VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

School of Engineering and Computer Science

DATA302: Techniques in Machine Learning (T1/2024)

Course Coordinator: Dr. Qi Chen

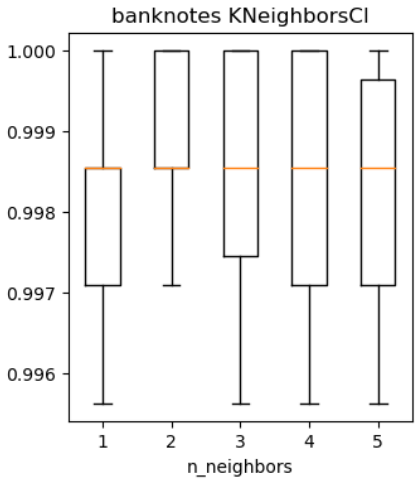
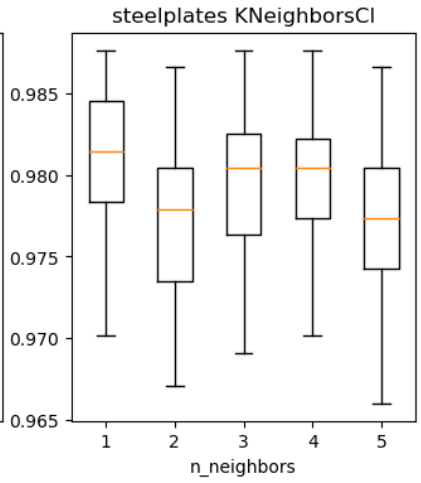
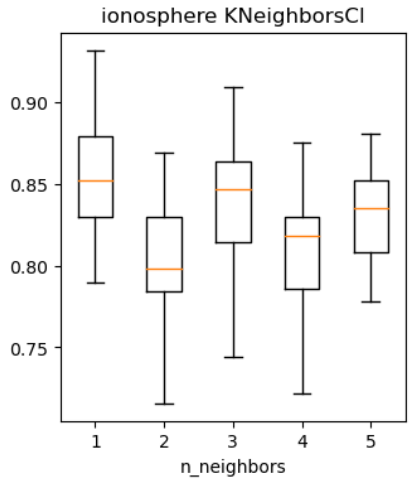
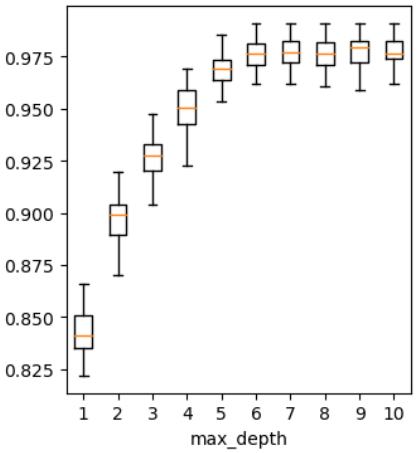
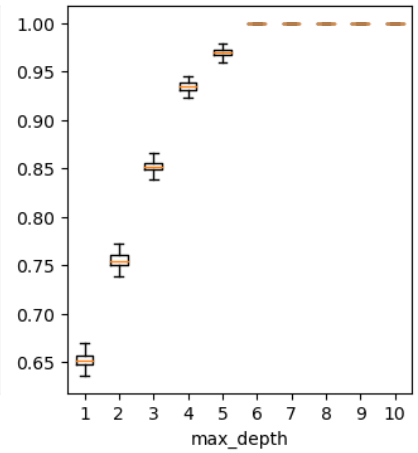
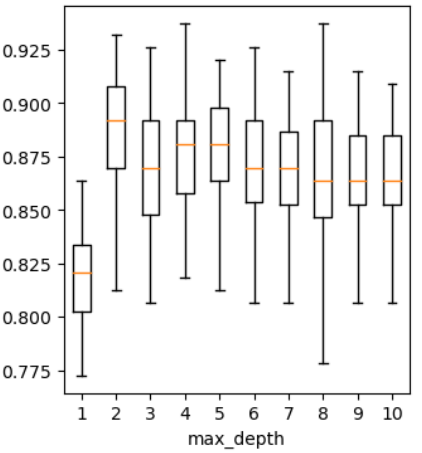
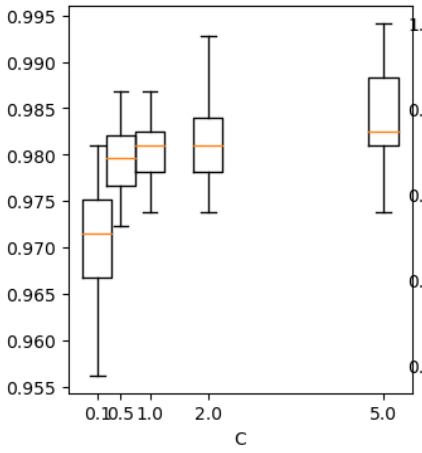
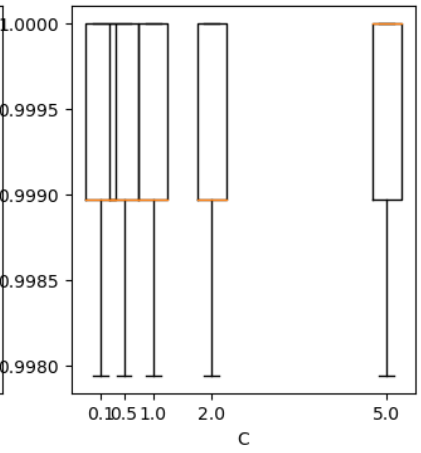
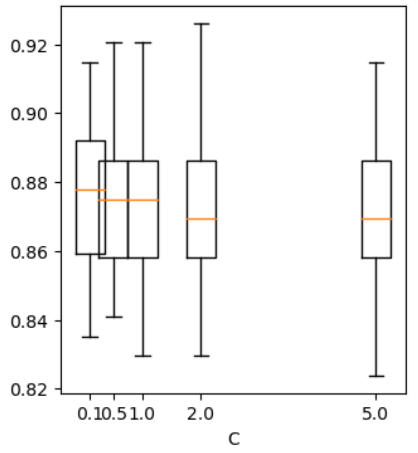
Lecturers: A/Prof Aaron Chen, and Dr Bach Nguyen

Assignment 1

Student Name: PHAM NGOC KHANH DUNG (Junn Pham)

Student ID: 300538618

1. Classification using the SKLearn library

Classifier	banknotes	steel-plates-fault	ionosphere
KNN			
Decision Tree			
Logistic Regression			

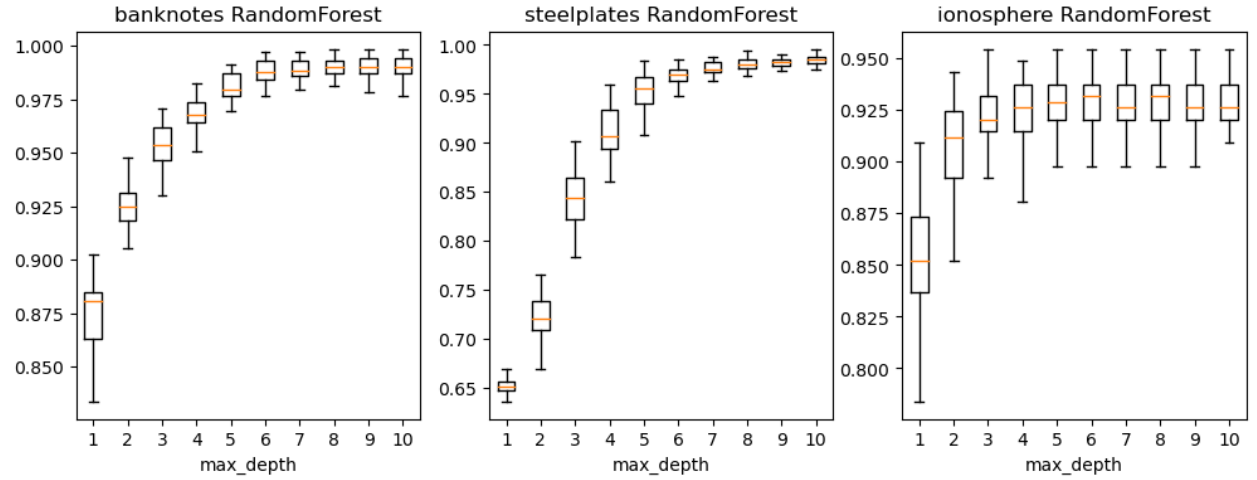
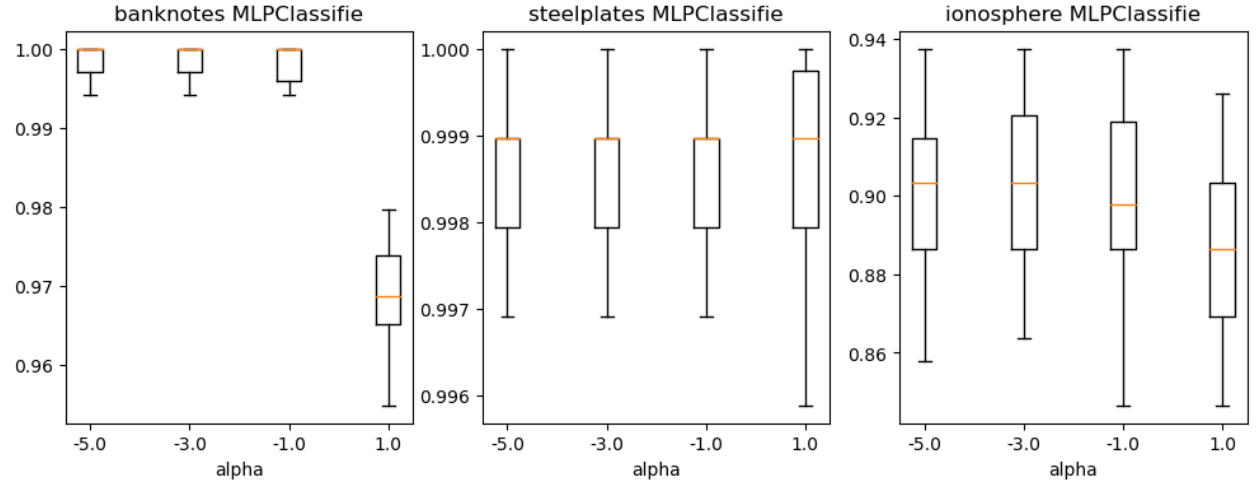
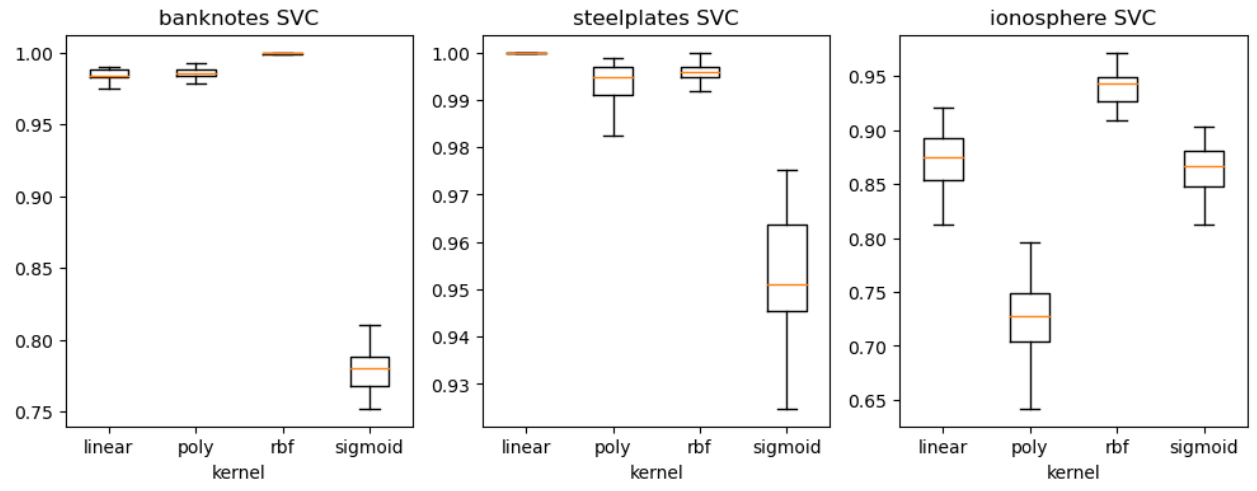
Random Forest	
MLP	
SVC	

Table i – The lowest mean test errors

	banknotes	steel-plates-fault	ionosphere
KNN	0.001	0.019	0.146
Decision Tree	0.022	0.000	0.114
Logistic Regression	0.016	0.000	0.124
Random Forest	0.010	0.015	0.072
MLP	0.002	0.001	0.098
SVC	0.001	0.000	0.060

Table ii – The corresponding hyperparameter values for obtaining the lowest mean test errors obtained in Table 1

	Hyperparameter	banknotes	steel-plates-fault	ionosphere
KNN	n_neighbors	2	1	1
Decision Tree	max_depth	10	6	2
Logistic Regression	C	5.0	5.0	0.1
Random Forest	max_depth	9	10	6
MLP	alpha	0.001	10.0	0.001
SVC	kernel	RBF	linear	RBF

The best performance of classifiers for each dataset:

For the banknotes dataset, KNN classifier with n-neighbors = 2 and SVC classifier with radial basis function (RBF) kernels generated the lowest mean test errors, which equal 0.001. MPL classifier with alpha = 0.001 also achieved a very low mean test error of 0.002.

For the steel-plates-fault dataset, three classifiers including the Decision Tree with max_depth = 6, Logistic Regression with C = 5.0, and SVC with linear kernel had the lowest mean test errors of 0.000. MPL classifier with alpha = 10.0 achieved the second lowest mean test error of 0.001.

For the ionosphere dataset, SVC with RBF kernel and a mean test error of 0.060 outperformed other classifiers, followed closely by Random forest with max_depth = 6 and a mean test error was 0.072.

Overall, SVC demonstrated the strongest performance across different datasets, consistently achieving the lowest mean test errors. Linear kernels performed well in the steel-plates-fault dataset, while RBF kernels were more suitable for the other datasets.

How sensitive these classifiers are to the control hyperparameters:

KNN is called a lazy learner due to its learning approach. It does not learn the data immediately, instead, it stores data sets and performs actions on the data set at the time of classification. It tries to predict the outcome based on calculating the distance of the test data with all the training points and selecting the k number of points that are closest to the chosen test data. Because of that, all three data sets are not sensitive to the change in the complexity control hyperparameter, because technically we just select more nearest neighbors to the chosen test data.

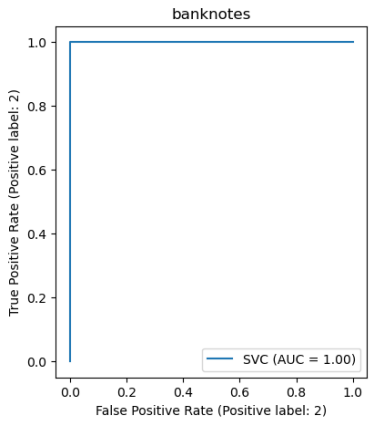
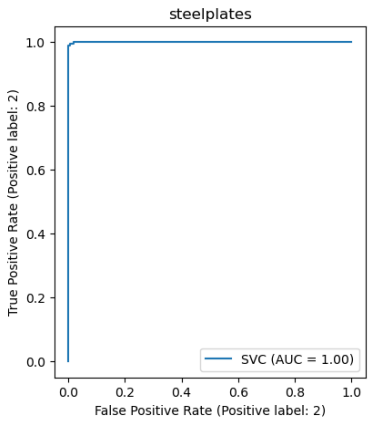
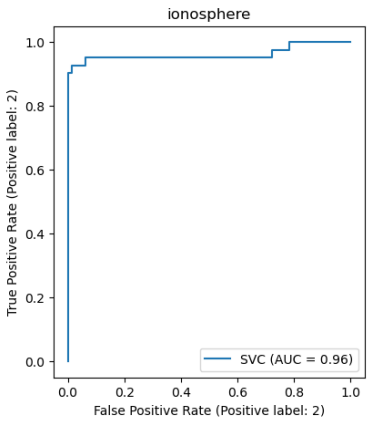
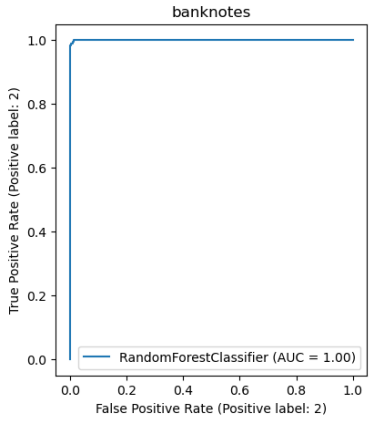
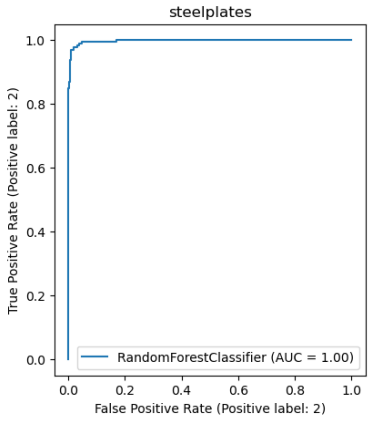
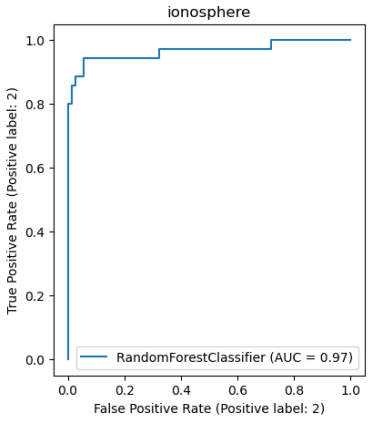
For decision trees, the maximum depth parameter (max_depth) controlled the complexity of the model. Optimal depths varied across datasets, indicating the importance of tuning this parameter. Decision Trees is one of the most powerful classifiers as the more layers (higher max_depth) it has, the better the prediction will be. This trend can easily be witnessed in all three data sets. So, we can conclude that the decision trees classifier is sensitive to its control hyperparameters.

In logistic regression, the labels are constrained to discrete values derived from the datasets. The boxplots of the three data sets show that the hyperparameter is not particularly sensitive. Changes across different hyperparameters are minimal and insignificant.

Random forest technically a many sets of decision trees. It creates many sets of decision trees randomly from the training set and collects votes from its decision trees to predict the final decision. It is an extremely precise classification but time-consuming when performing. therefore, its hyperparameter is sensitive to any change and we need to be more careful while tuning the model.

For the MLP classifier, increasing in Alpha may fix high variance which is a sign of overfitting by encouraging smaller weights, thereby producing a decision boundary plot with reduced curvatures. Or we can say that the higher the Alpha is, the less accurate the prediction of the MLP classifier is likely to be. The hyperparameter of MLP is sensitive to all data sets, elevated Alpha levels tend to induce overfitting, leading to poorer predictions.

SVC's choice of kernel function impacted performance significantly. It can drastically influence the shape of the decision boundary and the classifier's ability to separate classes. From the experiments, Linear kernels performed well in the steel-plates-fault dataset, while radial basis function (RBF) kernels were more suitable for the other datasets.

Classifier	Hyper-parameter and value	banknotes	steel-plates-fault	ionosphere
SVC	Kernel = RBF			
Random Forest	Max_depth = 5			

Based on the ROC curves in the image, both SVC and RandomForest classifier appear to perform well on all three datasets with an Area Under the Curve (AUC) close to 1. However, the SVC classifier with RBF kernel appeared to be more efficient than the RandomForest classifier because of its higher AUCs values

- Banknotes: The SVC classifier achieved a perfect score (AUC = 1.00), and the RandomForest classifier also gained a nearly perfect score. This means The SVC classifier was able to distinguish between genuine and counterfeit outcomes without any errors while the RandomForest classifier still had a very small number of errors in its prediction.
- Steel-plates-fault: Again, both classifiers performed very well. SVC achieved a perfect score, while RandomForestClassifier had a minor misclassification.
- Ionosphere: Once more, both classifiers achieved a high AUC. In this dataset, SVC performed slightly better than RandomForest classifier.

Overall, both SVC and RandomForestClassifier achieved very good performance on all three datasets. However, it is difficult to choose which one is better for the three datasets, as there are some additional factors that need to be considered when choosing between SVC and RandomForestClassifier:

- Training time: RandomForest classifier typically trains faster than SVC.
- Interpretability: SVC is generally considered more interpretable than RandomForest classifier, as it can provide the weights assigned to different features.

2. Implement a simple KNN classifier

(3) Test your classifier

With a k value of 3, the KNN classifier achieved an impressive 100% accuracy on the test set. This indicates that all instances were correctly classified using the given parameters and Euclidean distance metric, demonstrating the model's ability to generalize well to new data and capture relevant patterns from the dataset. However, it's important to remain vigilant for overfitting, especially when evaluating model performance on the same dataset used for training.

(4) Experimentation

Results of the experiment:

Value of k	Accuracy
1	0.999
3	1
5	1
7	0.994
9	0.996
11	0.996

The result of the experiment suggests that the test accuracy of the KNN classifier varies with different values of k. As the value of k increases from 1 to 3, the accuracy tends to increase, reaching its peak at k=3. At this stage, a few nearest neighbors help improve the classifier's performance. However, as k continues to increase beyond 3, the accuracy remains unchanged or even slightly decreases. This phenomenon indicates that including too many neighbors leads to overfitting, where the model captures noise instead of underlying patterns. In this case, a moderate value of k, such as 3 or 5, seems to yield the best performance.

3. Implement a basic Decision Tree classifier

a. Brief explanation of how the decision tree learning algorithm works.

Explanation:

Decision tree classification is one of the most powerful classifications. Technically it works like a flowchart, each node of the tree will represent a decision point that is split into two new leaf nodes. Each node performs a decision

which means it can be a leaf that leads to the final decision, or it can be a decision node and split into two decision nodes. It splits recursively until it finds the leaf. The top node is called the root node, any other nodes are called the decision node, and the final decisions are referred to leaf. The complexity control hyperparameter of the decision tree is called max_depth, it shows how deep the tree can be. The reason why the decision tree is one of the best classifications is because the more layers it has, the better the prediction will be. So that proves why all these three data sets are sensitive to hyperparameters. When the hyperparameter is higher, all data sets provide a better accuracy prediction, we can witness this trend in all three data sets.

Pseudo-code:

Input:

- Train dataset: S= {($Outlook_1$, $Humidity_1$, $Wind_1$), ($Outlook_2$, $Humidity_2$, $Wind_2$),... }
- Attribute set: A= {$PlayTennis_1$, $PlayTennis_2$, $PlayTennis_3$, ...}
- Threshold setting *max_depth*, *min_samples_split*.
- Attribute selection method: The decision tree algorithm selects the attribute for splitting based on Information Gain, a measure indicating attribute importance. The highest Information Gain attribute guides the split at each node.

Method:

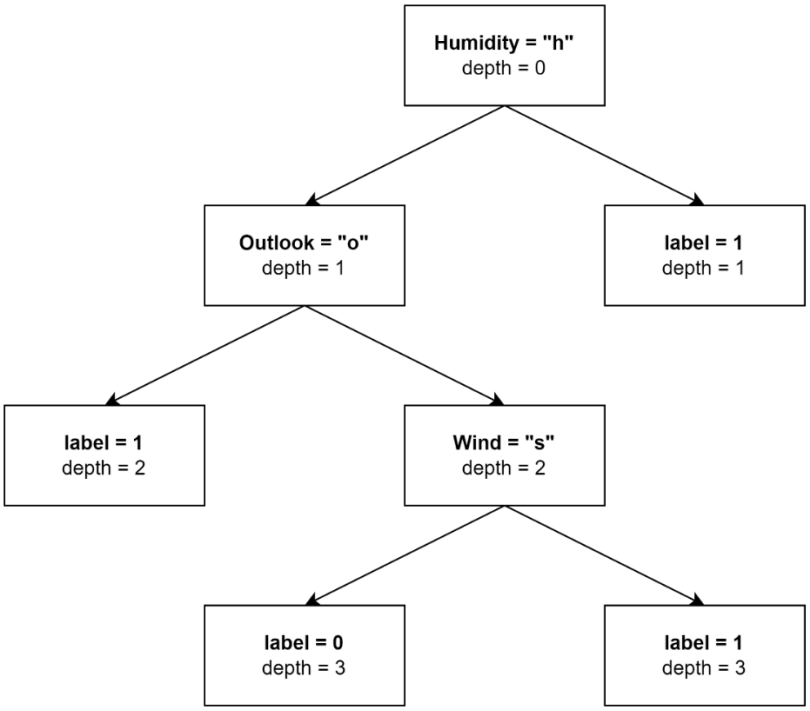
- Create node N
- **If** *depth* is equal to *max_depth* or there's only one unique class in y or len(y) < min_samples_split, **then return** N as a leaf node with the majority label.
- **If** not, then find the best feature and value to split the data.
- Split the data based on the best feature and value.
- Recursively build the left and right subtrees.
- Return the current node.

b. The predicted class labels for the two test data instances. Explain how your learned decision tree makes its classification decisions with respect to each test data instance.

Test data		Predicted class label	Explanation
Test 1	Outlook = “o” Humidity = “n” Wind = “w”	1	– Root Node (Humidity=h): The humidity is not high (Normal), so we move to the right subtree. – Subtree (Outlook=o): Since the outlook is Overcast, we reach a leaf node that predicts playing tennis (label 1). – The humidity is not high (Normal), which is favourable. – The outlook is Overcast, which historically has been associated with good tennis-playing conditions.

Test 2	Outlook = “s” Humidity = “h” Wind = “s”	0	– Root Node (Humidity=h): The humidity is high, so we move to the left subtree. – Subtree (Outlook=o): Since the outlook is not Overcast, we move to the right subtree. – Subtree (Wind=s): The wind is strong, leading to a leaf node that predicts not playing tennis (label 0). – The humidity is high, indicating potentially unfavourable playing conditions. – The outlook is Sunny, which historically may not be ideal for playing tennis, especially when combined with high humidity. – The strong wind adds to the unfavourable conditions for playing tennis.

c. Diagram of the decision tree (max_depth=3, min_samples_split=2):



d.

Formular: Entropy = $-\sum_{i=1}^{k} p_i log_2(p_i)$

Current root node: **h**

Total sample size: 14

Samples where Play Tennis = 1 (play tennis): 11

Samples where Play Tennis = 0 (do not play tennis): 3

$P_1 = \frac{4}{7}$ and $P_2 = \frac{3}{7}$

Entropy = $\left(-\left(\frac{11}{14}\right) \times log_2\left(\frac{11}{14}\right)\right) - \left(\left(\frac{3}{14}\right) \times log_2\left(\frac{3}{14}\right)\right) = 0.7495$

Information gain = H(Y) – H (Y |X) = 0.7495 – $\frac{3}{7}$ = 0.321