Jashan Shah

11/6/2022

Dr. Vibhav Gogate

Report Part 1

Part 1: Tree Classifiers

1. (15 points) Use the sklearn.tree.DecisionTreeClassifier on the 15 datasets. Use the validation set to tune the parameters (see the documentation for parameters; e.g., criterion, splitter, max depth, etc.). After tuning the parameters, mix the training and validation sets, relearn the decision tree using the “best parameter settings found via tuning” and report the accuracy and F1 score on the test set. For each dataset, also report the “best parameter settings found via tuning.” (Note that we expect that no two students will have the same “best parameter settings”’).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Set Name | Hyperparameters | Accuracy on Validation | Accuracy on Testing | F1 Score |
| 300\_d100 | {'criterion': 'entropy', 'max\_depth': 30, 'max\_features': 'sqrt', 'min\_samples\_split': 3, 'splitter': 'best'} | 0.6549 | 0.52 | 0.5636 |
| 300\_d1000 | {'criterion': 'entropy', 'max\_depth': 15, 'max\_features': None, 'min\_samples\_split': 5, 'splitter': 'best'} | 0.6525 | 0.6715 | 0.6726 |
| 300\_d5000 | {'criterion': 'entropy', 'max\_depth': 10, 'max\_features': None, 'min\_samples\_split': 7, 'splitter': 'random'} | 0.738 | 0.7803 | 0.7882 |
| 500\_d100 | {'criterion': 'gini', 'max\_depth': 35, 'max\_features': None, 'min\_samples\_split': 2, 'splitter': 'random'} | 0.6799 | 0.635 | 0.6572 |
| 500\_d1000 | {'criterion': 'gini', 'max\_depth': 5, 'max\_features': None, 'min\_samples\_split': 2, 'splitter': 'random'} | 0.6925 | 0.682 | 0.6870 |
| 500\_d5000 | {'criterion': 'gini', 'max\_depth': 10, 'max\_features': None, 'min\_samples\_split': 5, 'splitter': 'random'} | 0.7434 | 0.7904 | 0.7998 |
| 1000\_d100 | {'criterion': 'gini', 'max\_depth': 5, 'max\_features': None, 'min\_samples\_split': 3, 'splitter': 'random'} | 0.7699 | 0.715 | 0.7298 |
| 1000\_d1000 | {'criterion': 'entropy', 'max\_depth': 15, 'max\_features': None, 'min\_samples\_split': 15, 'splitter': 'random'} | 0.795 | 0.806 | 0.8077 |
| 1000\_d5000 | {'criterion': 'entropy', 'max\_depth': 10, 'max\_features': None, 'min\_samples\_split': 7, 'splitter': 'random'} | 0.8432 | 0.8573 | 0.8629 |
| 1500\_d100 | {'criterion': 'entropy', 'max\_depth': 5, 'max\_features': 'log2', 'min\_samples\_split': 3, 'splitter': 'best'} | 0.8549 | 0.76 | 0.7692 |
| 1500\_d1000 | {'criterion': 'gini', 'max\_depth': None, 'max\_features': None, 'min\_samples\_split': 15, 'splitter': 'best'} | 0.9039 | 0.911 | 0.9134 |
| 1500\_d5000 | {'criterion': 'entropy', 'max\_depth': 25, 'max\_features': None, 'min\_samples\_split': 3, 'splitter': 'random'} | 0.9425 | 0.9567 | 0.9571 |
| 1800\_d100 | {'criterion': 'gini', 'max\_depth': 5, 'max\_features': None, 'min\_samples\_split': 11, 'splitter': 'best'} | 0.9650 | 0.935 | 0.9365 |
| 1800\_d1000 | {'criterion': 'entropy', 'max\_depth': 45, 'max\_features': None, 'min\_samples\_split': 11, 'splitter': 'random'} | 0.9625 | 0.9715 | 0.9717 |
| 1800\_d5000 | {'criterion': 'entropy', 'max\_depth': 50, 'max\_features': None, 'min\_samples\_split': 11, 'splitter': 'random'} | 0.9856 | 0.9873 | 0.9873 |

1. (15 points) Repeat the experiment described above using: sklearn.ensemble.BaggingClassifier with “DecisionTreeClassifier” as the base estimator. Again, use the validation set to tune the parameters, mix training and validation after tuning to learn a new classifier and report (a) Best parameter settings after tuning and (b) Classification accuracy and F1 score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Set Name | Hyperparameters | Accuracy on Validation | Accuracy on Testing | F1 Score |
| 300\_d100 | {'criterion': 'gini', 'max\_depth': 33, 'max\_features': 'sqrt', 'min\_samples\_split': 15, 'splitter': 'best'} | 0.635 | 0.59 | 0.59 |
| 300\_d1000 | {'criterion': 'entropy', 'max\_depth': 5, 'max\_features': None, 'min\_samples\_split': 2, 'splitter': 'best'} | 0.6505 | 0.6725 | 0.7095 |
| 300\_d5000 | {'criterion': 'entropy', 'max\_depth': 10, 'max\_features': None, 'min\_samples\_split': 11, 'splitter': 'random'} | 0.7388 | 0.7798 | 0.7876 |
| 500\_d100 | {'criterion': 'gini', 'max\_depth': 15, 'max\_features': None, 'min\_samples\_split': 3, 'splitter': 'best'} | 0.685 | 0.65 | 0.6788 |
| 500\_d1000 | {'criterion': 'gini', 'max\_depth': 5, 'max\_features': None, 'min\_samples\_split': 15, 'splitter': 'random'} | 0.6925 | 0.682 | 0.6870 |
| 500\_d5000 | {'criterion': 'gini', 'max\_depth': 10, 'max\_features': None, 'min\_samples\_split': 11, 'splitter': 'random'} | 0.7428 | 0.7911 | 0.8009 |
| 1000\_d100 | {'criterion': 'entropy', 'max\_depth': 15, 'max\_features': 'sqrt', 'min\_samples\_split': 2, 'splitter': 'random'} | 0.7799 | 0.71 | 0.7184 |
| 1000\_d1000 | {'criterion': 'entropy', 'max\_depth': 50, 'max\_features': None, 'min\_samples\_split': 15, 'splitter': 'random'} | 0.7929 | 0.8005 | 0.8021 |
| 1000\_d5000 | {'criterion': 'entropy', 'max\_depth': 10, 'max\_features': None, 'min\_samples\_split': 2, 'splitter': 'random'} | 0.8423 | 0.8532 | 0.8593 |
| 1500\_d100 | {'criterion': 'entropy', 'max\_depth': None, 'max\_features': 'sqrt', 'min\_samples\_split': 3, 'splitter': 'random'} | 0.85 | 0.835 | 0.8374 |
| 1500\_d1000 | {'criterion': 'gini', 'max\_depth': None, 'max\_features': None, 'min\_samples\_split': 15, 'splitter': 'random'} | 0.9039 | 0.9165 | 0.9186 |
| 1500\_d5000 | {'criterion': 'entropy', 'max\_depth': 30, 'max\_features': None, 'min\_samples\_split': 7, 'splitter': 'random'} | 0.942 | 0.9555 | 0.9556 |
| 1800\_d100 | {'criterion': 'gini', 'max\_depth': 30, 'max\_features': None, 'min\_samples\_split': 15, 'splitter': 'random'} | 0.9650 | 0.96 | 0.9595 |
| 1800\_d1000 | {'criterion': 'entropy', 'max\_depth': 33, 'max\_features': None, 'min\_samples\_split': 5, 'splitter': 'random'} | 0.9605 | 0.9745 | 0.9747 |
| 1800\_d5000 | {'criterion': 'entropy', 'max\_depth': 33, 'max\_features': None, 'min\_samples\_split': 11, 'splitter': 'best'} | 0.9865 | 0.9871 | 0.9871 |

1. (15 points) Repeat the experiment described above using: sklearn.ensemble.RandomForestClassifier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Set Name | Hyperparameters | Accuracy on Validation | Accuracy on Testing | F1 Score |
| 300\_d100 | {'class\_weight': None, 'criterion': 'entropy', 'max\_depth': 50, 'max\_features': 'auto', 'min\_samples\_split': 7} | 0.78 | 0.78 | 0.78 |
| 300\_d1000 | {'class\_weight': 'balanced', 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'auto', 'min\_samples\_split': 4} | 0.867 | 0.858 | 0.8594 |
| 300\_d5000 | {'class\_weight': None, 'criterion': 'entropy', 'max\_depth': 10, 'max\_features': 'sqrt', 'min\_samples\_split': 9} | 0.8982 | 0.9137 | 0.9159 |
| 500\_d100 | {'class\_weight': 'balanced', 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'sqrt', 'min\_samples\_split': 9} | 0.8800 | 0.875 | 0.8780 |
| 500\_d1000 | {'class\_weight': 'balanced\_subsample', 'criterion': 'entropy', 'max\_depth': 5, 'max\_features': 'log2', 'min\_samples\_split': 7} | 0.9385 | 0.9475 | 0.9480 |
| 500\_d5000 | {'class\_weight': 'balanced\_subsample', 'criterion': 'entropy', 'max\_depth': 25, 'max\_features': 'sqrt', 'min\_samples\_split': 9} | 0.9475 | 0.9532 | 0.9534 |
| 1000\_d100 | {'class\_weight': 'balanced\_subsample', 'criterion': 'gini', 'max\_depth': 25, 'max\_features': 'log2', 'min\_samples\_split': 7} | 1.0 | 0.99 | 0.9898 |
| 1000\_d1000 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 50, 'max\_features': 'log2', 'min\_samples\_split': 9} | 0.998 | 0.9925 | 0.9924 |
| 1000\_d5000 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 30, 'max\_features': 'log2', 'min\_samples\_split': 9} | 0.9976 | 0.9971 | 0.9970 |
| 1500\_d100 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'auto', 'min\_samples\_split': 2} | 1.0 | 1.0 | 1.0 |
| 1500\_d1000 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'log2', 'min\_samples\_split': 4} | 1.0 | 1.0 | 1.0 |
| 1500\_d5000 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 10, 'max\_features': 'log2', 'min\_samples\_split': 4} | 1.0 | 0.9999 | 0.9998 |
| 1800\_d100 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'auto', 'min\_samples\_split': 2} | 1.0 | 1.0 | 1.0 |
| 1800\_d1000 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'auto', 'min\_samples\_split': 2} | 1.0 | 1.0 | 1.0 |
| 1800\_d5000 | {'class\_weight': None, 'criterion': 'gini', 'max\_depth': 5, 'max\_features': 'sqrt', 'min\_samples\_split': 9} | 1.0 | 0.9998 | 0.9998 |

1. (15 points) Repeat the experiment described above using: sklearn.ensemble.GradientBoostingClassifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Set Name | Hyperparameters | Accuracy on Validation | Accuracy on Testing | F1 Score |
| 300\_d100 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 35, 'max\_features': 'sqrt'} | 0.745 | 0.765 | 0.7729 |
| 300\_d1000 | {'learning\_rate': 0.3, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'auto'} | 0.913 | 0.984 | 0.9842 |
| 300\_d5000 | {'learning\_rate': 0.3, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'auto'} | 0.9947 | 0.9992 | 0.9992 |
| 500\_d100 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 35, 'max\_features': 'sqrt'} | 0.85 | 0.84 | 0.8476 |
| 500\_d1000 | {'learning\_rate': 0.3, 'loss': 'exponential', 'max\_depth': 7, 'max\_features': 'auto'} | 0.9385 | 0.9875 | 0.9875 |
| 500\_d5000 | {'learning\_rate': 0.3, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'auto'} | 0.9968 | 0.9995 | 0.9995 |
| 1000\_d100 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 13, 'max\_features': 'log2'} | 0.99 | 0.98 | 0.9801 |
| 1000\_d1000 | {'learning\_rate': 0.1, 'loss': 'exponential', 'max\_depth': 13, 'max\_features': 'log2'} | 0.9955 | 0.9935 | 0.9935 |
| 1000\_d5000 | {'learning\_rate': 0.3, 'loss': 'exponential', 'max\_depth': 7, 'max\_features': 'auto'} | 0.9991 | 0.9999 | 0.9999 |
| 1500\_d100 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'sqrt'} | 1.0 | 1.0 | 1.0 |
| 1500\_d1000 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'log2'} | 1.0 | 1.0 | 1.0 |
| 1500\_d5000 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'log2'} | 1.0 | 1.0 | 1.0 |
| 1800\_d100 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'sqrt'} | 1.0 | 1.0 | 1.0 |
| 1800\_d1000 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'sqrt'} | 1.0 | 1.0 | 1.0 |
| 1800\_d5000 | {'learning\_rate': 0.1, 'loss': 'deviance', 'max\_depth': 7, 'max\_features': 'auto'} | 1.0 | 0.9999 | 0.9999 |

1. (15 points) Record the classification accuracy and F1 score for each dataset and classifier (recall that we have four classifiers and we are using the best/tuned parameter settings for the classifier) in a table and then answer the following questions using the table:

• Which classifier (among the four) yields the best overall generalization accuracy/F1 score? Based on your ML knowledge, why do you think the “classifier” achieved the highest overall accuracy/F1 score.

Using the data that was collected from the table, Gradient Boosting was the best overall generalization accuracy/F1 score. If there is a lot of noise, then gradient boosting might not be the best choice because it could cause overfitting. At 2nd place, it would be Random Forest. Boosting and Bagging are overall better than decision tree.

• What is the impact of increasing the amount of training data on the accuracy/F1 scores of each of the four classifiers.

Generalization error is decreased when more examples are added. The model gets better by being more general when more examples are added. When the amount of training data increases, the accuracy and F1 scores of each classifier increases.

• What is the impact of increasing the number of features on the accuracy/F1 scores of each of the four classifiers.

When the amount of features increases, the accuracy and F1 scores of each of the four classifiers increases. Overfitting might occur when more input features are added. Overfitting will increase accuracy and F1 scores, but it is not good. The model needs to be generalized. Some of the features used may be redundant or irrelevant.

1. (Extra credit, 5 points) Evaluate the four tree classifiers you used above on the MNIST dataset from Homework 2 (do not compute F1 scores on MNIST, just classification accuracy). Which classifier among the four yields the best generalization accuracy on the MNIST dataset and why?

The accuracy of DT: 0.8787

The accuracy of Bagging: 0.945

The accuracy of RF: 0.9693

The accuracy of Boosting: 0.9459

The best accuracy is the Random Forest classifier as it has an accuracy of 0.9693.

The Random Forest Classifier performed the best in terms of accuracy. Gradient boosting is the 2nd best and bagging is at 3rd. There might be noise in the data which is why random foresting did better than gradient boosting.