Lucas Summers (lsumme01@calpoly.edu)
Xiuyuan Qiu (xiqiu@calpoly.edu)

CSC 466 Lab 3 Report

Implementation:

We implement a random forest model (RF) with our c45 cl;ass from Lab 2. We provide a rfEval script to perform a grid search on each of the three main RF hyperparameters. All decision trees in the grid search (both sklearn and our own) are set to use info_gain w/threshold 0.1, as we found that to produce nice trees (not overly simplistic) for all datasets in the previous lab.

Our random forest results are generally better than sklearn's with the info_gain threshold 0.1 that we chose. This can be attributed to sklearn's use of CART, which produces smaller trees than c45.

For efficiency, we refactored our c45 tree .fit() to use vectorized NumPy operations and arrays instead of the much slower pandas dataframe operations such as .unique() and .value_counts() and iloc[]. Because each decision tree is independent of the others in the forest, in our RF class we were also able to leverage parallel processing for both .fit() and .predict() methods via the **joblib** library. This produced a > 20x speed boost for larger datasets like nursery.csv, and puts our times for grid search in line with sklearn's implementation, barring cases where super large c45 trees are produced (see Letter results below). Note that sklearn's implementation is not parallel, but its optimized C-based operations still give it an advantage over ours.

Results:

Iris

	Our Implementation	sklearn
Grid search time	6 sec	5 sec
Test Accuracy	100%	100%
Train accuracy	95%	95.83%
Best Hyperparams	25 Trees 1 Attribute 15% of data per tree	25 Trees 1 Attribute 15% of data per tree

- Both sklearn and our implementation select the same hyperparameters in grid search in the same time with almost identical train and test accuracies
- 95% training accuracy (less than test accuracy) suggests that the model is not overfitting and is learning meaningful patterns in the data without unnecessary complexity
- Because of the simple and easily seperable nature of the dataset, they are both able to use a small forest of 1 attribute trees to perfectly predict the small test set.

Lab 2 accuracy: 94.67% — no real benefit
This suggests that Iris may not have enough complexity for Random Forest to provide a significant boost over a single well-optimized decision tree

Nursery

	Our Implementation	sklearn
Grid search time	14 sec	14 sec
Test Accuracy	87.35%	81.60%
Train accuracy	87.27%	82.05%
Best Hyperparams	100 Trees 4 Attributes 10% of data per tree	25 Trees 3 Attributes 5% of data per tree

- Test accuracy suggests that our method might generalize better for this dataset, however the higher train accuracy also suggests our model might be overfitting slightly more.
- Our selected hyperparameters all being larger suggest our Random Forest might be more diverse and capturing more complexity than sklearn's, leading to better generalization and results in this case.

Lab 2 accuracy: 91.23% - RF is marginally worse
This could be because the dataset has patterns that a single tree captures well, and adding more trees only increases variance rather than improving predictions. It could also be that our hardcoded c4.5 parameters are not optimal for this dataset and could be improved.

Letters

	Our Implementation	sklearn
Grid search time	16 min 41 sec	1 min 31 sec
Test Accuracy	94.20%	62.10%
Train accuracy	91.61%	62.69%
Best Hyperparams	500 Trees 5 Attributes 25% of data per tree	1000 Trees 3 Attributes 5% of data per tree

- In addition to test accuracy, the train accuracy difference is also large (91.61% vs. 62.69%), suggesting that sklearn's implementation is underfitting heavily
- Sklearn's 1000 trees with only 3 attributes and 5% data per tree suggests an attempt to prevent overfitting, but it may have been too conservative, leading to underfitting
- Our model likely learned better patterns with more data and attributes per tree
- This dataset might have high complexity, and a larger, deeper Random Forest might be necessary to learn effectively

NOTE: Our runtime is significantly longer than sklearn's, but that is because C45 produces large (~4500 node) trees on this dataset for the hyperparams we chose, while sklearn's CART algorithm stays relatively small (~20). This could also be a reason why sklearn's RF has much lower train/test accuracies than ours because it may not be capturing enough complexity due to overly simple trees.

Lab 2 accuracy: 86.62% - RF is better!
This suggests that ensemble methods truly help with complex datasets, where a single tree might struggle, but a combination of trees provides a more robust classification.

Heart

	Our Implementation	sklearn
Grid search time	32 sec	16 sec
Test Accuracy	87.50%	87.50%
Test Precision/Recall/F1	88.18 / 90.65 / 89.40	86.85 / 92.52 / 89.59
Train accuracy	86.24%	84.33%
Train Precision/Recall/F1	84.56 / 91.52 / 87.90	80.17 / 94.76 / 86.86
Best Hyperparams	120 Trees 5 Attributes 10% of data per tree	60 Trees 1 Attribute 20% of data per tree

- Both our implementation and sklearn achieved the same test accuracy as well as similar train accuracies
- Precision/Recall/F1 Scores are also similar for both models, showing that both implementations are performing mostly equivalently
- Both model have different ways of achieving similar results: sklearn used fewer trees but gave them more data (60 trees + 20%), while our method used more trees but gave them less data (120 trees + 10%)

Conclusion:

Overall, our implementation produced comparable or better results than sklearn's in terms of accuracy, but it had a higher runtime, especially on large datasets like Letters, mostly due to differences in the implementation of the underlying decision trees. On simpler datasets like Iris and Nursery, the Random Forest model provided little to no improvements over a single decision tree. But for larger and more complex datasets like Letters, the Random Forest model provided significant improvements by capturing more complexity in the data than is possible by just a single tree.