Mikias Berhanu | 2021280115
Assignment V
Supervised and Unsupervised Learning

Lab Experiments with Decision Trees

Decision trees are one of the algorithms used for supervised learning in which they are used for both classification and regression tasks based on previous labeled data. When dealing with decision trees there are few key points to consider. **Root Node:** it's the main root or the base of every decision tree and a decision tree will have one root node. **Splitting:** this is the idea of splitting nodes into several sub-nodes, creating parent child relationships. **Decision Node:** when sub-nodes which are created from prior nodes are splitted into other sub-nodes. **Leaf Node:** these are nodes which can't be further splitted into sub-nodes and usually are the last nodes in the tree architecture. **Pruning:** this is the process of removing sub-nodes from the tree, which don't have much significance in the decision making process. **Branch:** part of the tree which contains a number of sub-nodes together.

This experiment is taken using python programming language and a popular machine learning library called scikit-learn. The first dataset used is the Iris dataset which contains information about flower petal length and width and also sepal length and width. Based on these given information which are numeric we want to classify the flowers as **setosa, versicolor, virginica**. The dataset has 150 entries where each class is equally represented, meaning each class has an equal number of data entries 50 for setosa, 50 for versicolor and 50 for virginica. The data types for the other features are given as a float value which is good for machine learning, there are no categorical values so no further processing is needed on the dataset. The data comes in the form of numeric series so it's a good idea to turn it into pandas Dataframe.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

The first 5 entries of the iris dataset

Once the data is loaded we can do all sorts of operations. The mean for the sepal length is 5.843333, for sepal width 3.057333, for petal length 3.758000, for petal width 1.199333 on the other hand the standard deviation for sepal length is 0.828066, for sepal width 0.435866, for petal length 1.765298, for petal width 0.762238. The **describe** function of pandas library gives more information about the numeric structure of the dataset.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

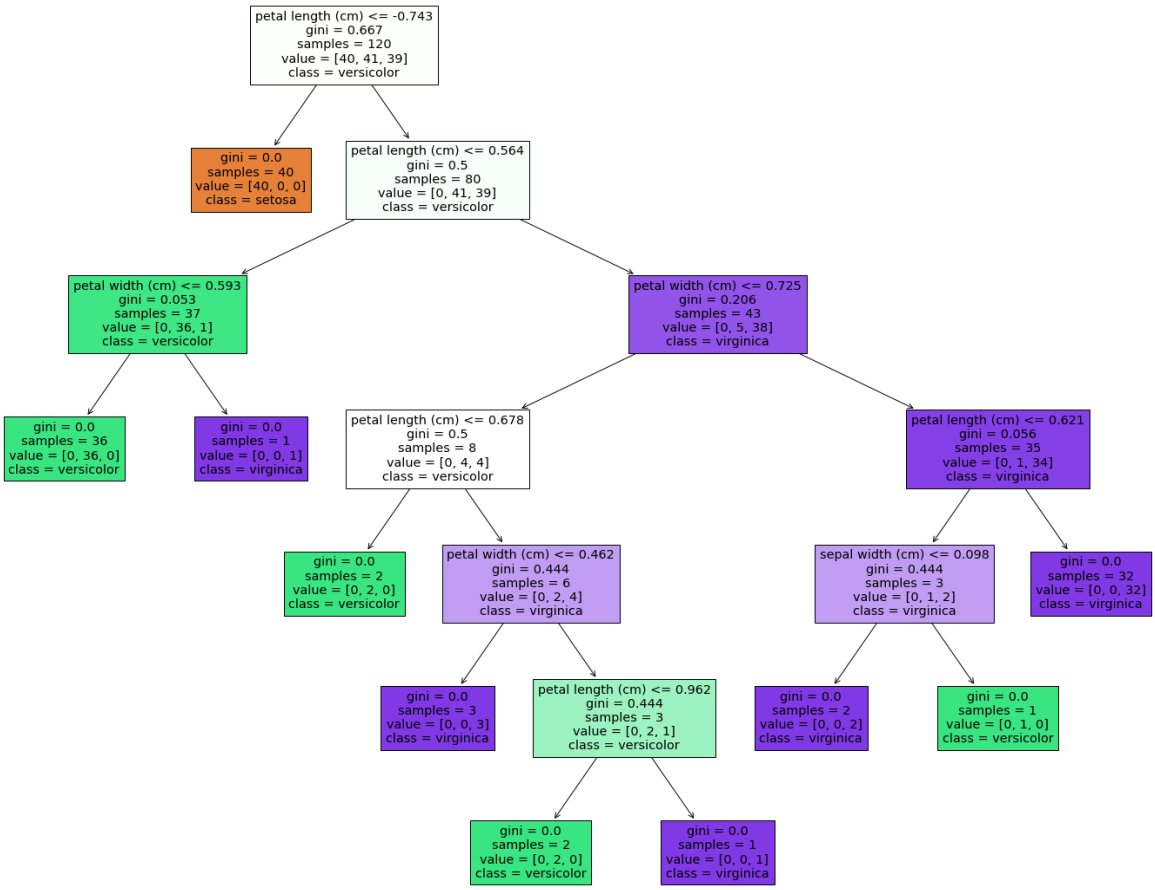
Numeric Information of the Iris dataset

The next thing to do is to get the dependent and independent variables, meaning the X and Y values for training our model in this case Decision Trees. The independent variables are speal length, sepal width, petal length and petal width which are the features or predictors. The dependent variable is going to be a class which is numerically labeled from 0 - 2, 0 for setosa, 1 for versicolor and 2 for

virginica. Once we extract the X and Y values the next thing to do is to scale the values of X into a specific range in this case in between 0 and 1, this is because the X values have different scale of numbers and this will make it hard for the model to generalize properly. Scaling is a data preprocessing stage and it will improve the performance of our model drastically.

Once the X values are scaled to a specific range, then we split the data into training and testing sets, the ratio of the split is 80:20 meaning 80% of the data will be used for training and the rest 20% will be used for testing the model. The data will be shuffled randomly as well this will help the model not to overfit and learn different patterns. Once the data is properly splitted we can then initialize our DecisionTreeClassifier model which is part of the scikit learn library for training and testing.

Once the model is trained and tested we can check the performance of the model using different metrics, the most common performance metrics for classification are **confusion matrix**, **F1-score**, **recall** and **precision.** The Decision Tree model for this specific dataset has an F1-score of 1, recall score of 1 and precision score of 1, which means 100% for all three of them and can perfectly predict new incoming inputs.



Decision Tree Architecture for Iris dataset

The second experiment is using a different dataset, this dataset has 2126 entries with 22 columns, compared to the previous dataset this one is a bit larger in size and has many features. Each entry is part of a class either 1, 2 or 3. The dataset is not balanced like the previous one. 1655 entries are part of class 1, 295 are part of class 2 and 176 are part of class 3. When we split the dataset we have to make sure that each class is represented well both in the training and testing set otherwise it

will create baise in the results. Due to this reason rather than a plain 80:20 ratio split, I used stratified sampling where the data is splitted into 5 folds and each class will be represented well and helps the performance of the model as well.

	LB	AC	FM	UC	DL	DS	DP	ASTV	MSTV	ALTV	...	Min	Max	Nmax	Nzeros	Mode	Mean	Median	Variance	Tendency	NSP
0	120	0.000000	0.0	0.000000	0.000000	0.0	0.0	73	0.5	43	...	62	126	2	0	120	137	121	73	1	2
1	132	0.006380	0.0	0.006380	0.003190	0.0	0.0	17	2.1	0	...	68	198	6	1	141	136	140	12	0	1
2	133	0.003322	0.0	0.008306	0.003322	0.0	0.0	16	2.1	0	...	68	198	5	1	141	135	138	13	0	1
3	134	0.002561	0.0	0.007682	0.002561	0.0	0.0	16	2.4	0	...	53	170	11	0	137	134	137	13	1	1
4	132	0.006515	0.0	0.008143	0.000000	0.0	0.0	16	2.4	0	...	53	170	9	0	137	136	138	11	1	1

5 rows × 22 columns

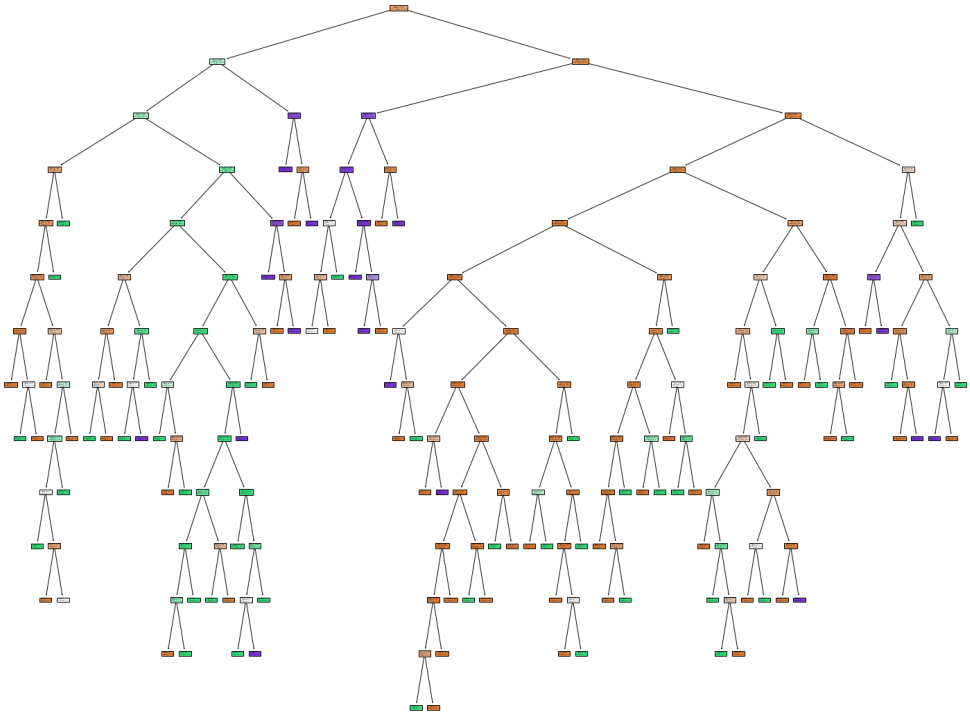
The first 5 entries of the dataset

	LB	AC	FM	UC	DL	DS	DP	ASTV	MSTV	ALTV	...	Min	Max
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.00000	...	2126.000000	2126.000000
mean	133.303857	0.003170	0.009474	0.004357	0.001885	0.000004	0.000157	46.990122	1.332785	9.84666	...	93.579492	164.025400
std	9.840844	0.003860	0.046670	0.002940	0.002962	0.000063	0.000580	17.192814	0.883241	18.39688	...	29.560212	17.944183
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	12.000000	0.200000	0.00000	...	50.000000	122.000000
25%	126.000000	0.000000	0.000000	0.001876	0.000000	0.000000	0.000000	32.000000	0.700000	0.00000	...	67.000000	152.000000
50%	133.000000	0.001630	0.000000	0.004482	0.000000	0.000000	0.000000	49.000000	1.200000	0.00000	...	93.000000	162.000000
75%	140.000000	0.005631	0.002512	0.006525	0.003264	0.000000	0.000000	61.000000	1.700000	11.00000	...	120.000000	174.000000
max	160.000000	0.019284	0.480634	0.014925	0.015385	0.001353	0.005348	87.000000	7.000000	91.00000	...	159.000000	238.000000

8 rows × 22 columns

Numerical description of the data

Once the data is loaded properly we can go ahead and extract the X and y values just like before. Scale the values back to 0 - 1 since the values have different ranges. The split the dataset using stratified sampling. Once the dataset is splitted properly we can start training our model. The performance of the model can be checked using previously mentioned metrics. This model has a f1-score of 91%, recall score 91% and precision score 91%.



Tree structure for second experiment

Summary

Decision Trees are one of the most widely used supervised learning models for both regression and classification tasks. Decision Trees are powerful and somehow imitate human thought process. In this experiment we have performed classification tasks on two different datasets with different content and size. The performance of the classification can be examined using confusion matrix, precision, recall and f1-score. When our data is not sampled we have to represent all the classes properly to avoid baise in our prediction. The source code of this project is linked here