COL774: Machine Learning
Assignment 3

Devanshi Khatsuriya, 2019CS10344

Note: Refer to README.md for instructions to run code.

1 Decision Trees and Random Forests

(a) Implemented decision tree learning algorithm that uses mutual information as the criteria to split on. Trained two types of trees:

- Using multi-way splitting for categorical attributes.
  The plot of accuracy as the tree grows is shown in Fig. 1.

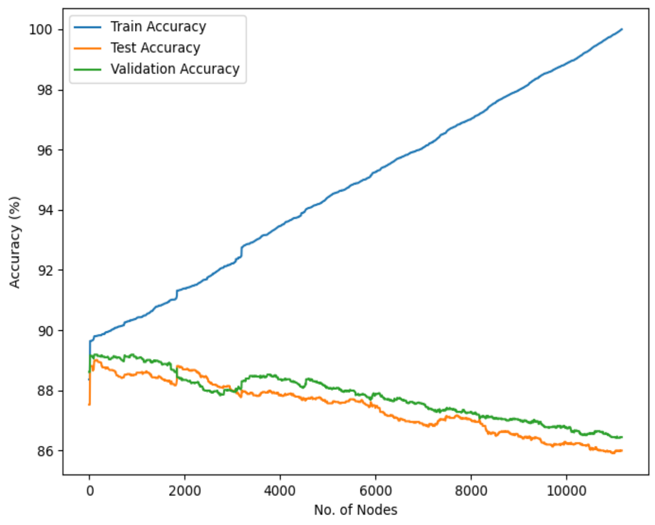

Figure 1: 1(a): Accuracies as tree grows (Multi-way splitting)

  - Train Set Accuracy: 100.0 %
  - Validation Set Accuracy: 86.44 %
  - Test Set Accuracy: 85.99 %
  - Time taken to grow Decision Tree: 128 seconds

- Using one-hot encoding to treat each categorical attribute as multiple boolean attributes.
  The plot of accuracy as the tree grows is shown in Fig. 2.
  - Train Set Accuracy: 100.0 %
  - Validation Set Accuracy: 87.59 %

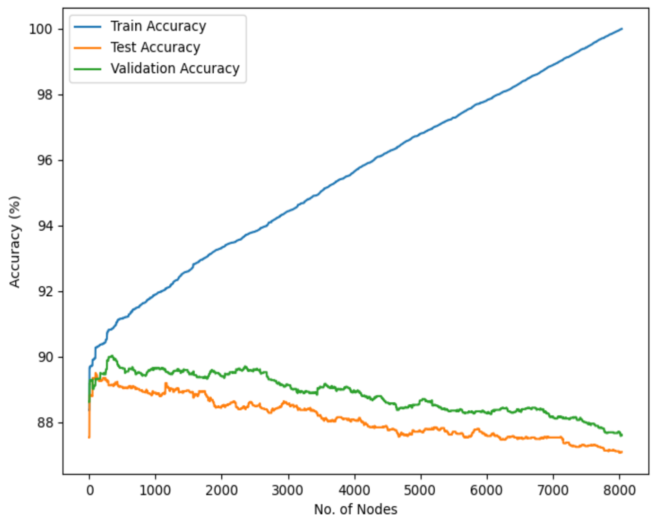

Figure 2: 1(a): Accuracies as tree grows (One-hot encoding)

- Test Set Accuracy: 87.08 %
- Time taken to grow Decision Tree: 422.46 seconds

It is observed that in both cases, the peaks of test and validation accuracies are achieved quite early, and after that further training leads to overfitting to training data noise.
Also, the test and validation accuracies are slightly better when using one-hot encodings for categorical attributes as compared to multi-way splitting, but the computational cost (time taken) is also more.

(b) Post-pruned the tree (that used one-hot encodings for categorical attributes) obtained in 1(a). Iteratively visited nodes in a bottom-up manner and pruned the tree below them if it increased prediction accuracy on validation set. The plot of accuracy as post-pruning progressed is shown in Fig. 3.

- Train Set Accuracy: 97.08 %
- Validation Set Accuracy: 92.61 %
- Test Set Accuracy: 88.47 %
- Time taken in Post Pruning: 153.15 seconds
- No. of nodes before and after post-pruning: 8035/5507

It is observed that post-pruning decision tree reduces overfitting to training data and improves validation and test set accuracies. It also reduces the number of nodes in the decision tree significantly.

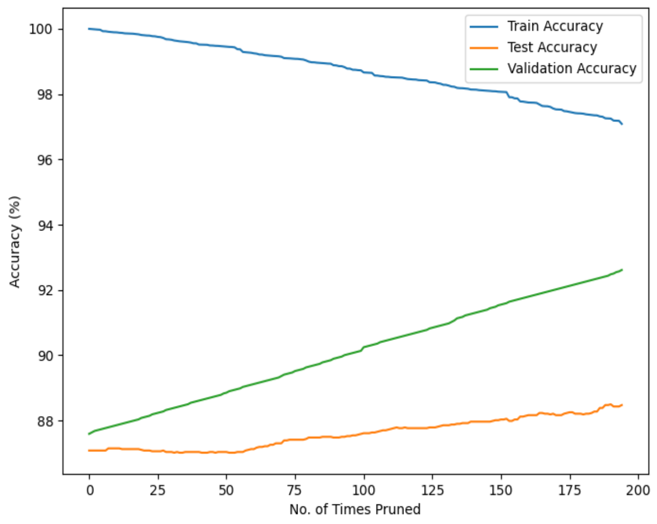

Figure 3: 1(b): Accuracies as post-pruning progresses

(c) Used sklearn library to train Random Forests Classifiers that use entropy as splitting criteria. Encoded categorical attributes as one-hot encodings and boolean attributes as 0/1. Performed a grid search over different values of parameters: *n_estimators*, *max_features* and *min_samples_split* to find the parameters that give the highest oob accuracy. The best parameter values obtained were: *n_estimators*=250, *max_features*=0.3 and *min_samples_split*=4. Corresponding accuracies were as follows:

- oob Accuracy: 90.80 %
- Validation Set Accuracy: 90.51 %
- Test Set Accuracy: 90.06 %

We see that the oob accuracy is more while the validation and test set accuracies are lesser compared to 1(b). This must be because this model didn't use the validation set but rather used only the out-of-bag-accuracy from the training set, because of which it is overfitting more to the training data compared to 1(b). On the other hand, it generalizes well as it performs better even on the test set.

(d) For the best parameters obtained in 1(c), varied each parameter while keeping others fixed to their best value. The plots are shown in Fig. 4, 5 and 6.

We observe that the best parameters based on oob score on training data also perform well on test and validation data. On varying any one parameter, the accuracies don't change much, which means the model is less sensitive to the values of each parameter given others are fixed to optimal.

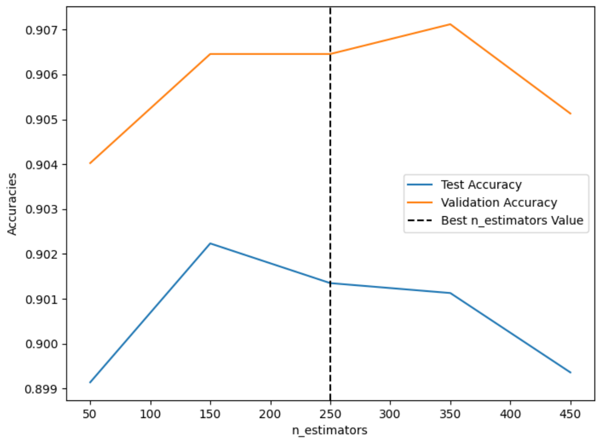

Figure 4: 1(d): Varying *n_estimators* value

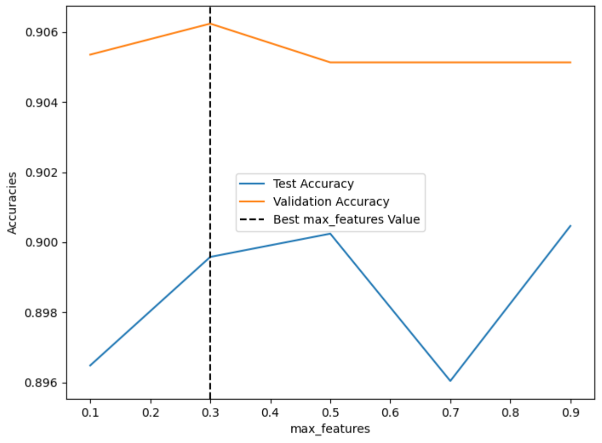

Figure 5: 1(d): Varying *max_features* value

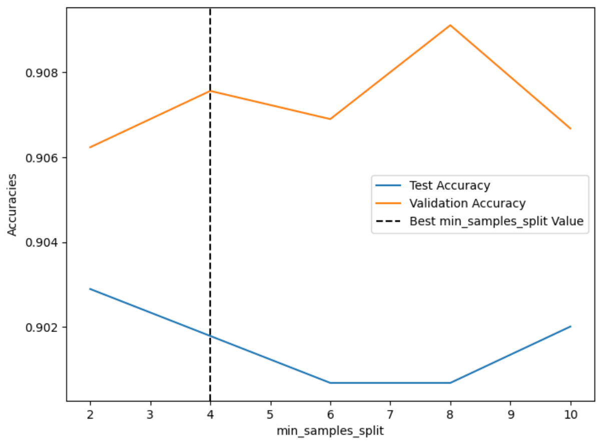

Figure 6: 1(d): Varying *min_samples_split* value

2 Neural Networks

(a) Converted all categorical attributes to multiple boolean attributes using their one-hot encodings.

(b) Implemented generic neural network architecture that does r-class multi-class classification from ordinal attributes using fully connected hidden layers. Uses mini-batch stochastic gradient to optimize MSE loss over mini batches.
Convergence criteria. Whenever iteration number is a multiple of 250, converged if the difference between average loss over the previous 250 iterations and the 250 iterations before that is less than 10^{-3}.

(c) Used 1 hidden layer architecture with size varying in $\{5, 10, 15, 20, 15\}$. The accuracy and time plots with varying hidden layer size are shown in Fig. 7 and 8.

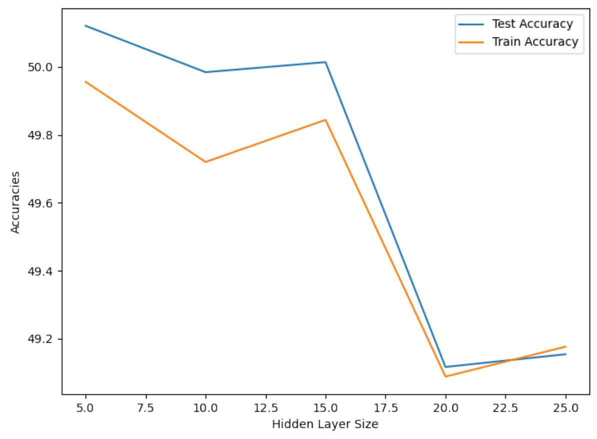

Figure 7: 2(c): Accuracy with varying hidden layer size

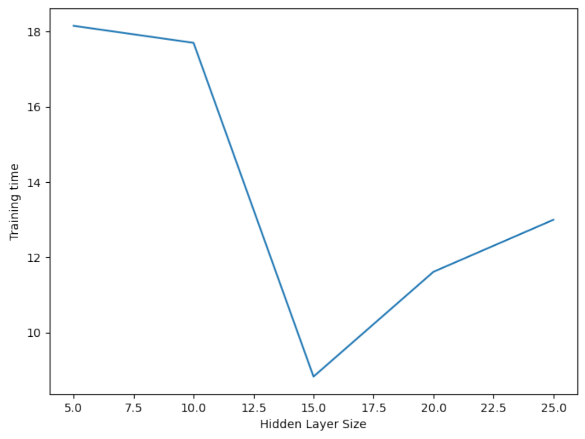

Figure 8: 2(c): Time taken with varying hidden layer size

(d) Used adaptive learning rate in addition to 2(c). The accuracy and time plots with varying hidden layer size are shown in Fig. 8 and 9.

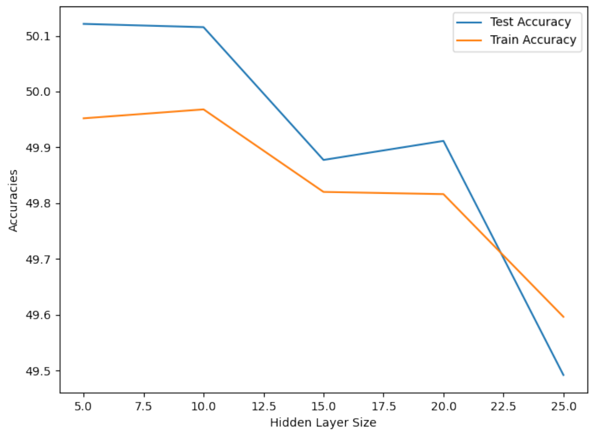

Figure 9: 2(d): Accuracy with varying hidden layer size (adaptive learning rate)

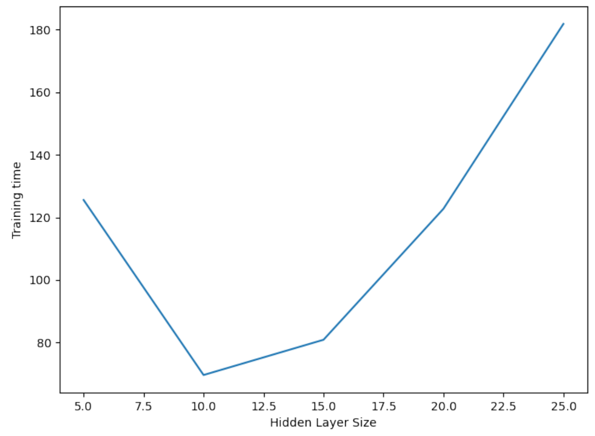

Figure 10: 2(d): Time taken with varying hidden layer size (adaptive learning rate)

(e) Implemented ReLU activation for hidden layers. Used hidden layer architecture with 2 hidden layers of size 100 each with adaptive learning rate. Trained two models:

(a) Sigmoid activation for all layers.

i. Train Accuracy: 49.95 %

ii. Test Accuracy: 50.12 %

iii. Training Time: 4790.63 seconds

(b) ReLU activation for hidden layers and sigmoid activation for output layers. (also used error threshold = 10^{-4} instead of 10^{-3})

i. Train Accuracy: 49.95 %

ii. Test Accuracy: 50.12 %

iii. Training Time: 3099.39 seconds

(f) Implemented the same architecture as 2(e) in sklearn library using MLPClassifier.

(a) Train Accuracy: 100 %

(b) Test Accuracy: 99.23 %

(c) Training Time: 708.31 seconds