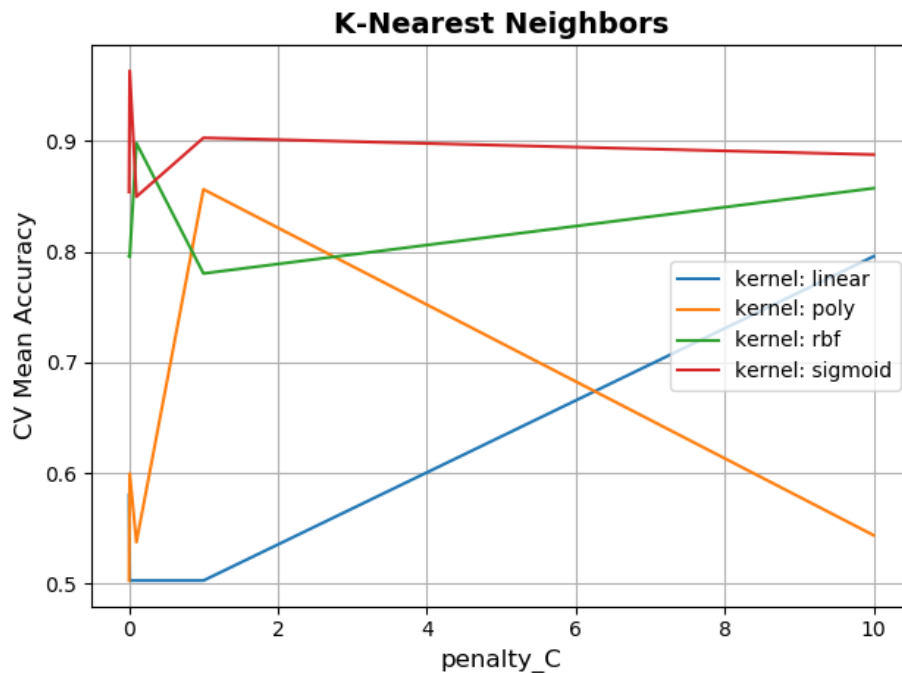
From the learning curve, I can find that uniform weight always has the better performance than distance weight and the accuracy decreases with the increasing of n_neighbors. From the grid search, I can get the best parameters are: {‘n_neighbors’: 4, ‘weights’: ‘uniform’} and the best average accuracy is 0.878.



For the Supported Vector Machine classifier, I prune the parameters of “kernel” in {‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’} and ‘C’ in {0.001, 0.01, 0.1, 1, 10}. Here, “kernel” is kernel type to be used in the algorithm and ‘C’ is Penalty parameter C of the error term.

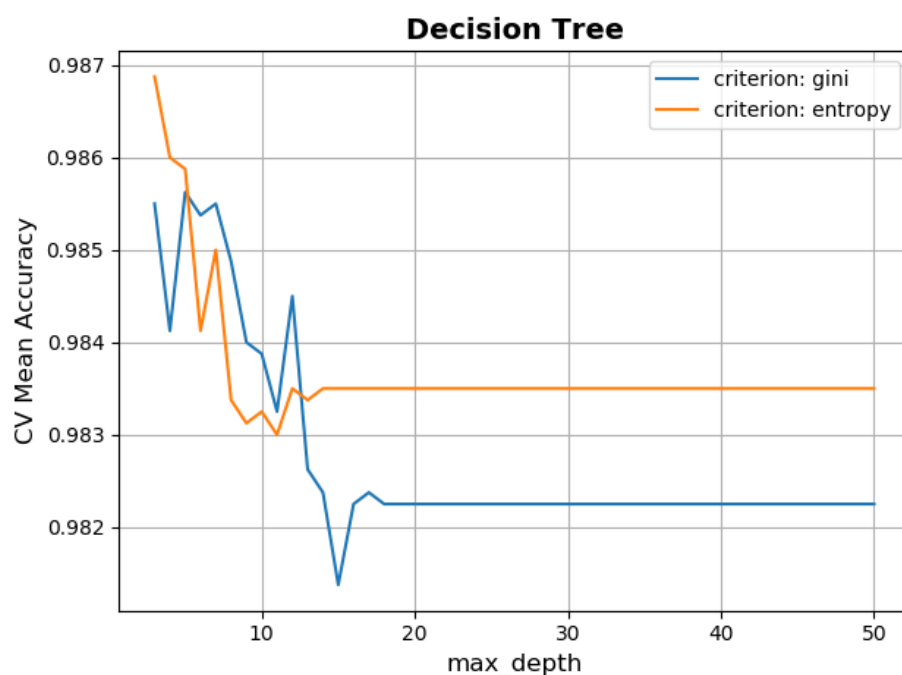
From the learning curve, I can find that when kernel is sigmoid, the SVM model has the

better performance than the rest of models with different kernels. And the different penalty C values really affect the performance of models a lot. From the grid search, I can get the best parameters are: {'C': 0.01, 'kernel': 'sigmoid'} and the best average accuracy is 0.963.



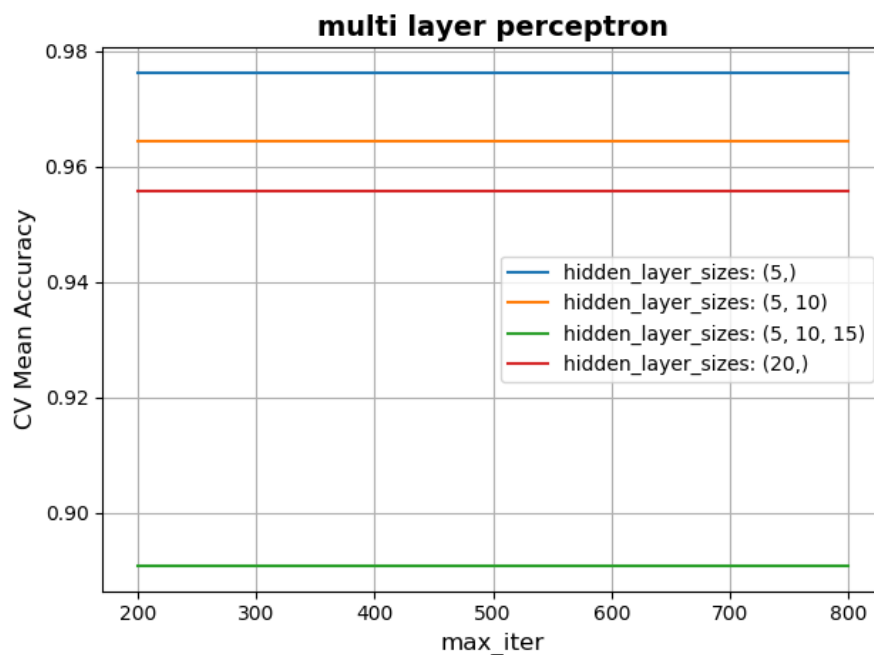
For the Decision Tree, I prune the parameters of “criterion” in {“gini”, “entropy”} and “max_depth” from 3 to 50. Here, “criterion” is the function to measure the quality of a split and “max_depth” is the the maximum depth of the tree.

From the learning curve, I can find that when the max_depth of the tree increases, the decision tree models have poorer performance and when the max_depth is over 15 or 17, the max_depth has no influence on the models anymore. From the grid search, I can get the best parameters are: {'criterion': 'entropy', 'max_depth': 3} and the best average accuracy is 0.987.



For the Multi-layer Perceptron, I prune the parameters of “hidden_layer_sizes” in [(5,), (5, 10), (5, 10, 15), (20,)] and “max_iter” from 200 to 800. Here, “hidden_layer_sizes” is the structure of neural network where (5, 10, 15) means that there are 3 hidden layers and they have 5, 10, 15 neurons respectively, and “max_iter” is the maximum number of iterations. Also, to make the neural network converge quickly, I set the initial learning rate as 0.1 and adopt the adaptive learning rate mechanism to keep the performance of the model.

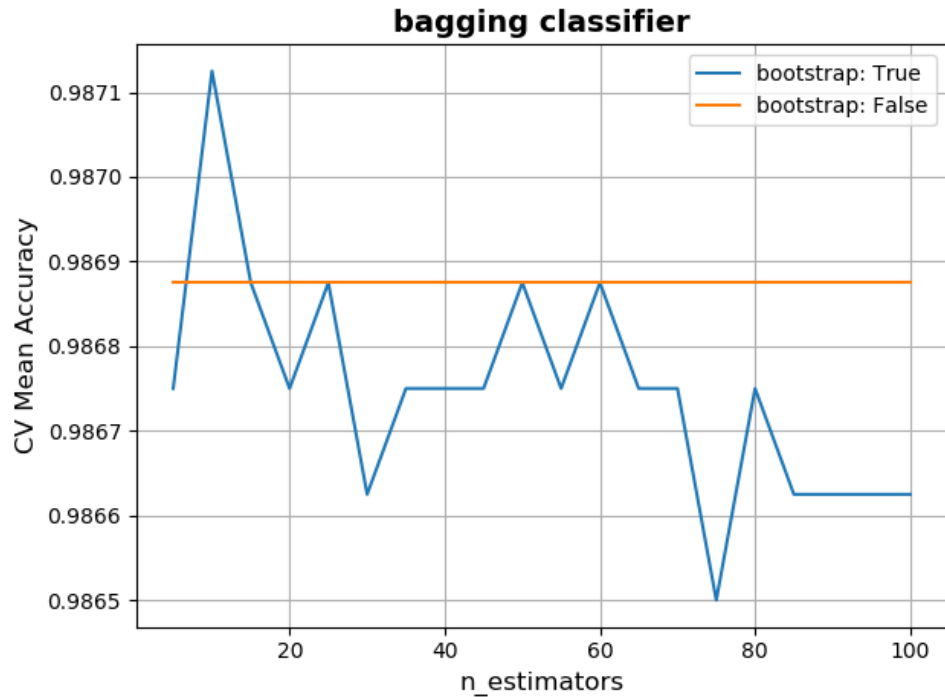
From the learning curve, I can find that all of the four different neural networks can converge when the number of iterations is over 200, and the number of hidden layer and number of neurons in each hidden layer affect the performance. From the grid search, I can get the best parameters are: {'hidden_layer_sizes': (5,), 'max_iter': 200} and the best mean accuracy is 0.976.



After I get the best parameters for the Decision Tree model, I use it as the weak estimator for the Bagging Classifier which adopts the bagging idea.

For the Bagging Classifier, I prune the parameters of “bootstrap” in {True, False} and “n_estimators” from 5 to 100. Here, “bootstrap” means whether bootstrap samples are used when building trees and “n_estimators” is the number of weak estimators, i.e. the number of my best decision tree.

From the learning curve, I can find that when bootstrap is False, the performance doesn't change with the increasing of number of estimators. And, when bootstrap is True, the model performs poorly when the number of estimators increases. From the grid search, I can get the best parameters are: {'bootstrap': True, 'n_estimators': 10} and the best average accuracy is 0.987.



Results

After I get the best model of each algorithm above, I use the test data to explore the performance of these models. The following table shows the quality metrics of these models after running the test data.

| Models | Accuracy | Macro Precision | Macro Recall | Macro F1-Score |
|--------|----------|-----------------|--------------|----------------|
| LR | 0.799 | 0.810 | 0.849 | 0.804 |
| KNN | 0.769 | 0.724 | 0.809 | 0.734 |
| SVM | 0.708 | 0.851 | 0.761 | 0.738 |
| MLP | 0.946 | 0.903 | 0.952 | 0.923 |
| DT | 0.980 | 0.946 | 0.974 | 0.959 |
| BC | 0.980 | 0.946 | 0.974 | 0.959 |

Analysis

- For the logistic regression model, it's a linear model and usually performs well for the binary classification task. Also, it can be adopted for multiclass classification task. In my experiment, I adopt the multinomial logistic regression which uses cross entropy loss function and performs well for the logically separate categories. During the grid search, the best model can reach the 0.898 accuracy and the test accuracy is 0.799. It looks like the generation ability of my best logistic regression model can still be improved. Also, I can find the macro precision, macro recall and macro F1-Score are all larger than accuracy. It shows that my logistic model has good ability to find the true positives in the dataset and missing few positives.
- For the k-nearest neighbor model, during the grid search, the best model can reach

the 0.878 accuracy and the test accuracy is 0.769. The generation ability of the model isn't so good either. Besides, I can find the macro precision and F1-score is lower than the accuracy and the macro recall is larger than the accuracy. It shows that in our prediction results, there are many false positives. However, it has good ability to find the positives from the sample.

- For the supported vector machine model, during the grid search, the best model can reach the 0.962 accuracy, but the test accuracy is only 0.708. The generation ability of this model is the worst among all the models. Also, I can find that the macro precision is much higher than accuracy and macro recall as well as macro F1-Score are higher than accuracy. It shows that the model has really good capacity to find the true positives in the prediction results.
- For the decision tree model, during the grid search, the best model can reach the 0.987 accuracy and the test accuracy is 0.980. This is the best model among all my models. The generation ability of the model is excellent. Also, all of the accuracy, macro precision, macro recall and F1-Score are relatively high and similar.
- For the multi-layer perceptron model, during the grid search, the best model can reach the 0.976 accuracy and the test accuracy is 0.946. This is also a pretty good model with great generation ability and stability. Since I only try 4 different neural network structures, there is still much space for the multi-layer perceptron model to improve. However, the training time for the multi-layer perceptron is much longer than decision tree. So, decision tree is still the best model for this problem.
- For the ensemble method, I choose the bagging. Bagging will fit each weak classifier on random subsets of the original dataset and then aggregate the individual prediction results of each weak estimator by voting to give the final prediction. In this way, it can reduce the variance of the weak estimator like decision tree by adopting randomization during construction and making an ensemble during training and prediction. It's a good ensemble method to understand and adopted for decision tree. From the result, it's surprised to find that the test accuracy, macro precision, macro recall and macro F1-score are the same as the results of decision tree. It looks like the ensemble method has no improvement of my original decision tree model. For the possible reason, I think, it may because the decision tree model already does excellent job for this problem and the improvement of accuracy may be very little like 0.00005 so that it doesn't show up explicitly. For another reason, since the bagging can reduce the variance of the weak estimator, maybe in the results of my best decision tree, the variance is very small, and the bias is relatively high. So, the bagging has little improvement of my best decision tree model. Maybe I can find some ensemble method which can reduce the bias of weak estimator.

Conclusions

- For this sky subjects multiclass classification problem, the best model is the decision tree with high accuracy, macro accuracy, macro precision and macro F1-score as well as low time cost.
- Data preprocessing is very important to construct the features. We can do data cleaning, normalization and so on. Further, it may be a good idea if I can use some dimensionality reduction skills like Random Projection, PCA, ICA and so on to reduce the feature dimensions or remove unrelated features.
- Model selection and parameters pruning are really important but time-cost work. If the size of dataset is large enough, we can split the data to training set, validation set and test set simply. If the size of the dataset is small, to make full use of data, cross

validation is a good practice. Also, to try different combinations of parameters, grid search can help us a lot.

- When evaluating the models, accuracy is not enough. To have more comprehensible evaluation of the models, we need to combine the accuracy with other metrics like precision, recall, f1-score, auc-curve and so on.
- For the ensemble method, it cannot guarantee for the improvement of the weak estimators. We need to really understand the mechanism of the specific ensemble method and adopt suitable ensemble method to the different machine learning algorithms and dataset.