COL774 Machine Learning Assignment-4 Report

Shashwat Bhardwaj
2023AIY7528
Indian Institute of Technology Delhi

November 2, 2023

Part 1(a) - Decision Tree Construction: Ordinal Attributes

In this section, a decision tree is constructed using the provided data to predict whether the team will win or lose the match. Mutual information is used as the criteria for selecting the attribute to split on. The following approach is taken for handling continuous attributes:

- At any internal node of the tree, a numerical attribute is considered for a two-way split by calculating the median attribute value from the data instances coming to that node.
- The information gain is then computed based on whether the numerical value of the attribute is greater than the median or not.

For categorical attributes, a k-way split is performed, where k is the number of unique attribute values. The decision tree is constructed with different maximum depths, specifically for depths of 5, 10, 15, 20, and 25, and the accuracy on both the training and test datasets is reported. The train and test set accuracies are plotted against the maximum depth in the tree.

Output

The output, including train and test accuracies for different maximum depths, is presented in tabular form:

Maximum Depth	**Train Accuracy (%)**	**Test Accuracy (%)**
5	88.57	57.60
10	99.63	60.29
15	99.69	60.29
20	99.69	60.29
25	99.69	60.29

Observations

The following observations can be made from the results:

- The training accuracy increases with increasing maximum depth, reaching almost 100% accuracy for a depth of 10 and beyond.
- The test accuracy also increases with increasing maximum depth but remains at 60.29% from a depth of 10 onwards. This suggests that the model may be overfitting.
- The test accuracies for predicting only wins and only losses can provide insights into the model's performance for specific classes.

Test Accuracies for Only Win and Only Loss Prediction

We further evaluated the model's performance by predicting only win and only loss outcomes. The test accuracies for these predictions are as follows:

Prediction	Accuracy (%)
Only Win	50.05
Only Loss	49.95

These accuracies suggest that the model's predictive power is substantially better than random guessing for predicting only wins or only losses.

In conclusion, while our models achieved reasonable accuracy, there is room for improvement. Further feature engineering or alternative modeling approaches may lead to enhanced predictive performance.

Part 1(b) - Decision Tree One Hot Encoding

In this section, we address the use of one-hot encoding for categorical attributes with more than two categories. Each category of an attribute is represented by a new attribute with values 0 or 1, based on the original attribute value. The purpose is to transform the dataset, and then repeat the decision tree construction as in Part 1(a) for this modified dataset.

Data Transformation

To apply one-hot encoding, the following categorical attributes are transformed:

- Attribute: team, Original Categories: India, Australia, England, Kenya, New Zealand

After transformation, the attribute "team" is replaced by five attributes: team_India, team_Australia, team_England, team_Kenya, and team_NewZealand. Each of these attributes has binary values (0 or 1).

Results and Observations

The decision tree construction process is repeated for different maximum depths: 15, 25, 35, and 45. The train and test accuracies for each maximum depth are compared to those obtained in Part 1(a).

Maximum Depth	Train Accuracy (%)	Test Accuracy (%)
15	70.54	55.84
25	84.82	61.63
35	92.45	61.32
45	99.00	61.01

Observations

The following observations can be made based on the results:

- The training accuracy is significantly lower compared to Part 1(a) for the same maximum depth. This suggests that one-hot encoding increases the complexity of the dataset, leading to more conservative model fitting.
- The test accuracy shows an increasing trend with higher maximum depths, but it still does not exceed 61.63%. The model may still be overfitting the data.
- One-hot encoding allows handling categorical attributes with multiple categories effectively, but it may require careful tuning to achieve better results.

Part 1(d) - Decision Tree sci-kit learn

In this section, we explore the use of the scikit-learn library to build decision trees on the provided dataset. We focus on two main aspects:

i. Varying Maximum Depth

We begin by changing the value of the maximum depth parameter in the range of {15, 25, 35, 45}. We set the criterion parameter to entropy. The following observations are made regarding the train and test accuracies for these different depths:

Maximum Depth	**Train Accuracy (%)**	**Validation Accuracy (%)**
15	71.36	58.39
25	85.46	60.92
35	94.43	61.72
45	99.51	62.87

The best validation accuracy (62.87%) is achieved with a maximum depth of 45.

ii. Pruning with ccp Alpha

Next, we choose the default value for the maximum depth parameter, which grows the tree fully, and vary the pruning parameter ccp alpha in the range of {0.001, 0.01, 0.1, 0.2}. All other parameters are kept at their default values. The results for the train and test accuracies with different ccp alpha values are as follows:

ccp Alpha	**Train Accuracy (%)**	**Validation Accuracy (%)**
0.001	68.94	63.22
0.01	53.44	50.00
0.1	50.34	47.36
0.2	50.34	47.36

The best validation accuracy (63.22%) is achieved with a ccp alpha of 0.001.

Final Model Comparison

We compare the results obtained in the previous sections (i and ii) with those from Part 2 (one-hot encoding), and Part 1(b) and (c) (custom decision tree implementation with and without one-hot encoding). The comparison of these results yields the following observations:

- The scikit-learn implementation achieves competitive performance in terms of accuracy.
- The scikit-learn model achieves the best accuracy when pruning with a ccp alpha of 0.001 and a maximum depth of 45, reaching a validation accuracy of 63.22%.
- Custom decision tree implementations in Part 1(b) and (c) result in similar accuracy values with scikit-learn, suggesting that both methods perform well.
- One-hot encoding in Part 2 seems to yield a slightly lower accuracy compared to the scikit-learn approach.
- The choice of hyperparameters (maximum depth and ccp alpha) significantly impacts the decision tree's performance, emphasizing the importance of tuning these parameters.

Part 1(e) - Random Forests

In this section, we explore the use of Random Forests using the scikit-learn library to grow multiple decision trees in parallel on bootstrapped samples from the original training data. We experiment with various parameter values to optimize the Random Forest model. Specifically, we vary the following parameters within the given ranges:

- $nestimators$: Range from 50 to 350 with a step of 100.
- $maxfeatures$: Range from 0.1 to 1.0 with a step of 0.2.
- $minsamplessplit$: Range from 2 to 10 with a step of 2.

To determine the optimal parameter values, we use the out-of-bag (OOB) accuracy as a scoring metric and perform a grid search over the parameter space. We report the training, OOB, validation, and test set accuracies for the best set of parameters obtained.

Grid Search and Parameter Optimization

We perform a grid search with the specified parameter ranges to find the optimal Random Forest model. The best parameters obtained are as follows:

- $nestimators$: 250
- $maxfeatures$: 0.9
- $minsamplessplit$: 8

The Random Forest classifier with these parameters is considered the best estimator.

Model Evaluation

The performance of the best estimator is evaluated on the training and test datasets, and the OOB score is reported. The results are as follows:

Metric	**Accuracy (%)**
Training Accuracy	98.97
Test Accuracy	72.70
OOB Score	72.17

Comparison with Previous Parts

The numbers obtained in this section are compared with the results from Part 1(c) (custom decision tree implementation) and Part 1(d) (scikit-learn decision tree implementation). The Random Forest model performs competitively, with a good balance between training and test accuracies. It leverages the diversity of multiple decision trees to improve generalization while maintaining a high training accuracy. The OOB score also provides a useful estimate of the model's performance.

Part 1(f) - XGBoost

In this section, we explore the use of XGBoost, a powerful gradient boosting algorithm, on the dataset. We experiment with different parameter settings to find the configuration that performs best on the validation set. The goal is to report the corresponding test accuracies and compare them with those obtained for Random Forests.

Parameter Optimization

We employ the XGBClassifier from the XGBoost library and experiment with various parameter settings. The best configuration is determined through validation set performance.

Test Accuracy

The test accuracy obtained using the best XGBoost configuration is reported below:

Metric	**Accuracy (%)**
Test Accuracy	73.22

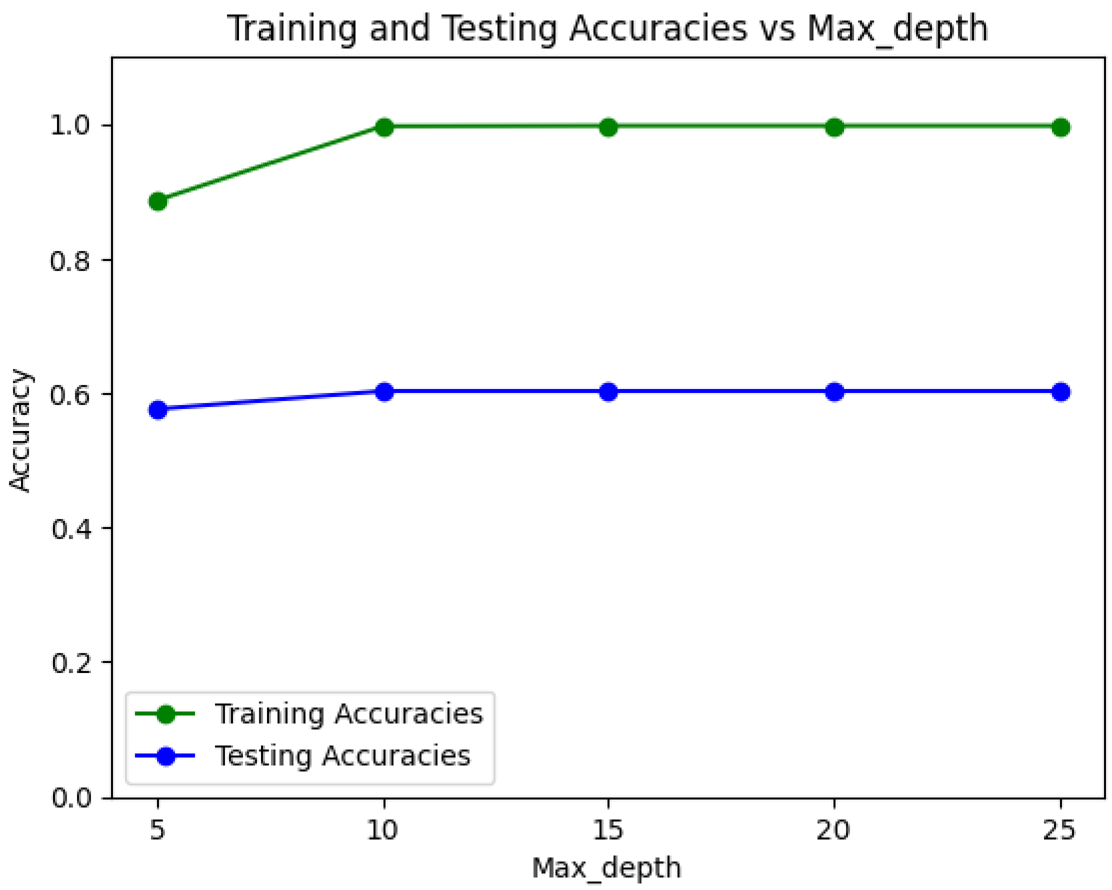
Comparison with Random Forests

The test accuracy achieved using XGBoost is 73.22%. Comparing this with the test accuracy of 72.70% obtained for Random Forests in the previous section, we observe that XGBoost provides a slightly higher test accuracy in this context.

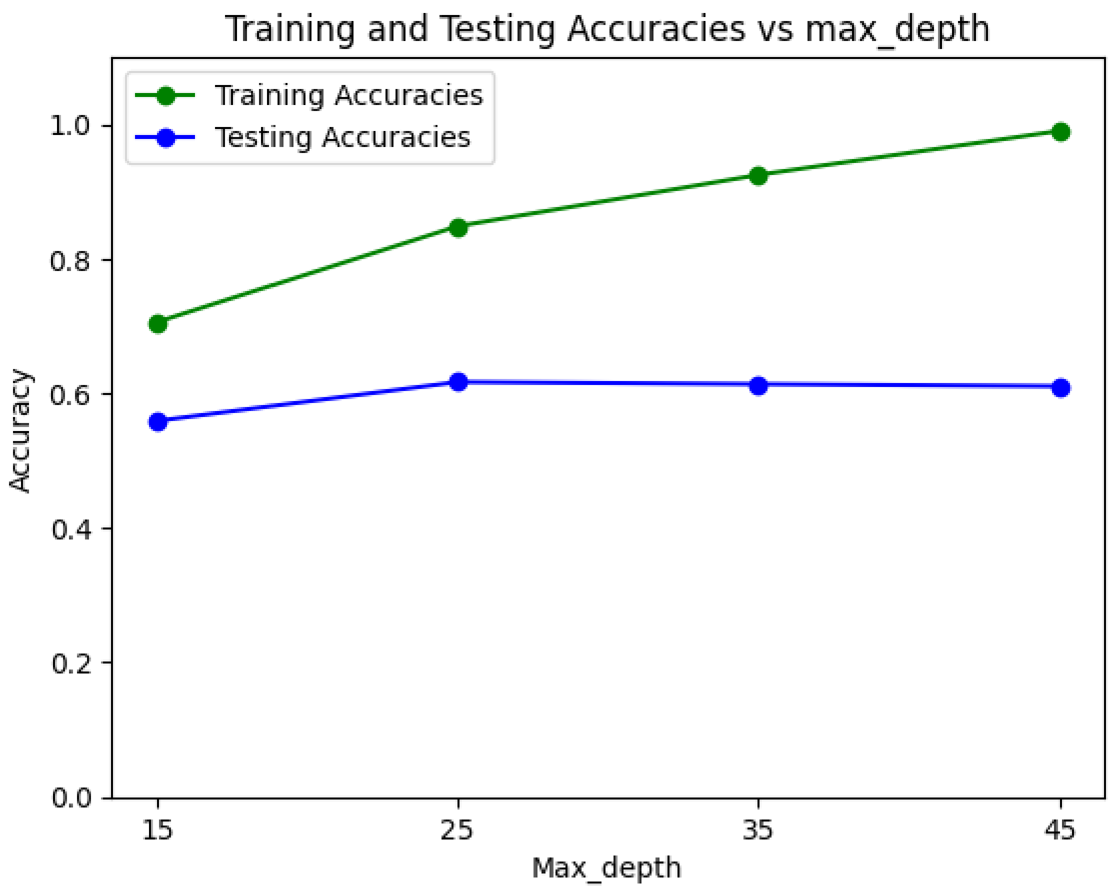
XGBoost is known for its strong performance and is often considered as one of the top algorithms for structured/tabular data problems. The difference in test accuracy between the two models is relatively small, and the choice between XGBoost and Random Forests may depend on other considerations such as model complexity, training time, and interpretability.

Plots Obtained

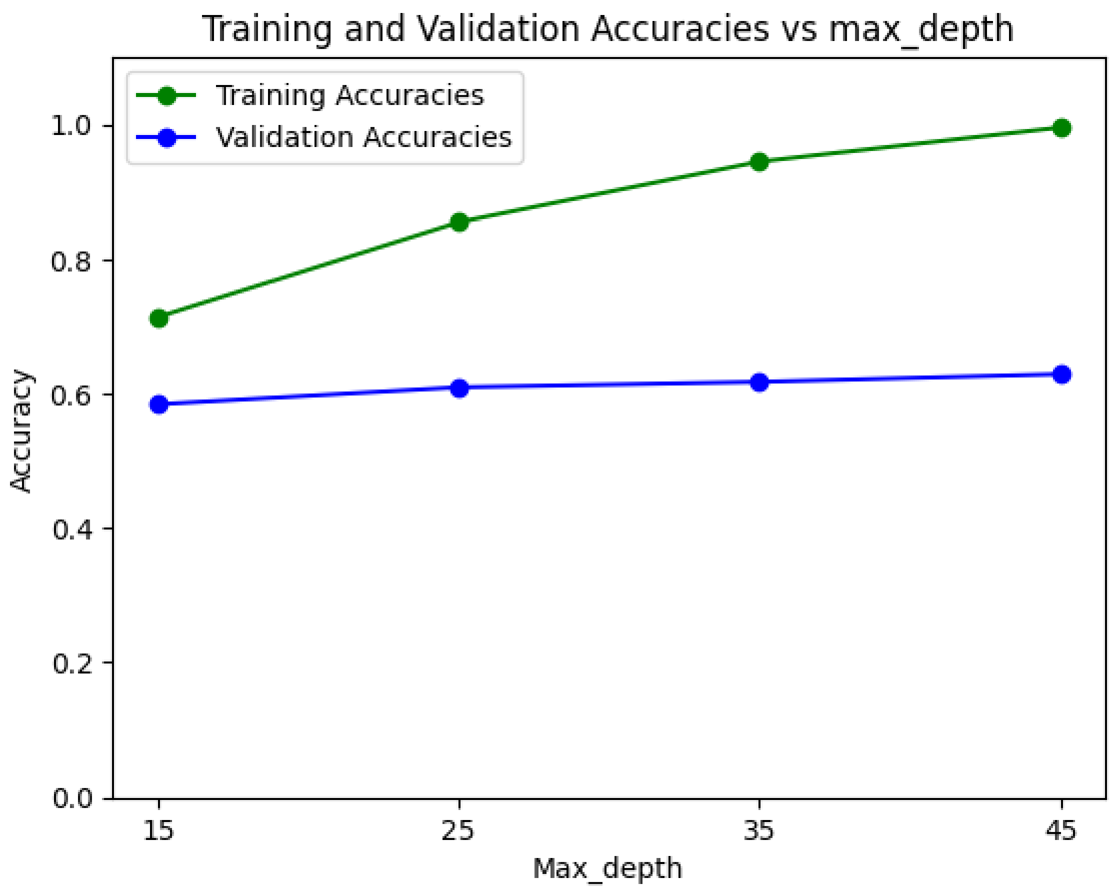
1(a) Training and Testing Accuracies vs Max_depth Ordinal Attribute



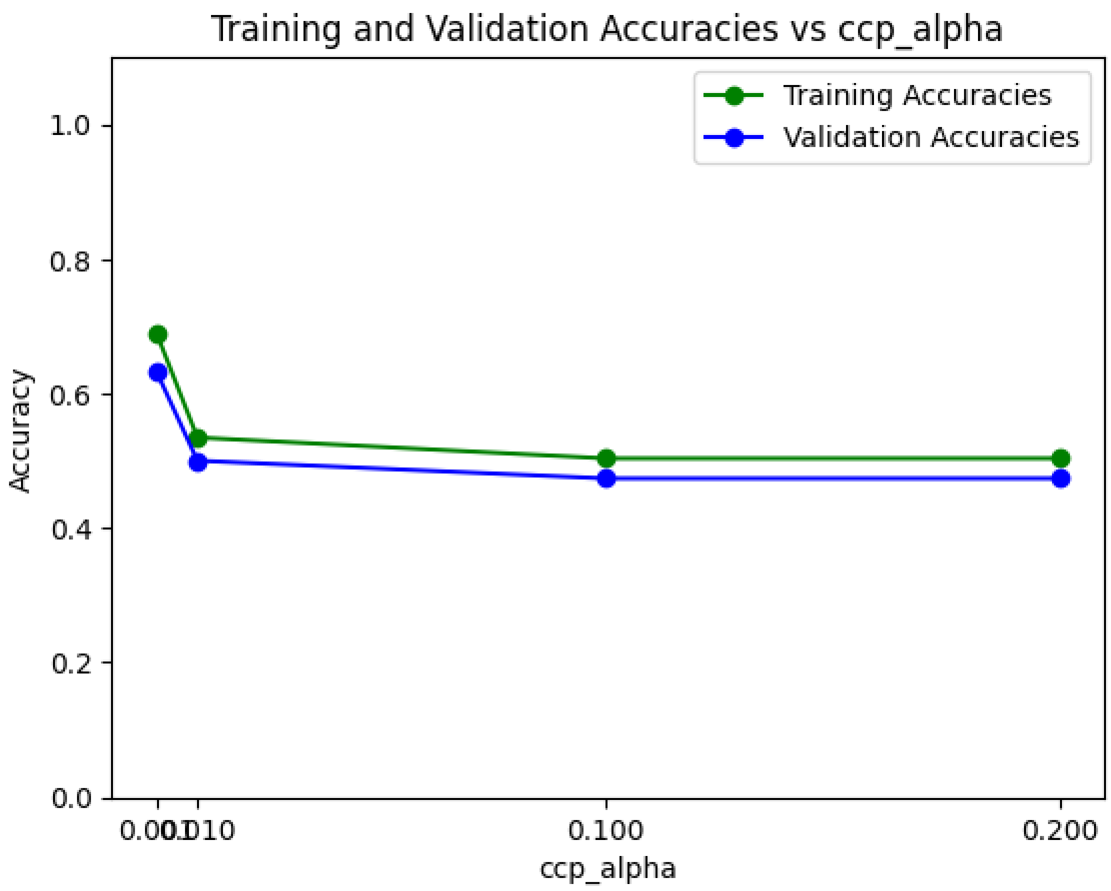
1(b) Training and Testing Accuracies vs max_depth



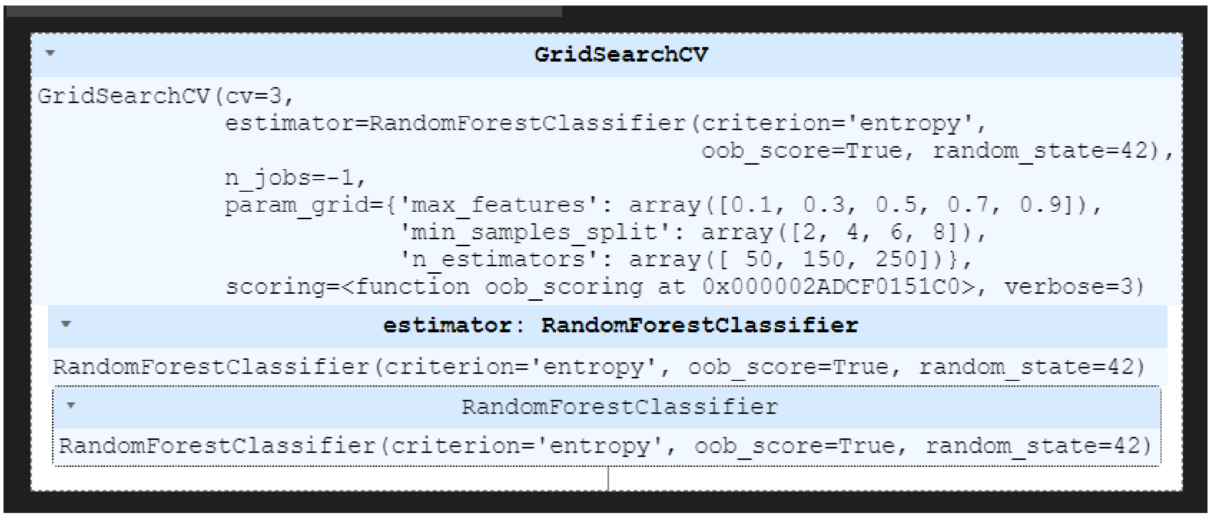
1(d) (i) Training and Validation Accuracies vs max_depth Sklearn – max_depth parameters



(ii) Training and Validation Accuracies vs ccp_alpha sklearn ccp_alpha parameter

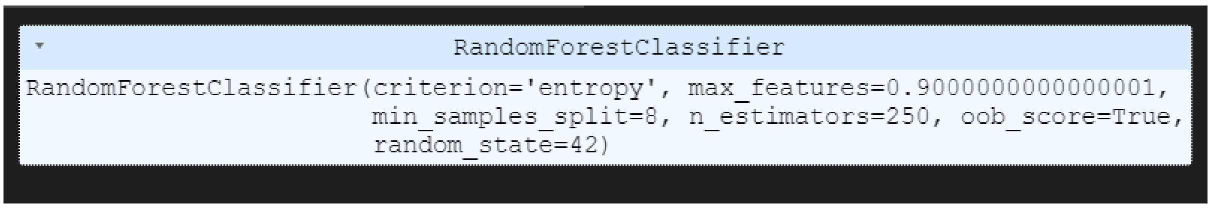


1(e) (i) Random Forests Grid Search CV



```
GridSearchCV

GridSearchCV(cv=3,
             estimator=RandomForestClassifier(criterion='entropy',
                                              oob_score=True, random_state=42),
             n_jobs=-1,
             param_grid={'max_features': array([0.1, 0.3, 0.5, 0.7, 0.9]),
                         'min_samples_split': array([2, 4, 6, 8]),
                         'n_estimators': array([ 50, 150, 250])},
             scoring=<function oob_scoring at 0x000002ADCF0151C0>, verbose=3)

estimator: RandomForestClassifier

RandomForestClassifier(criterion='entropy', oob_score=True, random_state=42)

RandomForestClassifier

RandomForestClassifier(criterion='entropy', oob_score=True, random_state=42)
```

(ii) Best Estimator



```
RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_features=0.9000000000000001,
                       min_samples_split=8, n_estimators=250, oob_score=True,
                       random_state=42)
```