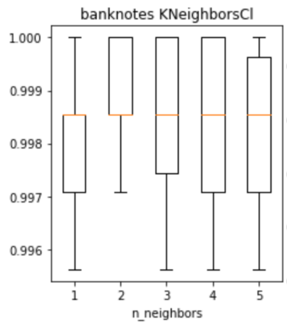
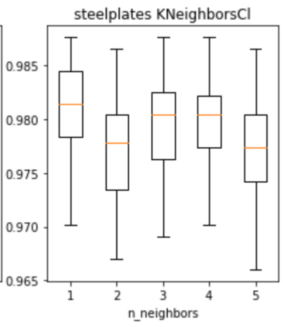
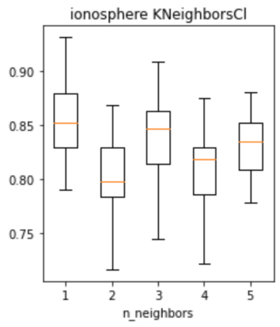
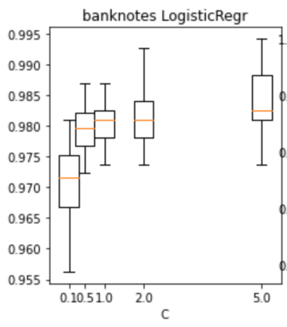
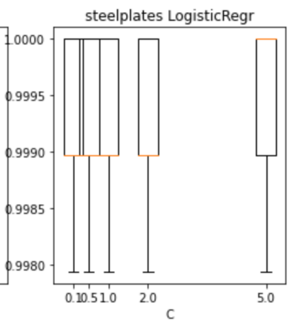
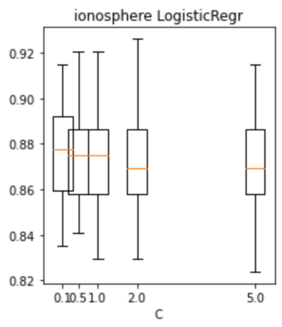
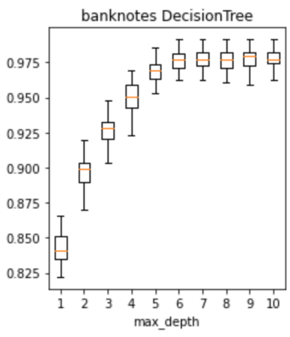
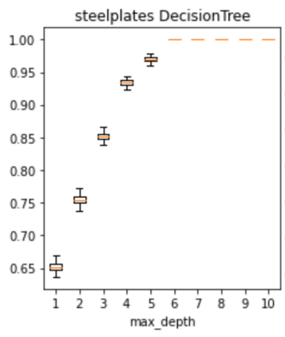
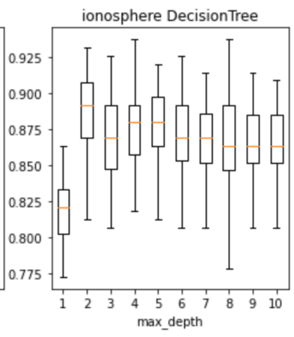
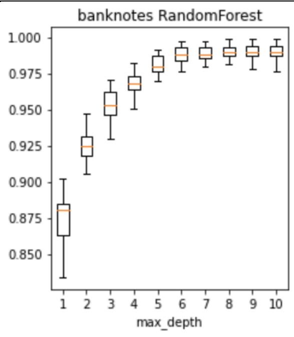
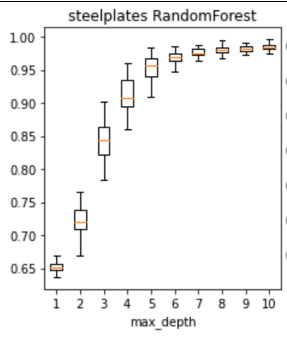
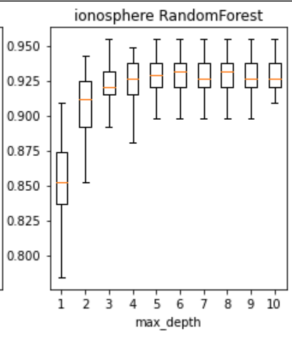
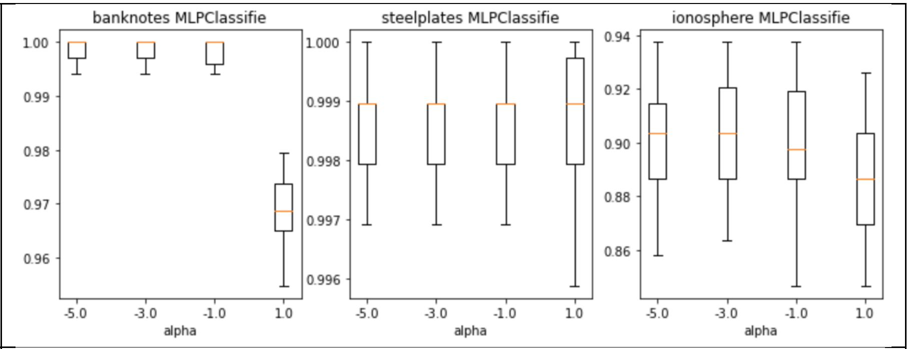
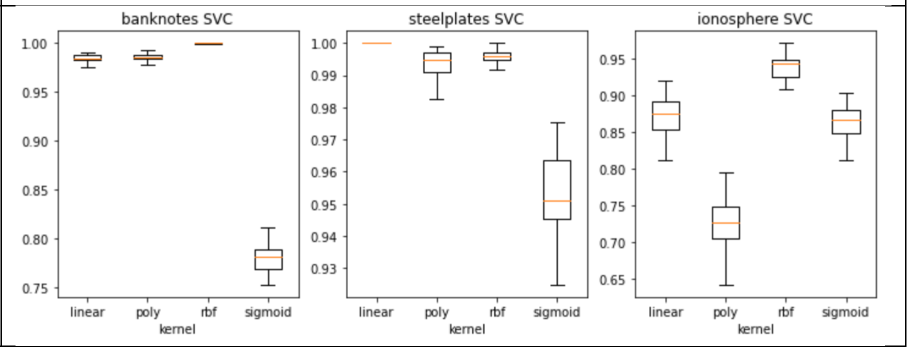
Trying Out a Variety of Classification Algorithms

Isabelle Southon, 300597453
Data302

Part 1

Table 1 – Boxplots on the classifier accuracy versus parameter values

	banknotes	**steelplates**	**Ionosphere**
KNeighbors Classifier			
Logistic Regression			
Decision Tree Classifier			
Random Forest Classifier			

MLP Classifier	
SVC Classifier	

Summary Tables

Table 2 – Lowest mean test errors

ID		**banknotes**	**steelplates**	**ionosphere**
1	**KNeighbors Classifier**	0.00	0.02	0.15
2	**Logistic Regression**	0.02	0.00	0.12
3	**Decision Tree Classifier**	0.02	0.00	0.11
4	**Random Forest Classifier**	0.01	0.02	0.07
5	**MLP Classifier**	0.00	0.00	0.10
6	**SVC Classifier**	0.00	0.00	0.06

Table 3 – Corresponding hyperparameters

ID		**banknotes**	**steelplates**	**ionosphere**
1	**KNeighbors Classifier (Hyperparameter = K)**	[1, 2, 3, 4, 5]	1	1
2	**Logistic Regression (Hyperparameter = C)**	5	5	0.1
3	**Decision Tree Classifier (Hyperparameter = max_depth)**	9	[6, 7, 8, 9]	2
4	**Random Forest Classifier (Hyperparameter = max_depth)**	[8, 9, 10]	10	[6, 8]
5	**MLP Classifier (Hyperparameter = “alpha”)**	[1 x 10^{-5}, 1 x 10^{-3}, 0.1]	[1 x 10^{-5}, 1 x 10^{-3}, 0.1, 10]	[1 x 10^{-5}, 1 x 10^{-3}]
6	**SVC Classifier (Hyperparameter = “kernel”)**	‘rbf’	‘linear’	‘rbf’

The model that performed the best on the banknotes dataset is the KNN classifier, MLP classifier, and the SVC classifier as they all captured mean test errors of 0. The reason why these models may have performed better than the logistic regression, the decision tree classifier and the random forest classifiers is because the banknotes dataset may have a complex decision boundary that the KNN, MLP, and SVC models capture best due to their non-linear natures. The KNN classifier obtained the best mean test error when the hyperparameter 'K' was set as 1, 2, 3, 4, or 5, which means that all values of 'K' that were provided obtained the best mean test error. The MLP classifier performed best when 'alpha' was set to -5, -3, & -1 (also seen as 1×10^{-5}, 1×10^{-3}, 0.1 in Table 3), meaning that the model performed best with lower values of 'alpha'. Additionally, the SVC captured the best mean test error when the 'kernel' was set to rbf. The rbf 'kernel' is useful when the decision boundary between classes is non-linear or complex as mentioned.

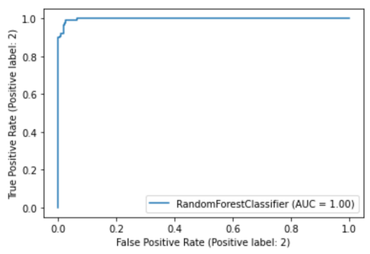
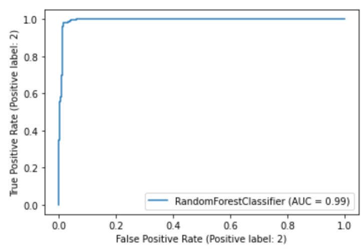
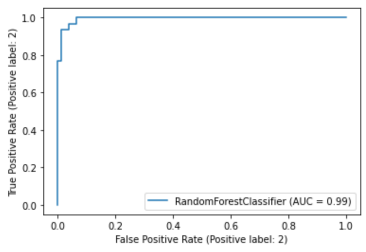
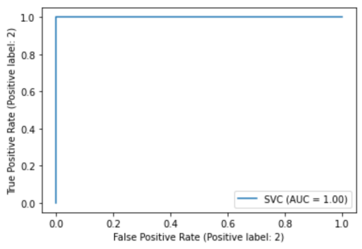
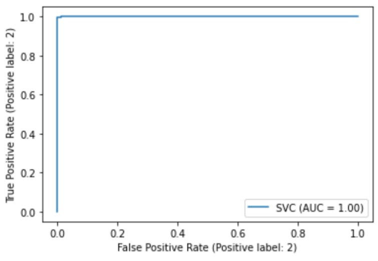
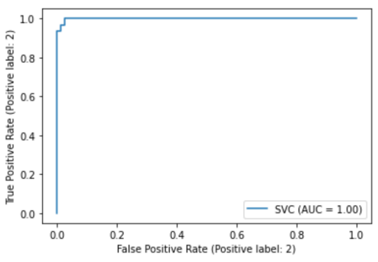
For the steelplates dataset, the models that obtained the lowest mean test errors are logistic regression, decision tree classifier, MLP classifier, and SVC classifier as they all give mean test errors of 0. This may indicate that the steelplates dataset has a simpler decision boundary that can be captured using these models. The logistic regression model got the lowest mean test error with the hyperparameter 'C' set to 5, meaning that regularization was moderate. The decision tree classifier got the lowest mean test error with 'max_depth' set as 6, 7, 8, & 9. The MLP classifier got the lowest mean test error with 'alpha' as -5, -3, -1, & 1 (or 1×10^{-5}, 1×10^{-3}, 0.1, 10 as seen in Table 3), thus meaning that at any 'alpha' we provided, the lowest mean test error was obtained. A linear 'kernel' for the SVC classifier gave the lowest mean test error for the steelplates dataset, suggesting that classes could be separated using a linear hyperplane, reiterating that steelplates may not have a complex decision boundary like the banknotes dataset has.

Lastly, the model that obtained the lowest mean test error for the ionosphere dataset is the SVC classifier, with a mean test error of 0.06. The reason for this might be because there is a complex decision boundary in the ionosphere dataset that the SVC classifier can separate well when the hyperparameter 'kernel' is set to rbf.

KNN might not be as sensitive as other classifiers are, as we can always use K = 1 for all datasets. However, the random forest classifier might be more sensitive as for each dataset a different value of 'max_depth' is almost used (max_depth = 10 as an exception), therefore being a more sensitive classifier to the control hyperparameter. The logistic regression model may be sensitive to its hyperparameter 'C', as the banknotes and steelplates dataset both use C = 5 to obtain the lowest mean test error, whereas the ionosphere dataset uses C = 0.1. This may be because the ionosphere dataset has a simpler decision boundary than the other two datasets, meaning that the model on the ionosphere dataset does not need as much regularization as the model does on the banknotes and steelplates dataset. The decision tree classifier on the three different datasets reiterates this, as a 'max_depth' of 9 got the lowest mean test error for the banknotes and steelplates dataset. This suggests that the decision tree classifier is sensitive to 'max_depth', as the banknotes and steelplates datasets are more complex than the ionosphere dataset and 'max_depth' needs to be adjusted accordingly for this complexity. MLP might not be as sensitive as the decision tree classifier, as most values of 'alpha' can be used to obtain the lowest mean test errors for all datasets.

Table 4 – Comparing AUC scores

	banknotes	**steelplates**	**ionosphere**

Random Forest Classifier	AUC = 1 	AUC = 0.99 	AUC = 0.99 
SVC	AUC = 1 	AUC = 1 	AUC = 1 

Both the random forest classifier and the SVC classifier seem to separate between the positive and negative classes well in all three datasets, however, the SVC model distinguishes between the classes the best out of these models. This is because, for the banknotes, steelplates, and ionosphere datasets, the AUC score obtained is 1 for the SVC model, meaning that this model perfectly distinguishes between positive and negative classes. For the banknotes dataset, the random forest classifier separates the positive and negative classes better than it does for the steelplates and ionosphere datasets.

Part 2

K-nearest-neighbors aims to find a neighborhood of the closest training points to help us predict the class label of a data point. For the banknotes dataset, when the parameter k (the size of the neighbourhood) is changed, the test accuracy changes. On the test set when k=3 or k=5, a perfect accuracy score of 1.0 is obtained, and is therefore perfectly predicting the class label of all instances in the test set. However, when k=1 or k=7, the accuracy score on the test set decreases. When k=1 or k=7, the classifier is almost predicting class labels for instances perfectly with scores of 0.9985 (99.8%) and 0.9941 (99.4%) respectively. We don't talk about overfitting or underfitting in this case as these are the accuracy scores on the test set, and we have not fit the model to the training set.

Part 3

The decision tree algorithm is a supervised algorithm that can classify both numerical and categorical features and can be used for classification or regression tasks. It works by following a flowchart structure to determine a class label for a data point. An internal node represents a feature, the edge represents a condition or decision rule, and each leaf node represents an outcome. At each node there is a feature, if it satisfies the condition on one edge, it will go one way, if it doesn't it will go a different way. It will do this until it reaches a leaf node (a node with no branches stemming off it) which has a class label, and this is where a decision is made and the data point is classified. For complexity control, it is important to control the hyperparameter "max_depth", which controls the size of the decision tree. Overfitting can occur if the size of the decision tree is too big. Additionally, the minimum partition size of each node can be controlled to prevent overfitting, this is done by adding the hyperparameter "min_partition_size". To figure out the depth of a tree, you count the number of edges there are on the path to get to a node, note that the first node starts at a depth of 0. A diagram of this can be viewed below, a decision tree that classifies whether someone is at risk of a heart attack.

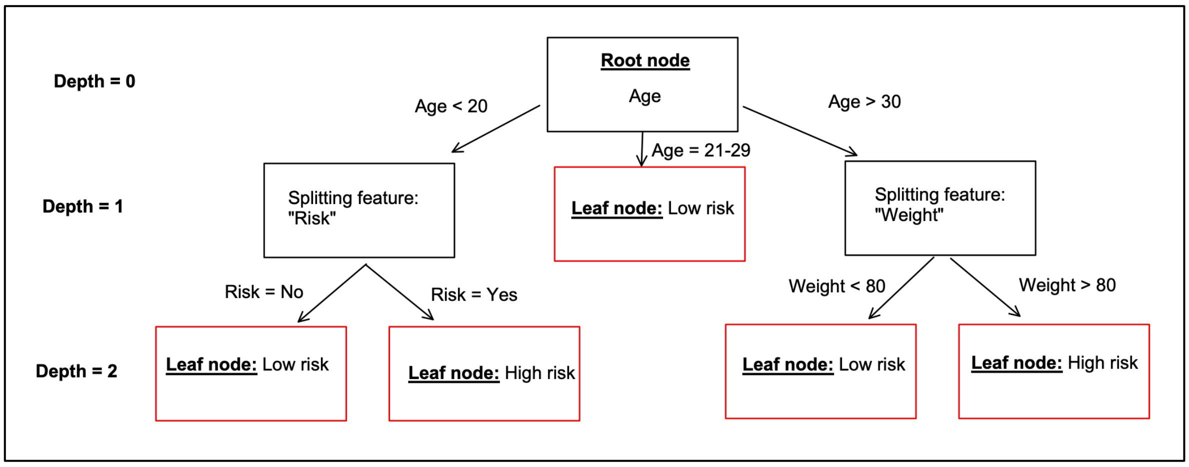
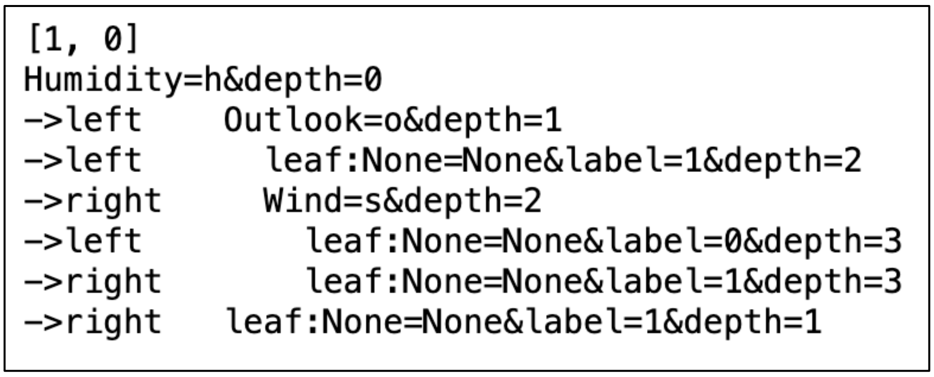


Figure 1: A simple flowchart diagram to understand a decision tree model

According to the decision tree above, if someone was older than 30 years old, with a weight greater than 80kg, they would be considered high risk for a heart attack. Note that 'Age' is in years, 'Risk' is defined as whether there is hereditary risk of a heart attack, and 'Weight' is measured in kilograms.

For instance 1, where [Outlook: 'o', Humidity: 'n', and Wind: 'w'], tennis will be played as a class label of 1 is obtained. The learned decision tree makes its classification decision for instance 1 by starting at the root node. The root node of the decision tree checks the 'Humidity' feature at depth 0. Since instance 1 has Humidity 'n', it follows that edge to the right where Humidity is 'n'. A leaf node is then met at depth 1, and a class label of 1 is assigned to instance 1. The decision tree predicts that instance 1 will play tennis when its Humidity is 'n'.

For instance 2, where [Outlook: 's', humidity: 'h', and wind: 's'], tennis will not be played as a class label of 0 is obtained. The learned decision tree makes its classification decision for instance 2 by starting at the root node. The root node of the decision tree checks the 'Humidity' feature at depth 0. Since instance 2 has Humidity 'h', it follows the branch to the left where Humidity is 'h'. The next child node evaluates the feature 'Outlook' at depth 1, which is 's' for instance 2, so it follows this edge to the right. Next, the instance checks the 'Wind' feature, and since it has 'Wind' as 's', it follows the left edge, reaching a leaf node. This instance reaches a leaf node that predicts the class label of 0 for this instance. The decision tree predicts that instance 2 will not play tennis when its Humidity is 'h', its Outlook is 's' and Wind is 's'.



```
[1, 0]
Humidity=h&depth=0
->left     Outlook=o&depth=1
->left       leaf:None=None&label=1&depth=2
->right      Wind=s&depth=2
->left         leaf:None=None&label=0&depth=3
->right        leaf:None=None&label=1&depth=3
->right    leaf:None=None&label=1&depth=1
```

Figure 2: Decision tree classifier predicted classes and string representation

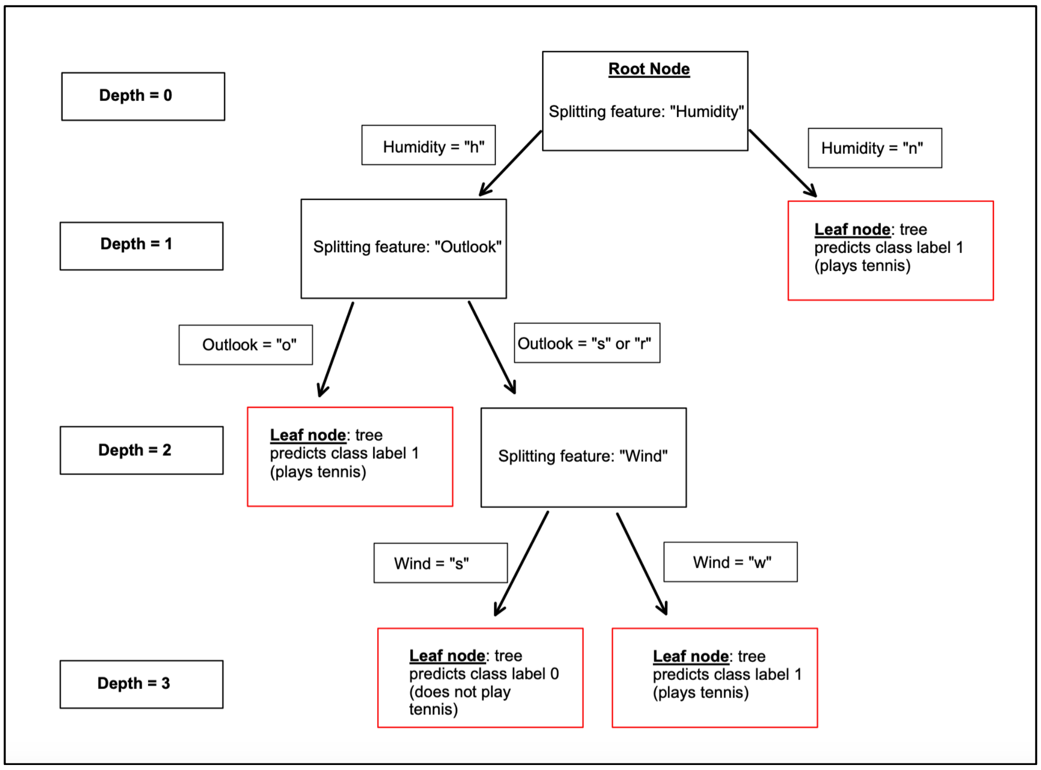
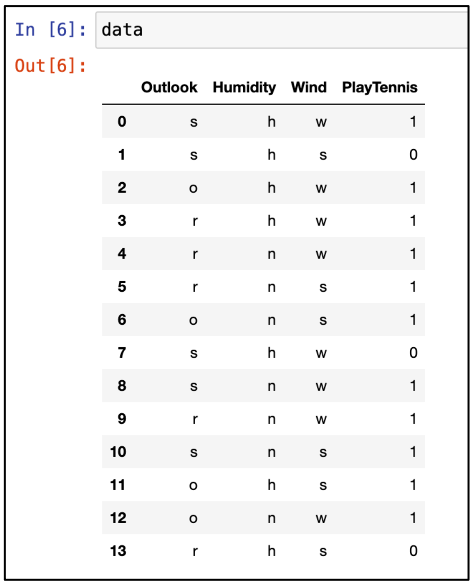

Depth = 0
Root Node
Splitting feature: "Humidity"
Humidity = "h"
Humidity = "n"
Depth = 1
Splitting feature: "Outlook"
Leaf node: tree predicts class label 1 (plays tennis)
Outlook = "o"
Outlook = "s" or "r"
Depth = 2
Leaf node: tree predicts class label 1 (plays tennis)
Splitting feature: "Wind"
Wind = "s"
Wind = "w"
Depth = 3
Leaf node: tree predicts class label 0 (does not play tennis)
Leaf node: tree predicts class label 1 (plays tennis)

Figure 3: The structure of my decision tree

The feature of Humidity is selected by the root node of the constructed decision tree, and the value 'h' is its condition for making a decision. The decision tree will follow a different branch depending on whether the condition of humidity being 'h' is satisfied or not.

Entropy and information gain calculations:



```
In [6]: data
Out[6]:
```

	Outlook	Humidity	Wind	PlayTennis
0	s	h	w	1
1	s	h	s	0
2	o	h	w	1
3	r	h	w	1
4	r	n	w	1
5	r	n	s	1
6	o	n	s	1
7	s	h	w	0
8	s	n	w	1
9	r	n	w	1
10	s	n	s	1
11	o	h	s	1
12	o	n	w	1
13	r	h	s	0

Figure 4: Using the dataset to calculate the entropy

The entropy of the dataset associated with the root node of 'Humidity' is 0.4926 and the information gain achieved by the corresponding split at 'Humidity' is 0.25699. The calculations used to get the entropy, conditional entropy, and information gain can be seen below.

Entropy calculation:

$$H(Y) = -\sum_{I=1}^{K} P(Y = yi) log_2 P(Y = yi)$$

If Y = "PlayTennis":

$$P(Y = 1) = \frac{11}{14}$$

$$P(Y = 0) = \frac{3}{14}$$

The entropy of "PlayTennis" is calculated by:

$H(Y) = -11/14log_2 11/14 - 3/14log_2 3/14$
$= 0.27337 + 0.47623$
$= 0.7496\ (4dp)$

Before the splitting feature is used, this represents the uncertainty in predictions of the class label of the target variable "PlayTennis".

Conditional entropy:

$$H(Y|X) = -\sum_{J=I}^{V} P(X = xj) \sum_{i=1}^{k} P(Y = yi \mid X = xj) log_2 P(Y = yi \mid X = xj)$$

If the "Humidity" feature is used to partition the dataset, there will be 2 partitions of size 7. Where, P(Humidity) = h = 7 and P(Humidity) = n = 7.

When Humidity = h, there are 4 observations with PlayTennis = 1 and 3 observations with PlayTennis = 0. For Humidity = n, there are 7 observations with PlayTennis = 1 and 0 observations with PlayTennis = 0.

P(PlayTennis=1 | Humidity = h) = 4/7
P(PlayTennis=0 | Humidity = h) = 3/7
P(PlayTennis=1 | Humidity = n) = 7/7
P(PlayTennis=0 | Humidity = n) = 0/7

The conditional entropy is calculated by:

$$H(Y|X) = -\frac{7}{14}\left(\frac{4}{7} log_2 \frac{4}{7} + \frac{3}{7} log_2 \frac{3}{7}\right) - \frac{7}{14}\ (1log_2 1 + 0log_2 0) =$$
$$-\frac{7}{14}(-0.46134 - 0.52388) - \frac{7}{14}(0) = 0.4926$$

Information gain:

$IG(X) = H(Y) - H(Y|X)$

The information gain is calculated by:

$H(Y) - H(Y|X) = 0.7496 - 0.4926 = 0.25699$

Where X represents “Humidity”, $IG(X) = 0.25699$

The information gain of Humidity is greater than 0, which means we prefer the spilt because the information gain represents a reduction in uncertainty. After using the Humidity feature to split the original dataset, the entropy that determines the label of “PlayTennis” (the target variable) is lower. Knowing the value of the “Humidity” feature for a new observation helps to predict if the new observation plays tennis as the uncertainty of the outcome is reduced.