Report

Group members:

Raj Shekhar Vaghela - 22CS60R32

Shaswata Dutta - 19EC39034

Sesetty Shanmukha Sai Vardhan - 19CH30018

The Decision Tree Classifier is a machine learning algorithm for supervised learning problems, especially classification tasks. The implementation of the Decision Tree Classifier in the above code uses certain functions implemented from scratch to train and test the model, make predictions, and evaluate the performance of the model. Later, its performance is compared with scikit-learn implementation of decision tree.

Implementation of Decision Trees & Interpretation

Initially, we converted the dataset suitably so that computation time decreases without hampering accuracy much. For this, we used the following functions to essentially handle both continuous and categorical feature values:

windd3_modified, winds3_modified, winds9_modified, rainfall_modified, etc (as mentioned in the code). This has been done to convert the continuous data into categorical data to save computation time, trading a little bit of accuracy.

The implementation of the **Decision Tree Classifier** consists of the following functions:

__init__: The class's constructor, which initialises the tree's root to None and sets the stopping conditions for the tree. In this case, the stopping condition is the maximum depth of the tree.

train_test_split: Split the input dataframe into training and testing dataframes (datasets) based on input argument *test_size* (which lies from 0 to 1).

check_uniqueness: Checks whether the subsampled dataset (which is now a numpy ndarray) is a leaf node (has samples of only a single label type)

classify_data: returns the label associated with the maximum number of samples within the subsampled dataset

get_possible_splits: returns a dict() of possible splitting threshold values for each feature within the subsampled dataset

split_data: splits the dataset (which is a numpy ndarray) into left and right datasets based on a splitting value

cal_entropy: calculates the entropy of a particular dataset depending upon the number of unique labels within this dataset.

cal_tot_entropy: calculates the weighted entropy obtained upon splitting the dataset at a particular splitting value.

determine_best_split: for input dataset it finds the best feature value (column) to be split on, and also finds the corresponding threshold value for splitting

decision_tree_algorithm: For input dataframe, it checks whether we are at leaf node (for which the label using classify_data()) is returned. Otherwise, cnt (variable for knowing the current depth of the tree) is updated, and recursively two trees are built depending upon the threshold to split. These trees are appended to the already available root of the tree.

classify_example: used to classify some particular example based on threshold value of some feature

cal_accuracy: calculates accuracy of the entire tree

filter_df: filters the dataframe based on continuous and categorical features/attributes

determine_leaf: determines label of the leaf node

make_predictions: predicts upon the test set based on the already learnt decision tree

determine_errors: get errors based on predicted labels which do not match with the actual test labels

pruning_result: prunes based on prediction error difference between leaf node and decision node

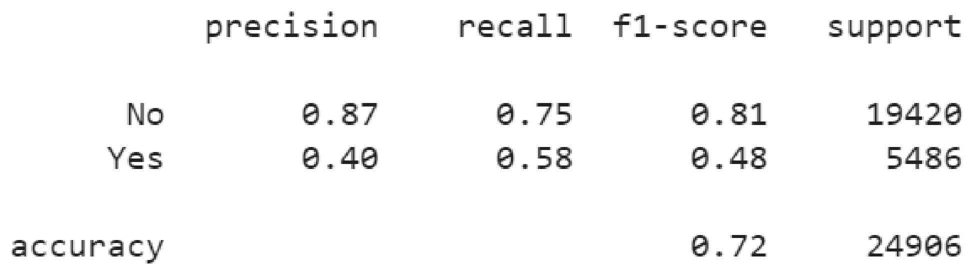
post_pruning: Performs the Reduced error pruning, where leaf nodes from the bottom of the tree are pruned one after another as long as accuracy increases.

create_plot: prints the built tree

predict_example: predicts the label of the example test sample using the already built tree

Classification Report:

The following is the classification report for the decision tree (without pruning):

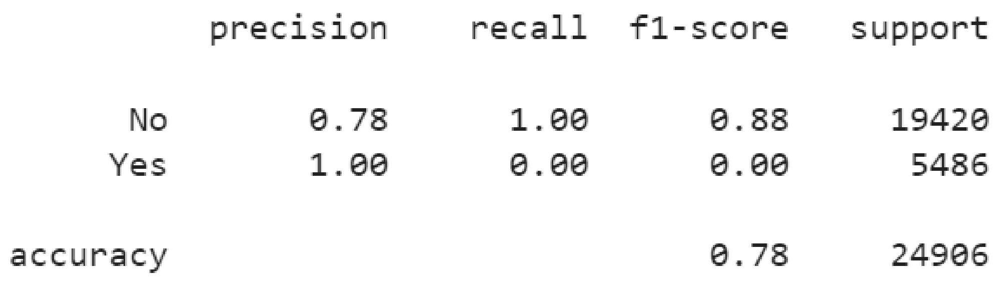


```
              precision    recall  f1-score   support

          No       0.87      0.75      0.81     19420
         Yes       0.40      0.58      0.48      5486

    accuracy                           0.72     24906
```

The following is the classification report for the decision tree with pruning:



```
              precision    recall  f1-score   support

          No       0.78      1.00      0.88     19420
         Yes       1.00      0.00      0.00      5486

    accuracy                           0.78     24906
```

(the recall for testing with the decision tree with pruning is 0 because of certain zero_division error problems encountered by sklearn).

Observations:

- Although initially we started with a node based structure for the decision tree, later we switched on to a dictionary based structure for the ease of implementation of the reduced error pruning algorithm (which is a form of post pruning).
- We had to use try and except conditions for getting the best depth of the decision tree (without pruning), so that certain key-errors could be handled properly.
- The depth-range of the tree was restricted from 3 to 13 for proper visualization of the accuracy vs depth plot. However, it must be noted that the accuracy vs depth will depend upon the random training and test datasets subsampled from the original dataset. Hence, the accuracy vs depth plot will change every time we subsample a new training set and test set from the original dataset, and then build the tree and test on the test datasets.

- The accuracy linearly increases until the depth of 9. After that, the accuracy remains almost the same for higher depths of the model. This is because the more partitions the tree makes, the more it can identify perfectly whether a given data point belongs to a particular class. Using the Best depth we found earlier, we test our model on the Test dataset and then provide the accuracy.

Results:

- Upon pruning the overall accuracy has increased from 82.8% to 84.3% (approx.).
- Upon executing multiple training epochs, it was observed that the depth of 9 is the best for training the decision tree (without pruning).
- The f1 scores of 0.81 and 0.88 are very good indicators of successful predictions by our constructed decision tree model.