L-Università ta' Malta
Faculty of Information & Communication Technology

Machine Learning Project

Karl Attard 203501(L)
B.Sc. (Hons) Artificial Intelligence

Study-unit: **Machine Learning, Expert Systems and Fuzzy Logic**
Unit Code: **ICS3206**
Lecturer: **Dr Kristian Guillaumier**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

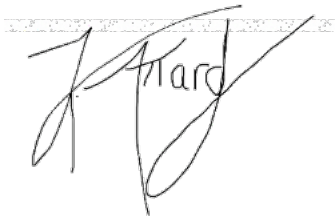
Plagiarism is defined as “the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I, the undersigned, declare that the assignment submitted is our work, except where acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected and will be given zero marks.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).



Karl Attard

Student Name Signature

ICS3206 **Machine Learning Project**

Course Code Title of work submitted

10/01/2022

Date

Table of Contents

Introduction

Machine Learning (ML) is pivotal in the advancements of Artificial Intelligence (AI). This phenomenon is a subset of AI, and it is a way of how computer algorithms learn patterns automatically from the given data.

This field of study has many types of techniques of how to achieve the aforementioned, for instance, supervised and unsupervised learning algorithms. In this research paper, we will only be focusing on a particular supervised learning algorithm, the Decision Tree Classifier. First of all, supervised learning (SL) is the task of ML where the trained data is labelled and thus the algorithm can map an input to a label. Supervised techniques can either be a classification task or a regression task. In this research, we will only be focusing on the classification aspect where this SL algorithm will perform binary classification as well as multi-class classification.

The decision tree classifier is one of the most popular algorithms within the SL family. This is because it requires less effort for data preparation than other algorithms, intuitive, simple to understand and much more. This algorithm works by creating a model on its training data and predicts unseen data by inferring decision rules from this training data. One of the most basic algorithms used to train this model is the ID3 algorithm. This algorithm is used to implement the scope of this project.

In this research study, classification datasets obtained from [1] will be used to train a decision tree classifier using the ID3 algorithm from scratch. Datasets which are both binary classification as well as multi-class classification having categorical and continuous variables are chosen as per requirements. Each dataset is split into Training set and Test set with the former being. Decision tree models will be trained on each of the datasets chosen and are then evaluated on the test set. For evaluation purposes, it was decided to output a classification report for each model containing the precision, recall, f-1 score and accuracy. This will be further highlighted in the sections below.

Discussion

This research study is all about ID3 decision tree learning. As previously mentioned, the ID3 algorithm is a very basic and simple algorithm which is used to train and build a decision tree classifier. This section will explain what a decision tree classifier is, how does the ID3 algorithm work as well as the methodology carried out.

1. Decision Tree Classifier

A decision tree is a supervised machine learning algorithm. As the name implies, a decision tree is a tree-like model starting from the root node up to the leaf node. Each internal node represents the decision on the attribute whilst the leaf node stores the class label/outcome. A path is defined by going through the respective nodes based on the decisions taken and thus, it represents all the decisions taken to classify a class label of that data (see figure 1 below).

A decision tree can take the form of a classification problem or a regression problem. The former involves the task of predicting discrete values whilst the latter involves the task of predicting continuous values. For the scope of this project, we will only be dealing with classification instances, both binary classification and multi-class classification.

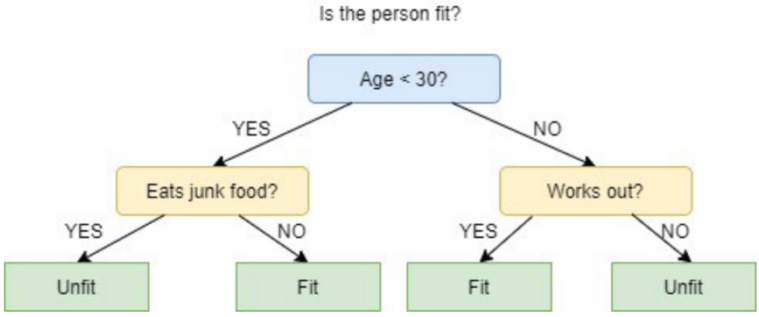


Figure 1: Decision Tree Classifier Example obtained from [2].

2. ID3 Algorithm

The ID3 algorithm, which stands for Iterative Dichotomiser 3 is an algorithm which builds and trains a decision tree in a top-down greedy approach [2] and follows the Occam's razor principle by attempting to build the simplest possible tree [3]. This former means that this algorithm builds the decision tree starting from the root node and at every iteration, it selects the best feature to create a node. To find the best feature, the ID3 algorithm uses the Information Gain (IG) metric. Essentially, this metric computes the difference of each attribute's entropy from the whole dataset's entropy. The feature with the highest information gain is chosen to be the best one, hence, the next node to be created in the tree. As a result, information gain is basically measuring how well that attribute separates the training set according to the target classification. On the other hand, entropy is simply a measurement of 'disorder' [2], or rather 'unpredictability' in a

variable expected outcome. High entropy denotes that it is impossible to predict what will come next whilst a low entropy denotes that the expected outcome is guaranteed in any scenario.

The following are the equations to calculate Entropy and Information Gain respectively obtained from [2]:

$$\mathbf{Entropy(S) = - \sum p_i * \log_2(p_i) \; ; \; i = 1 \text{ to } n}$$

Where n is the total number of classifications possible.
p_i is the probability of class 'i'

$$\mathbf{IG(S, A) = Entropy(S) - \sum((|S_v| / |S|) * Entropy(S_v))}$$

Where S_v is the set of rows in S such that feature A has a value v.
$|S_v|$ and $|S|$ denotes the number of rows respectively.

Figure 2 below highlights how the decision tree is built using ID3 in a graphical way:

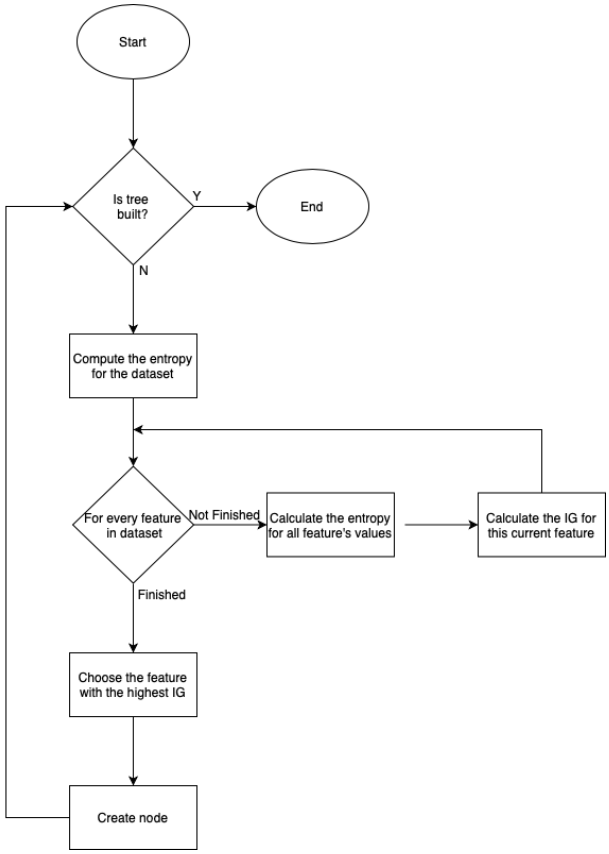


Figure 2: Building the tree using ID3 algorithm flowchart representation

As explained in this section, the ID3 algorithm is being used to build a decision tree from the training data. This tree is then used to classify unseen data by using the paths within the tree. This search strategy algorithm is efficient in building a tree and prediction rules are simple to understand, however, data can be overfitted. Therefore, an overfitting method will also be implemented and explained later in this report.

3. Alternative Approach to this Task

As for every task, one can use a different approach to solve the required task. Apart from using ID3, one can use C4.5 algorithm. C4.5 is the successor and extension to ID3 which was created by the same person who invented the ID3 algorithm [4]. The motivation behind implementing a new algorithm is that the major drawback of ID3 is that it does handle features which are continuous. Although this can be supported by converting them to categorical, as done in our project, it is still nonetheless a drawback to have this restriction. The C4.5 achieves this by dynamically defining a discrete attribute which partitions the continuous feature value into a discrete set of intervals. The definition of discrete attribute is based on numerical variables [5].

Moreover, another disadvantage of ID3 is that this algorithm can overfit the training data. Hence, the C4.5 algorithm utilises the Single Pass Pruning Process [6] to reduce overfitting. This is accomplished by removing a rule's precondition if and only if the accuracy of that rule improves without it [5]. Apart from this, the C4.5 algorithm also handles attribute having 'unknown' values better than ID3 [4] and converts trained decision trees into sets of if-then rules [5]. Additionally, the accuracy for each rule is then evaluated which will then be used to determine their order of how which they will be applied.

In terms of whether C4.5 will work better for this project task; I believe that yes this will be case. This is because C4.5 can handle 'unknown' values better than ID3. In the datasets chosen, both of them contains some empty-valued attributes and thus the C4.5 will perform better when having incomplete data. Moreover, datasets can contain categorical and continuous valued attribute, and therefore, this brings me to another reason why C4.5 would be a better option than ID3 in our case. Another advantage would be that C4.5 will perform pruning as its overfitting method after the decision being created to generalise better on unseen data. Finally, in terms of performance, the C4.5 performs better in accuracy as well as execution time, as highlighted by Hssina et al. [4].

Methodology

In this section, the methodology will be explained for which a decision tree using ID3 algorithm has been built from scratch, using Python as the programming language. This section will begin by explaining the chosen datasets [1]; how data is being read and pre-processed; how the decision tree is built [7]; and the overfitting method implemented.

1. Datasets

As per requirements of this project, two or more classification datasets had to be chosen from [1] where at least one dataset contains multi-classes in its target column as well as features having continuous variables too.

Hence, to satisfy these requirements, the 'Car Evaluation' dataset [8] as well as the 'Wine' dataset [9] were chosen which are both CSV files when downloaded. These will be explained below.

I. Car Evaluation Dataset

This dataset has a total of 6 attributes which are all categorical and its target column is multi-class (4 classes), having a total of 1728 rows. This multivariate dataset purpose it to evaluate cars based on their features.

II. Wine Dataset

This multivariate dataset has a total of 13 attributes which are all continuous. Its target column is a multi-class classification having 3 different classes possibilities and has a total number of 178 rows.

It is important to note that both datasets chosen contains missing values and thus have incomplete data. To read the dataset, the 'read_csv()' Pandas library function was used. The above datasets were chosen because it was decided that these fully satisfy the requirements of the project since the former contains categorical features with multi-class labels whilst the latter contains various continuous-valued features with multiclass labels.

2. Implementation

This project was implemented with Python3 using PyCharm as the IDE. It contains 3 python scripts called 'main.py', 'preprocessing.py' and 'decision_tree_id3.py'.

The first script uses all methods in the other 2 scripts to read the CSV datasets, pre-process them and builds the tree using ID3 algorithm. The second script reads and pre-processes the datasets where essentially methods are created to split the dataset into a training set and test set and converts any continuous features. The third script contains all the

methods which are used to build the tree including functions to calculate the required entropies as well as a recursive method to build the decision tree.

This section will not describe the all the functions implemented but will solely focus on how attribute nodes were selected, how continuous valued features are supported as well as the overfitting method.

I. How Best Attributes are Chosen

The developed and implemented ID3 algorithm calculates the information gain for every feature in the training data. Information gain (IG) is a statistical property that measures how well a given feature separates the training instances depending on their respective target classification. This statistical component uses a measure called entropy which essentially computes the unpredictability in an attribute's outcome. The equation of Entropy and IG were previously listed above. Essentially, information gain of an attribute is calculated by subtracting the entropy of that respective attribute from the entropy of the dataset (see equation above). In the developed implementation, all the information gains for each attribute are stored in a list and the algorithm then selects that attribute having the highest information gain. This is because our goal is to maximise IG to generate the best split in the dataset and thus, this is the greedy-approach aspect of the ID3 algorithm. Subsequently, a new node based on the chosen IG is created and added to the tree and this process is repeated until the whole desired tree is built, as shown in figure 2 above.

II. Support Continuous-Valued Features

This implementation also supports continuous valued attributes. To achieve this, a function called 'convertContinuousToCategorical()' is implemented which converts all continuous valued features to categorical, each containing 3 categories (outlined below). This function takes in the dataset and its continuous features. If no continuous features are present, then the algorithm terminates, meaning there are no conversions to make. Otherwise, it iterates over all these features (except target column obviously), and for each feature, it computes their respective First and Third Quartile metrics. These computed metrics are considered as the thresholds of that feature for which the conversion from continouous to categorical will be made. Hence, all continuous values are converted to their respective categorical variable depending on where they lie. For instance, suppose Q1 metric is 5, then all the feature's values less than this Q1 are converted to '<5'. The same logic is applied to other values, for example, if Q3 is 10, then all values of that feature which lie between 5 and 10 are converted to '<10' whilst all values greater than Q3, i.e., 10, then they are converted to '>=10. The same thing is done to all other remaining continuous features.

III. Overfitting Method

In this project, it was also required to develop and implement an overfitting strategy to the tree. Overfitting happens when there is noise or error in our training data, or else if the number of training instances is insufficient to represent out true target, then the tree will overfit these training examples. The term overfitting implies that there exists some other tree which fits the training data worse but performs better on unseen data. For this project, the 'Reduced Error Pruning' method was used. A function called 'postPruning' was implemented and is called after a decision tree is created and trained. This function takes in the trained tree, the training and test set. Remember that our decision tree is of type 'Dict' and thus the keys represent the feature name whilst its value is either a 'Dict' (sub-tree) or a value (leaf node classification), representing the 'test' on that key. Therefore, this algorithm iterates top to bottom and checks if the child of the current key's value is a leaf node or a sub-tree. A list of Boolean values is stored to denote whether pruning should take place or not in the respective child of key's value, i.e., if it contains a sub-tree or not. Then, the implementation moves on for the recursion part. The base case for recursion is that there are no sub-trees to prune whilst the recursive part will recursively recall this function, each time checking if the child of the sub-tree key's value is a leaf node or not. If it is a leaf node, then we store this leaf node. Otherwise, the recursive call is made passing in the respective child's sub-tree of the key's value. In the base case, a function called 'prune()' is being called which takes in a tree, the training and test set. This function is responsible of performing pruning on the passed sub-tree and is achieved by changing the sub-tree to a leaf node if and only if the resulting error in the pruned tree is better than the error of the current tree as it is. This essentially means that the sub-tree is pruned if the pruned sub-tree has higher accuracy on the test set. If the error of performing pruning is better, then change the sub-tree to the leaf node where the leaf node is the most occurred class label over the training instances.

Essentially, what this algorithm is doing is that it is iterating over every path in the decision tree, and if a sub-tree is present, then prune it and if the resulting pruned tree has better error than the current trained tree, then replace the sub-tree with the leaf node having the most frequent class label. Therefore, the resulting pruned tree is smaller in size and has higher accuracy over unseen data.

Evaluation

In this section, the decision trees built for each dataset will be evaluated accordingly. To evaluate models, a classification report is outputted highlighting the Precision, Recall, F-1 score and accuracy for each trained tree. These will then be compared to the evaluation shown in [8] and [9] respectively. Finally, these results will be discussed accordingly.

Firstly, it is important to understand that decision trees are built and evaluated in the following manner:

1. Read the whole dataset from the CSV file.
2. Pre-process it to support continuous-valued attributes.
3. Split it into a Training set and a Test set (80-20 respectively).
4. Train the decision tree classifier on the train data set.
5. Perform Pruning to avoid overfitting.
6. Run the model built on the test date, i.e., unseen data and evaluate it.

Now, several experiments will be carried out to guarantee a fair evaluation with the models built. It is important to note that prior to splitting the dataset, it is randomly shuffled to prevent any bias. All trained models built will perform classification on the test set generated previously and subsequently, a classification report is outputted to evaluate it accordingly.

These experiments will be conducted by re-running the code implemented, each time the training and test set changes in terms of content not size. It is important to note that 2 tables for each dataset will be shown to evaluate trees before and after performing pruning.

When running the python scripts, the following tabulated outputs are obtained:

1. First Experiment

Target Class Labels				
	acc	good	unacc	vgood
Precision	90.67	70.0	100.00	83.33
Recall	77.27	70.0	97.08	62.50
F1-Score	83.44	70.0	98.52	71.43
Overall Accuracy is *90.46%* before pruning the tree.				

Table 1: Decision Tree Classifier on Car Dataset before pruning.

Target Class Labels				
	acc	good	unacc	vgood
Precision	91.76	70.0	100.00	85.71
Recall	88.64	70.0	97.08	75.00
F1-Score	90.17	70.0	98.52	80.00
Overall Accuracy is *93.64%* after pruning the tree.				

Table 2: Decision Tree Classifier on Car Dataset after pruning.

Table 1 and 2 above shows the obtained classification report of the tree trained on the car dataset before and after performing pruning respectively. It can immediately be notice that performing pruning on the trained tree yielded a better accuracy on unseen data. This means that the pruned tree generalises better the unseen data. However, it is worth mentioning that the tree prior to performing pruning already had a high overall accuracy, i.e., 90.46% but the pruned tree significantly increased this accuracy to 93.64%. Moreover, in the 'Evaluation' section of [8] where accuracies for different models on this dataset are shown, it can be noted that the obtained accuracy is similar to the accuracies of these models. For instance, the 'Random Forest Classifier' in [8] obtained an accuracy between 90% and 95%. Both our accuracies lies within this range, and thus, this means that our ID3 algorithm and pruning method are working correctly and achieving good results.

Target Class Labels			
	1	2	3
Precision	100.00	82.35	100.00
Recall	91.67	100.00	80.00
F1-Score	95.65	90.32	88.89
Overall Accuracy is *91.67%* before pruning the tree.			

Table 3: Decision Tree Classifier on Wine Dataset before pruning.

Target Class Labels			
	1	2	3
Precision	100.00	82.35	100.00
Recall	91.67	100.00	80.00
F1-Score	95.65	90.32	88.89
Overall Accuracy is *91.67%* after pruning the tree.			

Table 4: Decision Tree Classifier on Wine Dataset after pruning.

Similarly, table 3 and 4 above highlight the evaluation report of the tree, but this time trained on the wine dataset. As can be seen above, the accuracy before and after performing pruning remained unchanged. Although pruning did not increase the accuracy, it does not mean that the tree has not been pruned. This is because the implemented overfitting strategy stills prunes the tree when the error of the pruned tree on the test data is the same or less than the error of the original tree. Hence, the trained tree is still pruned but accuracy did not change. However, the accuracies obtained are still relatively high and when compared to the 'Evaluation' section of [9], these metrics are similar to ours. For instance, the 'Logistic Regression' model in [9] obtained an accuracy between 87% and 100% and thus, our obtained accuracy of 91.67% lies in this range, meaning that the implemented algorithms are working as desired.

2. Second Experiment

Target Class Labels				
	acc	good	unacc	vgood
Precision	91.38	85.71	98.39	57.14
Recall	66.25	54.55	97.99	66.67
F1-Score	76.81	66.67	98.19	61.54
Overall Accuracy is *88.73%* before pruning the tree.				

Table 5: Decision Tree Classifier on Car Dataset before pruning.

Target Class Labels				
	acc	good	unacc	vgood
Precision	92.86	90.00	99.19	62.50
Recall	81.25	81.82	97.99	83.33
F1-Score	86.67	85.71	98.59	71.43
Overall Accuracy is *93.35%* after pruning the tree.				

Table 6: Decision Tree Classifier on Car Dataset after pruning.

Similar results were obtained when the code implemented was rerun for the second time. As can be observed above, the overall accuracy of the tree before performing pruning is less than the respective accuracy in table 1 above. This is because training and test set are split differently each time the code is executed due to randomly reshuffling the whole dataset. This could be a factor why the initial accuracy dropped slightly, however, after performing pruning, the trained tree's accuracy on the car dataset increased dramatically. The overall accuracy after performing pruning reach the same (almost) as figure 2 above.

Target Class Labels			
	1	2	3
Precision	88.89	91.67	80.00
Recall	100.00	73.33	92.31
F1-Score	94.12	81.48	85.71
Overall Accuracy is *86.11%* before pruning the tree.			

Table 7: Decision Tree Classifier on Wine Dataset before pruning.

Target Class Labels			
	1	2	3
Precision	88.89	92.31	85.71
Recall	100.00	80.00	92.31
F1-Score	94.12	85.71	88.89
Overall Accuracy is *88.89%* after pruning the tree.			

Table 8: Decision Tree Classifier on Wine Dataset after pruning.

From the results obtained above, it can be deduced that this time performing pruning on the trained tree has increased accuracy, when compared to the first experiment. However, in this experiment, although the overall accuracy increased after performing pruning, it was still nonetheless less than tables 3 and 4 respectively. Moreover, the accuracy of the pruned tree on the wine dataset lies within the 'Evaluation' section of [9], as previously mentioned. This accuracy is considered quite well and thus is generalising with high accuracy on unseen data.

3. Third Experiment

Target Class Labels				
	acc	good	unacc	vgood
Precision	85.45	72.73	100.00	100.00
Recall	78.33	57.14	95.40	72.73
F1-Score	81.74	64.00	97.65	84.21
Overall Accuracy is *90.17%* before pruning the tree.				

Table 9: Decision Tree Classifier on Car Dataset before pruning.

Target Class Labels				
	acc	good	unacc	vgood
Precision	86.89	78.57	100.00	100.00
Recall	88.33	78.57	95.40	90.91
F1-Score	87.60	78.57	97.65	95.24
Overall Accuracy is *93.35%* after pruning the tree.				

Table 10: Decision Tree Classifier on Car Dataset after pruning.

Target Class Labels			
	1	2	3
Precision	91.67	93.33	88.89
Recall	91.67	87.50	100.00
F1-Score	91.67	90.32	94.12
Overall Accuracy is *91.67%* after pruning the tree.			

Table 11: Decision Tree Classifier on Wine Dataset before pruning.

Target Class Labels			
	1	2	3
Precision	100.00	93.75	88.89
Recall	91.67	93.75	100.00
F1-Score	95.65	93.75	94.12
Overall Accuracy is *94.44%* after pruning the tree.			

Table 12: Decision Tree Classifier on Wine Dataset after pruning.

Tables 9 and 10 follow the same reasoning as tables 1 and 2 above respectively whilst tables 11 and 12 also follow the same logic as tables 7 and 8 above. This is because in tables 9 and 10, the overall accuracy has increased after performing pruning and reached similar accuracy

score as tables 1 and 2 respectively. On the other hand, in tables 11 and 12, it can be noted that this time the accuracy before and after pruning were both better than tables 7 and 8 respectively.

4. Forth Experiment

Target Class Labels				
	acc	good	unacc	vgood
Precision	91.67	85.71	98.39	75.00
Recall	75.34	46.15	97.60	60.00
F1-Score	82.71	60.00	97.99	66.67
Overall Accuracy is *89.88%* before pruning the tree.				

Table 13: Decision Tree Classifier on Car Dataset before pruning.

Target Class Labels				
	acc	good	unacc	vgood
Precision	92.86	88.89	100.00	80.0
Recall	89.04	61.54	97.60	80.0
F1-Score	90.91	72.73	98.79	80.0
Overall Accuracy is *93.93%* after pruning the tree.				

Table 14: Decision Tree Classifier on Car Dataset after pruning.

Target Class Labels			
	1	2	3
Precision	100.00	87.5	100.00
Recall	100.00	100.00	84.62
F1-Score	100.00	93.33	91.67
Overall Accuracy is *94.44%* before and after pruning the tree.			

Table 15: Decision Tree Classifier on Wine Dataset before and after pruning.

In this experiment, the accuracy obtained from the trained tree on the car dataset again follows the same logic as above. The tree is achieving a high accuracy even without pruning, however, when performing pruning method, the accuracy increases, leading to a more generative tree on unseen instances. With regards to the trained tree on the wine dataset, the classification report remained unchanged before and after pruning. This also happened in experiment 1 above but this time, the overall accuracy is higher, i.e., 94.44% when compared to 91.67% in tables 3 and 4 above. This means that this tree performs better on unseen data than the one in experiment 1 above.

5. Fifth Experiment

Target Class Labels				
	Acc	good	unacc	vgood
Precision	84.75	62.50	100.00	88.89
Recall	73.53	45.45	94.96	88.89
F1-Score	78.74	52.63	97.42	88.89
Overall Accuracy is *89.02%* before pruning the tree.				

Table 16: Decision Tree Classifier on Car Dataset before pruning.

Target Class Labels				
	Acc	good	unacc	vgood
Precision	87.32	72.73	100.00	90.00
Recall	91.18	72.73	94.96	100.00
F1-Score	89.21	72.73	97.42	94.74
Overall Accuracy is *93.64%* after pruning the tree.				

Table 17: Decision Tree Classifier on Car Dataset after pruning.

Target Class Labels			
	1	2	3
Precision	100.00	100.00	92.31
Recall	100.00	92.86	100.00
F1-Score	100.00	96.30	96.00
Overall Accuracy is *97.22%* before and after pruning the tree.			

Table 18: Decision Tree Classifier on Wine Dataset before and after pruning.

The final experiment conducted performed comparably with the above. As can be observed, and as already mentioned above, the pruned tree on the car dataset is performing better on unseen data, having an accuracy of 93.64%, an increase of 4.62% when compared to the tree generated before performing pruning. This is a great improvement since the tree is generalising more accurate on unseen data. On the other hand, the results tabulated in table 18 above highlights the fact that accuracy remained the same before and after performing pruning. This does not mean that the tree was not pruned but it means that when pruning was done, the pruned tree and original tree had the same error, but it is still better to prune it nonetheless for a shallower tree. The accuracy obtained in this experiment is the highest overall all experiment conducted with respect to the tree trained on the wine dataset. This overall accuracy stood at 97.22% which is considered great and thus, is generalising very well on unseen instances.

NB: All the above evaluation metrics are outputted in the program, and it is important to note that more experiments were conducted, but not recorded. However, the same conclusions could be inferred from these other experiments conducted visually (just by seeing output of the program).

Conclusion

This project presented a detailed and comparative study on the ID3 algorithm as well as decision trees. The goal of this project was to implement an ID3 algorithm from scratch to build a decision tree classifier using any given datasets [1]. The implementation also had to deal with continuous-valued features which the classification datasets could contain. Moreover, an overfitting strategy method had to be implemented to prevent overfitting on the trained trees. The 'Reduced-Error' Pruning method was chosen for this project. Apart from the implementation, it was vital to evaluate each tree built. A classification report on each tree built on the test data was outputted to the user before and after performing pruning on that tree. To make this evaluation fair, multiple experiments were carried out, each time the dataset was split differently since it was shuffled randomly (different in content and not size). Therefore, different accuracies and evaluation metrics were obtained each time and observations were then conducted.

From the observations carried out, it could be deduced that in general, the trees built on each dataset all had high accuracies, ranging from approximately 88% to 97% (depending on dataset). Moreover, after performing pruning pn the trees, this accuracy metric continued to increase, making the tree more generative and therefore, performing better on unseen data. However, accuracy before and after pruning did not always change for the wine dataset, however, pruning was still done on the tree, making it shallower. This was the case because when performing pruning, the pruned tree had identical errors as the original tree, but pruning was still performed.

In addition, a further comparative analysis was conducted when the results obtained from the experiments were compared to the 'Evaluation' section in [8] or [9] respectively, depending on the dataset. From the analysis, it was concluded that the accuracies obtained from the experiments carried out were similar to the aforementioned section. Therefore, the implementation developed was working as desired.

In conclusion, the ID3 algorithm is a very simple and effective algorithm used to build decision trees. In this research study, an alternative algorithm to ID3 was also mentioned and speculated on this project task purpose.

Statement of Completion

Item	Completed (Yes/No/Partial)
Data Selection and Import	Yes
ID3	Yes
Support Continuous Attributes	Yes
Overfitting Management	Yes
Good discussion on an alternative method	Yes
If partial, explain what has been done	

References

[1] U. Irvine, “UC Irvine Machine Learning Repository,” 1987. [Online]. Available: https://archive-beta.ics.uci.edu/ml/datasets. [Accessed 11 December 2021].

[2] Y. Sakkaf, “Decision Trees: ID3 Algorithm Explained,” 31 March 2020. [Online]. Available: https://towardsdatascience.com/decision-trees-for-classification-id3-algorithm-explained-89df76e72df1. [Accessed 11 December 2021].

[3] A. Rizvi, “ID3 Algorithm,” 2010. [Online]. Available: http://athena.ecs.csus.edu/~mei/177/ID3_Algorithm.pdf. [Accessed 11 December 2021].

[4] A. M. E. E. E. E. Badr Hssina, “A comparative study of decision tree ID3 and C4.5,” *International Journal of Advanced Computer Science and Applications Special Issue on Advances in Vehicular Ad Hoc Networking and Applications,* 2014.

[5] “Tree algorithms: ID3, C4.5, C5.0 and CART,” DataDrivenInvestor, 20 February 2019. [Online]. Available: https://medium.datadriveninvestor.com/tree-algorithms-id3-c4-5-c5-0-and-cart-413387342164. [Accessed 11 December 2021].

[6] S. Saha, “What is the C4.5 algorithm and how does it work?,” 20 August 2018. [Online]. Available: https://towardsdatascience.com/what-is-the-c4-5-algorithm-and-how-does-it-work-2b971a9e7db0. [Accessed 11 December 2021].

[7] T. R. Joy, “Decision Tree: ID3 Algorithm,” [Online]. Available: https://medium.com/geekculture/step-by-step-decision-tree-id3-algorithm-from-scratch-in-python-no-fancy-library-4822bbfdd88f. [Accessed 11 December 2021].

[8] “Car Evaluation Dataset,” UCI Machine Learning Repository, 1997. [Online]. Available: https://archive-beta.ics.uci.edu/ml/datasets/car+evaluation. [Accessed 11 December 2021].

[9] “Wine Dataset,” UCI Machine Learning Repository, 1991. [Online]. Available: https://archive-beta.ics.uci.edu/ml/datasets/wine. [Accessed 11 December 2021].